UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO DE MULTISDICIPLINAR

LUCAS RIBEIRO FERREIRA
WILSON DE OLIVEIRA MENDONÇA

WppScrapper: Uma ferramenta para a extração de mensagens do WhatsApp

Prof. Filipe Braida do Carmo, D.Sc.
Orientador

Nova Iguaçu, Maio de 2021

WppScrapper: Uma ferramenta para a extração de mensagens do WhatsApp

Lucas Ribeiro Ferreira

Wilson de Oliveira Mendonça

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Lucas Ribeiro Ferreira

_____

Wilson de Oliveira Mendonça

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Bruno José Dembogurski, D.Sc.

_____

Profª. Natália Chaves Lessa Schots, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Maio de 2021

Emitido em 04/05/2021

DOCUMENTOS COMPROBATÓRIOS Nº 5622/2021 - CoordCGCC (12.28.01.00.00.98)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 04/05/2021 14:59)
BRUNO JOSE DEMBOGURSKI
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: 2124964

(Assinado digitalmente em 04/05/2021 14:59)
FILIPE BRAIDA DO CARMO
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: 3929524

(Assinado digitalmente em 04/05/2021 15:00)
NATALIA CHAVES LESSA SCHOTS
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: 2143584

(Assinado digitalmente em 04/05/2021 15:11)
WILSON DE OLIVEIRA MENDONCA
DISCENTE
Matrícula: 2012785301

(Assinado digitalmente em 04/05/2021 15:11)
LUCAS RIBEIRO FERREIRA
DISCENTE
Matrícula: 2012780197

Para verificar a autenticidade deste documento entre em https://sipac.ufrrj.br/documentos/ informando seu número: **5622**, ano: **2021**, tipo: **DOCUMENTOS COMPROBATÓRIOS**, data de emissão: **04/05/2021** e o código de verificação: **649a400943**

Agradecimentos

Lucas Ribeiro Ferreira

Gostaria de agradecer primeiramente aos principais responsáveis pela minha formação acadêmica e pessoal. Àqueles que fizeram sacrifícios pessoais para poder prover os recursos necessários a minha educação desde meu nascimento até os dias atuais. Àqueles que me proveram tudo o que me era necessário para que eu pudesse focar em minha formação intelectual, que me proveram alicerce emocional e material durante toda a minha vida. Aos que moldaram meu caráter e estiveram presente sempre que precisei, meus pais, Eloecy e Joaquim.

Apesar de os principais, não apenas eles foram determinantes. Então gostaria de estender esses agradecimentos aos que forneceram os alicerces necessários a esses responsáveis pelo meu. Forneceram a base de toda a família Ribeiro e a família Ferreira, que em momentos diferentes estiveram presentes de formas diferentes, porém foram todos, a sua maneira, determinantes para que eu pudesse estar entregando meu trabalho de conclusão de curso. Então meu agradecimento especial aos meus avós maternos, Lecy e Eloy e meus avós paternos, Benita e Armando.

Em decorrência de acontecimentos recentes, gostaria de fazer um agradecimento especial a minha avó Benita, que faleceu no inicio do ano em decorrência da crise pandêmica e política vivida em nosso país. Minha querida avó sempre foi especialmente carinhosa comigo e é um bom exemplo de como cada pessoa de minha família foi determinante em minha formação. Obrigado minha queria avó por ter estado comigo no intervalo entre a escola e o pré-vestibular, me fornecendo um almoço delicioso e repleto do seu amor e em outras infinidades de momentos.

Estendo ainda meus agradecimento a Beatriz, minha queria esposa, que conheci nos corredores da universidade e desde então tem sido minha companheira na jornada da vida, tornando-a mais alegre, mais amável e mais divertida. Beatriz foi fundamental ao fazer a cobrança na medida certa para que eu me dedicasse ao trabalho que estou entregando. Obrigado Bia por dividir essa vida comigo e me amar da forma que me ama, a você só posso garantir a reciprocidade de cada sentimento.

Dentre a infinidade de pessoas que atravessaram a minha vida me fornecendo ensinamentos valiosos, ajudas necessárias e companheirismo, gostaria de destacar e agradecer a minha amiga Letícia, que muito me ensinou e ensina ainda hoje, inclusive ajudando em revisões pontuais desse texto. Dentre essas pessoas também gostaria de agradecer aos meus amigos que me forneceram oportunidades de trabalho e de aprendizado profissional e pessoal Guilherme, André e Higor, estendendo a cada colega e amigo que fiz trabalhado para e com eles.

Agradeço ainda ao corpo docente da Universidade Federal Rural do Rio de Janeiro, profissionais dedicados que se esforçam para entregar um ensino de altíssima qualidade aos seus alunos. Um agradecimento especial ao professor Filipe Braida, que me orientou no curso desse trabalho com a devida paciência e cobrança, além de ter me inspirado com a dedicação que demonstrou nas aulas que ministrou ao decorrer da minha formação.

Apesar de desejar agradecer nominalmente a muitas outras pessoas, encerro agradecendo a todos aqueles que no passado lutaram e que ainda lutam para que eu pudesse ter acesso a uma universidade gratuita de qualidade e a toda a sociedade brasileira, que financiou meus estudos. Obrigado pelo investimento feito em mim e em meus colegas, me comprometo a retribuir.

Wilson de Oliveira Mendonça

Primeiramente gostaria de agradecer aos meus pais Heraldo e Zilma por sempre terem me dado apoio em tudo que fiz desde que nasci, por terem feito sacrifícios para me dar condições necessárias para estudar. São sem duvidas os principais responsáveis pela minha formação pessoal e acadêmica e são meus maiores exemplos.

Gostaria de agradecer ao corpo de docentes do curso de ciência da computação da Universidade Federal Rural do Rio de Janeiro por proporcionar um excelente curso, em especial ao professor Filipe Braida por todo suporte, compreensão e paciência.

Gostaria de agradecer também ao meu colega Lucas Ferreira por toda a ajuda que me deu e pela dedicação a este trabalho.

RESUMO

WppScrapper: Uma ferramenta para a extração de mensagens do WhatsApp

Lucas Ribeiro Ferreira e Wilson de Oliveira Mendonça

Maio/2021

Orientador: Filipe Braida do Carmo, D.Sc.

O WhatsApp é o aplicativo de troca de mensagens instantâneas mais utilizado no Brasil. Seu principal uso é de cunho privado entre os usuários, mas estudos recentes vêm mostrando o aumento de importância da plataforma como fonte de informação, incluindo desinformação, e como ferramenta para organização e agitação de eventos de grande relevância social, como greves e protestos. Nesse contexto, vem sendo demonstrado a importância de estudar as mensagens trocadas dentro da plataforma e, através desses estudos, ajudar a compreender as diferentes determinações da realidade social. Muitos trabalhos têm realizado a tarefa de extração de mensagens da plataforma fazendo uso de técnicas de *WebScrapping* ou até descriptografando as mensagens contidas no banco de dados de um *smarthphone*. Desta forma, o presente trabalho busca colaborar com essa tarefa apresentando uma ferramenta que é capaz de se conectar ao servidor do WhatsApp e baixar todas as mensagens trocadas por uma conta, possuindo uma interface de programação simples e objetiva que permite seu uso para a implementação de diferentes formas de interfaces de usuário, sendo agnóstica a qualquer que seja essa forma. Ainda é apresentado aqui uma aplicação com interface gráfica de usuário para computadores domésticos, implementada usando a *API* citada anteriormente, que se propõe se de fácil uso para auxiliar pesquisadores e jornalistas a realizarem seus trabalhos mais facilmente. O objetivo que se espera alcançar com essas ferramentas é que mais estudos sejam feitos e, portanto, que possam compreender melhor nossa sociedade.

ABSTRACT

WppScrapper: Uma ferramenta para a extração de mensagens do WhatsApp

Lucas Ribeiro Ferreira and Wilson de Oliveira Mendonça

Maio/2021

Advisor: Filipe Braida do Carmo, D.Sc.

WhatsApp is the most used instant messaging application in Brazil. Its main use is private among users, but recent studies have shown the increasing importance of the platform as a source of information, including misinformation, and as a tool for organizing and agitating great social relevance events, such as strikes and protests. In this context, it has been demonstrated the importance of studying the messages exchanged within the platform and, through these studies, helping to understand the different determinations of social reality. Many works have performed the task of extracting messages from the platform using WebScrapping techniques or even decrypting the messages contained in a Smartphone database. In this way, the present work seeks to collaborate with this task by presenting a tool that can connect to the WhatsApp server and download all messages exchanged by an account, it has a simple and objective programming interface that allows its use for the implementation of different forms of user interfaces, being agnostic to any form. An application with a graphical user interface for home computers is also presented here, implemented using the API mentioned above, which is easy to use to help researchers and journalists to carry out their work more easily. The goal that is expected to be achieved with these tools is that more studies are done and, therefore, that they can better understand our society.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

WhatsApp[1] é um serviço de troca de mensagens instantâneas, sendo o aplicativo de dispositivos móveis de sua categoria mais instalado no mundo (SEVIT, 2018). A população que o utiliza com mais frequência no mundo é a população brasileira (NEWMAN et al., 2019). Cerca de 53% dos brasileiros utiliza o aplicativo de forma massiva para consumo de noticias, o que torna a torna mais vulnerável a propagação de desinformação (NEWMAN et al., 2019).

Marfianto e Riadi (2018) afirmam que muitas pessoas utilizam da rede para crimes digitais como fraudes, redes de drogas e pornografia. Machado et al. (2019) demonstraram que 13% das mensagens trocadas em grupos públicos de cunho politico na rede social durante o período da campanha eleitoral de 2018 no Brasil difundiam informações falsas. Das mensagens que possuem *link* para vídeos do *Youtube*[2], 31% foram consideradas desinformação.

Esses e outros diversos trabalhos acadêmicos e jornalísticos enfatizam a cada dia a importância da rede social no dia a dia do cidadão moderno. A rede social também tem sido usada para organização de protestos (TARDAGUILA, 2019; RESENDE et al., 2018), propaganda partidária (MACHADO et al., 2019) e diversos outros temas (GARIMELLA; TYSON, 2018).

[1] https://www.whatsapp.com
[2] www.youtube.com

Garimella e Tyson (2018) defendem a importância de que sejam feitos estudos usando as mensagens trocadas em grupos públicos dentro do WhatsApp afirmando que merecem a mesma atenção que outras redes sociais. Para que esses estudos possam ser realizados, é necessária a extração das mensagens de dentro da plataforma. Contudo, o WhatsApp, devido a suas politicas de privacidade, não provê uma *API* oficial para uso de pesquisadores, como outras redes sociais, *e.g. Twitter*. Na intenção de contornar essa questão, trabalhos acadêmicos foram feitos apresentando metodologias para a coleta dessas mensagens (GARIMELLA; TYSON, 2018; RESENDE et al., 2018). No entanto, para fazer uso dessas metodologias se faz necessário o conhecimento de programação.

Visando facilitar o trabalho de estudar as mensagens trocadas através do WhatsApp, o presente trabalho propõe duas ferramentas análogas, mas com distintas interfaces. A primeira consiste numa interface de programação que seja capaz de extrair todas as mensagens de uma conta de WhatsApp enquanto expõe uma interface simples, coesa e objetiva. Com essa API espera-se que outros trabalhos e projetos sejam realizados usando-a para criação de programas com diferentes tipos de interface, estudos usando os dados extraídos ou melhorando a própria API. A segunda ferramenta consiste numa aplicação com interface gráfica de usuário desenvolvida para os sistemas operacionais de computadores domésticos usando a API aqui proposta nesse trabalho. Tal ferramenta poderá ser utilizada pelo usuário sem que seja exigido dele qualquer conhecimento de programação. A expectativa é que, com tal ferramenta disponível, uma quantidade maior de estudos possam ser realizados.

O trabalho está divido em cinco capítulos, incluindo esta breve introdução do problema que se propõe a ajudar a resolver. No segundo capítulo são apresentados os conceitos de Mineração de Dados, junto a uma pequena introdução de Mineração de Texto e Mineração de Web, e *Web Scraping*. No capítulo seguinte é apresentado a motivação desse trabalho seguido dos trabalhos relacionados. Nesse mesmo capítulo ainda é apresentado com mais detalhes a proposta. No quarto capítulo estão descritas as tomadas de decisão feitas para a implementação do projeto, junto com uma breve descrição das tecnologias utilizadas, e uma apresentação da aplicação desenvolvida. O quinto e último capítulo conta as conclusões do trabalho e trabalhos futuros.

Capítulo 2

Fundamentação

Esse capitulo trata da mineração de dados descrevendo os principais objetivos desse campo de estudos, suas principais técnicas e aplicabilidades. Ele também descreve a técnica de *Web Scraping*.

2.1 Mineração dos Dados

Tradicionalmente, para extrair conhecimento de dados, é necessária uma análise e interpretação manual dos mesmos (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). Fayyad, Piatetsky-Shapiro e Smyth (1996) exemplificam dizendo que é comum que, na área de saúde, sejam gerados relatórios periódicos feitos por especialistas por meio da análise dos dados do plano de saúde e que, com base nesses relatórios, são feitos os planejamentos gerenciais e futuras tomadas de decisão.

Porém, com o volume de dados crescendo drasticamente, esse método de análise está se tornando impraticável para muitos domínios. Mesmo quando é possível realizar a análise, esse processo pode ser lento, caro e muito subjetivo (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). Empresas gastam milhões de reais para coletar e armazenar dados sem que informações uteis possam ser identificadas (CAMILO; SILVA, 2009). Foi com o intuito de solucionar esse problema que, no final da década de 80, foi proposta a Mineração de Dados (CAMILO; SILVA, 2009).

Alguns autores tratam a Descoberta de Conhecimento nas Bases de Dados (*Knowledge Discovery in Databases* - KDD) como sinônimo de Mineração de dados, enquanto outros tratam a Mineração de Dados como uma parte do processo de KDD (CAMILO; SILVA, 2009). Para Fayyad, Piatetsky-Shapiro e Smyth (1996, p. 40-41, tradução nossa), KDD é "um processo não trivial de identificação de novos padrões válidos, úteis e compreensíveis". Côrtes, Porcaro e Lifschitz (2002) definem Mineração de Dados como:

> (...) um processo altamente cooperativo entre homens e máquinas, que visa a exploração de grandes bancos de dados, com o objetivo de extrair conhecimentos através do reconhecimento de padrões e relacionamento entre variáveis, conhecimentos esses que possam ser obtidos por técnicas comprovadamente confiáveis e validados pela sua expressividade estatística (CÔRTES; PORCARO; LIFSCHITZ, 2002, p. 1-2).

O processo de mineração é iterativo, interativo e dividido em fases (CAMILO; SILVA, 2009). Fayyad, Piatetsky-Shapiro e Smyth (1996) separa o processo de *KDD* em seleção, pré-processamento, transformação, mineração de dados e interpretação/avaliação, ilustrado na figura 2.1.

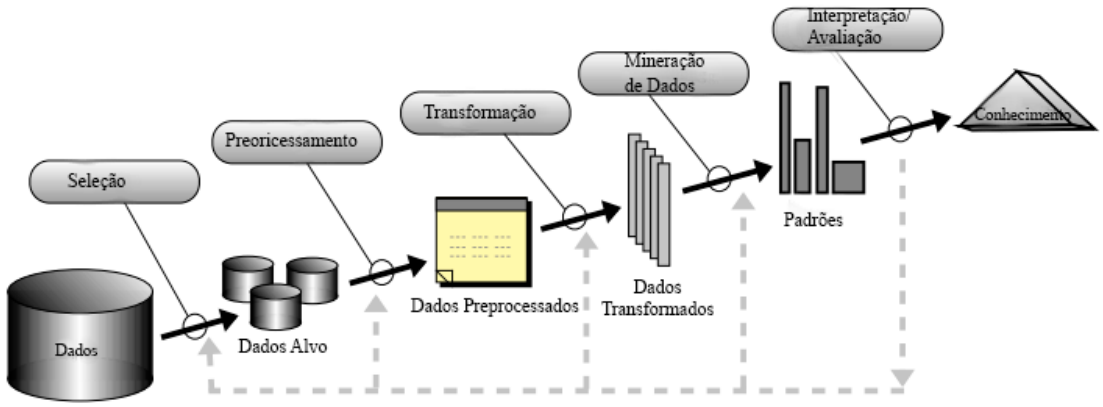

Figura 2.1: Uma visão geral dos passos que compõem um processo KDD. Adaptado de Fayyad, Piatetsky-Shapiro e Smyth (1996)

O processo CRISP-DM (*Cross-Industry Standard Processo of Data Mining*), que pode ser considerado o de maior aceitação hoje em dia (CAMILO; SILVA, 2009), é dividido em seis fases, apresentadas na figura 2.2: entendimento do negócio, entendimento dos dados, preparação dos dados, modelagem, avaliação e implantação (OLSON; DELEN, 2008).

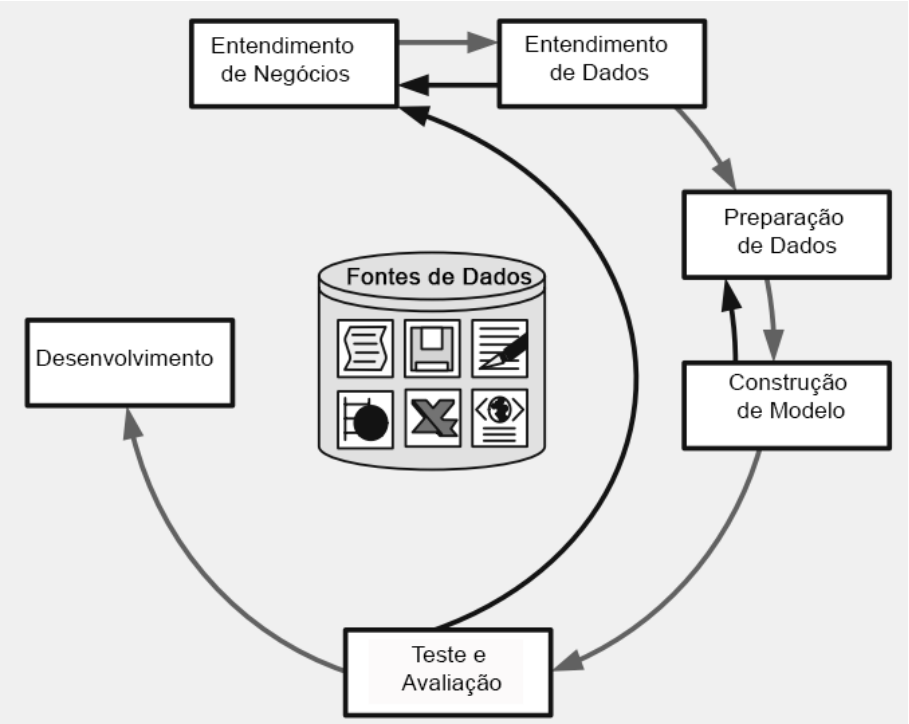

Figura 2.2: Processo CRISP-DM. Adaptado de Olson e Delen (2008)

Os dados podem ser qualitativos ou quantitativos. O primeiro são dados com valores nominais e categóricos. Já o segundo, diz respeito a dados com valores numérico, sejam eles discretos ou contínuos (CAMILO; SILVA, 2009). Para escolher o método a ser utilizado, é fundamental entender qual o tipo do dados será utilizado como entrada, e para utilizá-los, geralmente é preciso prepará-los (CAMILO; SILVA, 2009). Camilo e Silva (2009) ressaltam:

> Devido às diversas origens possíveis, é comum que os dados não estejam preparados para que os métodos de Mineração de Dados sejam aplicados diretamente. Dependendo da qualidade desses dados, algumas ações podem ser necessárias. Este processo de limpeza dos dados geralmente

> envolve filtrar, combinar e preencher valores vazios (CAMILO; SILVA, 2009, p. 4).

A fase de preparação dos dados pode ocupar, sozinha, a maior parte do processo (CAMILO; SILVA, 2009). Para melhor entender os dados de forma a auxiliar na escolha de como prepará-los, são utilizados diversas técnicas de visualização de dados (CAMILO; SILVA, 2009). Após obtido conhecimento sobre dados, é possível escolher a melhor forma de prepará-lo para a mineração. Na preparação se realiza a limpeza de dados, integração de dados, transformação dos dados e redução dos dados (HAN; PEI; KAMBER, 2011).

Muita vezes, quando se cria um modelo para um conjunto de dados de um problema, ele não se mostra satisfatório para um outro conjunto de dados do mesmo problema. Isso se dá pois o modelo se torna enviesado aos dados com os quais ele foi treinado. Esse efeito é conhecido como efeito *Bias*. Para evitar isso, normalmente se divide o conjunto em três partes: conjunto de treinamento, conjunto de testes e conjunto de validação (CAMILO; SILVA, 2009).

Camilo e Silva (2009) ressaltam que, apesar da existência massiva de dados sob posse das empresas, normalmente esses dados não são disponibilizados para pesquisadores. Devido a isso é comum se criar algoritmos teóricos e validá-los com dados sintéticos. Desta forma, não tendo a possibilidade de testá-los em um ambiente real.

A mineração de dados possui dois principais objetivos: predição e classificação. A predição consiste em usar parte dos dados ou dos campos para inferir valores de outros campos ou valores futuros dos mesmos. A classificação foca em encontrar padrões descritivos que humanos possam interpretar (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Existem diversas técnicas para auxiliar a alcançar os objetivos. Dentre elas estão: classificação, regressão, descrição, agrupamento, associação e predição (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). Cada uma dessas técnicas possuem diversos métodos e algoritmos, que podem ser supervisionados, não-supervisionados

ou uma variação entre os dois (CAMILO; SILVA, 2009).

Regressão Linear, por exemplo, é um método supervisionado de predição, que objetiva encontrar um possível valor futuro para uma variável. O método consiste em encontrar uma função linear a partir dos dados existentes onde o valor de y é estimado em função de uma variável x (CAMILO; SILVA, 2009).

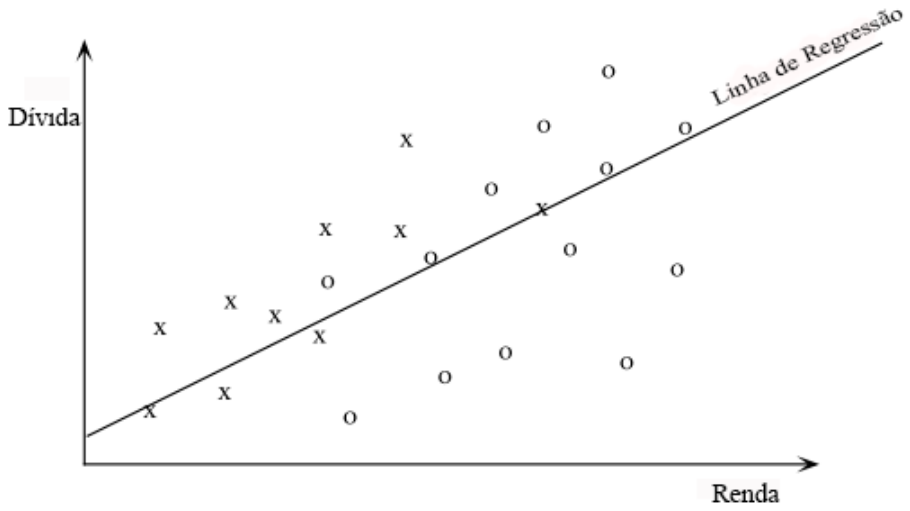

Figura 2.3: Uma regressão linear simples de um conjunto de dados de empréstimo. Adaptado de Fayyad, Piatetsky-Shapiro e Smyth (1996)

Uma das limitações da mineração de dados está no formato do dado de entrada. Inicialmente pensadas para serem aplicadas em dados estruturados em bancos de dados ou arquivos, as técnicas de mineração podem não ter mesma eficácia ao lidar com dados em outros formatos, *e.g.* multimídia, texto, espacial. Devido a isso muitos estudos tratando a mineração de dados complexos vem sendo feitos (CAMILO; SILVA, 2009). Um exemplo disso é a mineração de textos.

Devido à falta de estrutura, dados textuais normalmente são manipulados por meio de motores de busca, diferente de dados estruturados, que normalmente são manipulados por meio de sistemas de banco de dados. Motores de busca são eficientes no objetivo de levar o usuário à informação correta no tempo correto, mas a mineração de texto é mais que isso. Mineração de texto pode ajudar o usuário a digerir a informação e ajudar na tomada de decisão (AGGARWAL; ZHAI, 2012). Aggarwal e Zhai (2012) completam:

> Existem diversas aplicações para a mineração de texto onde o objetivo primário é analisar e descobrir algum padrão interessante, incluindo tendencias e *outliers*, em dados textuais, e a noção de busca não é essencial ou mesmo relevante (AGGARWAL; ZHAI, 2012, p. 2, tradução nossa).

Marfianto e Riadi (2018) apresentam o que pode ser usado com um exemplo de uso de mineração de texto. O trabalho utiliza de técnicas de mineração de texto para fazer uma análise forense das mensagens do WhatsApp de um suspeito com o intuito de ajudar a extrair evidências criminalísticas.

Uma das fontes de texto mais comuns são as redes sociais. Elas permitem que as pessoas se expressem de forma rápida e livre em contextos muito abrangentes. Para minerar texto dessas fontes é necessário técnicas específicas, pois podem conter um vocabulário coloquial e sem padronização. Métodos que fazem uso tanto das conexões da rede social quanto do conteúdo, por exemplo, conseguem prover resultados mais eficientes que os que usam apenas o conteúdo ou conexões (AGGARWAL; ZHAI, 2012).

As formas mais comuns de mineração de conteúdo da internet é feito através de mineração de dados não-estruturados, mineração de dados estruturados, mineração de dados semi-estruturados ou mineração de dados multimídia. A coleta de dados para mineração é uma tarefa importante para a mineração de dados não-estruturados e um dos maiores desafios, devido à complexidade de todas as *tags HTML* que podem estar presentes. *Web Scrapers* ajudam a simplificar essa tarefa (DASTIDAR; BANERJEE; SENGUPTA, 2016).

2.2 *Web Scraping*

A internet se tornou a maior fonte de dados do mundo (BANERJEE, 2014). A geração de dados e o crescimento de sua taxa está aumentando a cada dia (DASTIDAR; BANERJEE; SENGUPTA, 2016). A internet foi projetada para que seja fácil para que pessoas encontrem informações (BANERJEE, 2014). Usuários na internet podem

usufruir abundantemente de serviços e informação, *e.g.* comércio eletrônico, *websites*, jornais eletrônicos, redes sociais e blog (DASTIDAR; BANERJEE; SENGUPTA, 2016). Mas, se a única forma de acessar esses dados for por meio de um navegador *web*, uma grande variedade de possibilidades será perdida (MITCHELL, 2018).

Muitas empresas, com o objetivo de localizar, capturar e guardar um enorme volume de informação que precisam de *websites*, ainda usam de métodos tradicionais de extração de dados da internet, como o manual "copia e cola"(BANERJEE, 2014). O processo de extração manual consiste na empresa contratar uma grande quantidade de funcionários e orientá-los a fazer a extração navegando pelas páginas copiando os dados para um banco de dados (BANERJEE, 2014). Tal processo, além de muito caro, também é muito suscetível a erros humanos (BANERJEE, 2014).

Outra possibilidade para recuperar essas informações seria por meio das APIs (*Application Program Interface*). Uma *API* define uma sintaxe padronizada que permite uma parte de um software se comunicar com uma outra parte, mesmo que eles tenham sido escritos em linguagens diferentes ou estruturados de forma diferente (MITCHELL, 2018). É possível encontrar *APIs* de diversos tipos de dados que podem ser utilizados, *e.g.* postagens do *Twitter*[1], paginas do *Wikipedia*[2]. Apesar de ser preferível fazer uso das APIs, elas podem não existir para o dado desejado ou possuir limitações de acesso, *e.g* quantidade de acessos por dia (MITCHELL, 2018).

Páginas HTML é a principal ferramenta de formação de informação na internet. Dastidar, Banerjee e Sengupta (2016, p. 25) define *Web Scraping* como o processo de extrair informações úteis dessas paginas usando qualquer linguagem de programação. Para Mitchell (2018), é na prática uma vasta variedade de técnicas de programação e tecnologias, como análise de dados, análise de linguagem natural e segurança da informação. Mitchell (2018) ainda define teoricamente como:

> (...) a prática de recuperar dados por meio de qualquer outro meio que não um programa interagindo com uma API (ou, obviamente, por meio de um humano usando um navegador). Normalmente, isso é alcançado

[1] www.twitter.com
[2] www.wikipedia.com

> escrevendo um programa automatizado que consulta um servidor *web*, requisita dados (normalmente em forma de HTML e outros arquivos que compõem uma página *web*), e então analisa esses dados pra extrair a informação necessária (MITCHELL, 2018, p. ix, tradução nossa).

Uma das aplicabilidades da técnica de *Web Scraping* é na mineração de dados na internet (DASTIDAR; BANERJEE; SENGUPTA, 2016). Mineração de internet, como explica Shi, Ma e He (2009, p. 197-198, tradução nossa), "é uma área de pesquisa que tenta identificar pedaços de informação aplicando técnicas de mineração de dados e aprendizado de máquina a documentos e dados da internet". As principais formas de mineração de conteúdo na internet é feito por meio de mineração de dados não-estruturados, mineração de dados estruturados, mineração de dados semi-estruturados e mineração de dados multimídia (DASTIDAR; BANERJEE; SENGUPTA, 2016).

Devido à complexidade das *tags HTML*, recuperar informações de documentos na internet pode se tornar uma tarefa complicada, mesmo sendo uma parte importante da mineração de dados não-estruturados. *Web Scraping* simplifica significativamente essa tarefa (DASTIDAR; BANERJEE; SENGUPTA, 2016).

Um exemplo de uso de *Web Scrapping* pode ser encontrado no trabalho de Boeing e Waddell (2017). Os autores extraíram a listagem de aluguéis de casas nos Estados Unidos do site *Craiglist*[3] e conseguiram demonstrar novas informações sobre padrões de distribuição espacial do mercado de casas no país de forma mais rica que utilizando outras fontes de informações públicas, como o censo.

[3]http://craigslist.org/

Capítulo 3

Coletor de Dados para WhatsApp

Esse capitulo apresenta as motivações do presente trabalho, seguido pelos trabalhos relacionados e a descrição da aplicação proposta.

3.1 Motivação

Com a facilidade de trocar mensagens e criar grupos de forma gratuita por meio de aplicativos mensageiros, criou-se um novo meio de formação de comunidades. Essas comunidades normalmente giram em torno de um tema, que são diversos, *e.g.* família, política, esportes, animes, ativismo, negócios, trocas e educação, grupos que podem ser públicos ou privados (GARIMELLA; TYSON, 2018).

O Whatsapp é o mensageiro mais instalado no mundo (SEVIT, 2018) e a população brasileira é uma das mais assíduas nessa rede social (NEWMAN et al., 2019). Cerca de 84% dos brasileiros são usuários do aplicativo, que é a rede social mais popular no país, 53% consomem e compartilham notícias por meio da rede (NEWMAN et al., 2019). Newman et al. (2019) também chamam atenção que no Brasil os usuários de Whatsapp fazem parte de grupos com pessoas que não conhecem com mais frequência que nos países ocidentais, associando isso ao uso da ferramenta como uma fonte de notícias, informação e facilitando a propagação de desinformação.

Há alguns anos esses grupos de Whatsapp vem desempenhando um papel impor-

tante na sociedade como uma ferramenta de mobilização de massas. Em 2018, a rede social foi primordial para a organização e agitação de uma das maiores greves de caminhoneiros do Brasil. Cerca de 45% dos caminhoneiros tiveram conhecimento da manifestação por meio do aplicativo (DEOTTI, 2018). Esse mesmo evento pôde ser monitorado pelo sistema de monitoramento de grupos públicos de WhatsApp apresentado por Resende et al. (2018), o que reforça a importância da rede na mobilização, como reportado por Rossi (2018).

De semelhante modo, os grupos de WhatsApp estão sendo utilizados para a disseminação de desinformação, com notícias falsas e discursos de ódio. Durante a campanha presidencial de 2018 no Brasil, Machado et al. (2019) coletaram mensagens grupos de cunho político no aplicativo e mostraram que 13% das mensagens difundiam informação falsa. Das mensagens que levavam a vídeos no *Youtube*[1], que representam 40% dos links compartilhados, 31% foram consideradas desinformação e das que levavam ao *Facebook*[2] 42%. Dentre todos os vídeos compartilhados pela rede $9,5\%$ correspondiam a discurso de ódio, violência extrema explícita ou pornografia atacando minorias. O estudo ainda aponta que mensagens desse último tipo frequentemente são utilizadas como estratégia para que se alcance uma disseminação viral.

Reforçando a importância da rede social na disputa eleitoral, Machado et al. (2019) demonstraram que 65% dos vídeos compartilhados nos grupos foram classificados como favoráveis ao candidato vencedor das eleições enquanto apenas 5% correspondiam a defesa do seu principal oponente na disputa. Dentre as imagens compartilhadas, a proporção foi de $58,5\%$ e $15,5\%$ respectivamente.

Estudar as interações em grupos do WhatsApp vem se mostrando uma forma eficiente de entender os acontecimentos sociais. Garimella e Tyson (2018) reforçam a ideia dizendo que estudar o WhatsApp deve ter a mesma relevância que estudar outras redes sociais. Porém, coletar as mensagens trocadas na plataforma não é uma tarefa simples devido a falta de uma ferramenta específica, como é possível encontrar para outras redes sociais populares como *Facebook* ou *Twitter*[3].

[1] http://www.youtube.com.br
[2] http://www.facebook.com
[3] http://www.twitter.com

Garimella e Tyson (2018) coletaram as mensagens acessando o banco de dados do dispositivo móvel e descriptografando-as, o que não seria possível ser feito sem conhecimentos de programação específicos. O WhatsApp possui uma plataforma Web onde os usuários podem trocar as mensagens usando o navegador. Tal plataforma foi utilizada por Resende et al. (2018) para contornar a criptografia ponta a ponta programando um *crawler* que fica coletando as mensagens presente no HTML da página. Apesar do trabalho disponibilizar seus códigos de forma pública na internet, ainda se faz necessário ter conhecimentos de programação para poder utilizar das mesmas metodologias.

Hoje, para que pesquisadores ou jornalistas possam estudar de forma quantitativa as interações em grupos públicos de WhatsApp, é vital o conhecimento de programação. Tornar a coleta de mensagens de Whatsapp uma tarefa menos técnica, pode facilitar que profissionais de outras áreas façam mais estudos.

3.2 Trabalhos Relacionados

Devido ao seu protagonismo em recentes eventos pelo mundo, a quantidade de trabalhos na literatura que exploram o uso do Whatsapp tem aumentado, alguns deles propondo metodologias para extração de dados junto a uma análise (RESENDE et al., 2018; GARIMELLA; TYSON, 2018). Por outro lado, outros trabalhos focam em classificar dados quantitativos de grupos públicos e analisar o comportamento desses, sem entrar em detalhes mais técnicos sobre a fase de coleta (MACHADO et al., 2019; CAETANO et al., 2018). Todos eles ressaltam a necessidade de ser fazer ainda mais estudos envolvendo o aplicativo, tanto para melhor entender como se dá o comportamento das pessoas na plataforma, como para compreender como ela vem sendo utilizada por atores da sociedade.

Resende et al. (2018) apresentam uma metodologia de extração de dados que se resume em três etapas: coleta de *links* para grupos de Whatsapp de interesse da pesquisa, inscrição nesses grupos e coleta de mensagens e demais informações. Para a última etapa foi utilizado um *crawler web* que extrai as informações disponíveis

na versão *web* do Whatsapp, livre de criptografias. Também foi utilizado um *script* para automatizar a primeira etapa. Tal trabalho confirmou seu valor ao ser utilizado em matérias jornalisticas como as publicadas na BBC (ROSSI, 2018), G1, Folha de São Paulo e Época (TARDAGUILA, 2019).

> Além de descrever nossa metodologia, também fornecemos uma breve caracterização do conteúdo compartilhado por 6.314 usuários em 127 grupos públicos brasileiros do WhatsApp com temáticas relacionadas à política e notícias gerais. Nós acreditamos que nosso sistema possa ajudar jornalistas e pesquisadores a entender a repercussão de eventos relacionados às eleições brasileiras dentro desse espaço midiático. (RESENDE et al., 2018, p. 387)

De forma semelhante a Resende et al. (2018), Garimella e Tyson (2018) também utilizaram de um *crawler* para registrar contas de Whatsapp em uma lista de grupos obtidos anteriormente. Mas, para coletar as mensagens, Garimella e Tyson (2018) extraíram e descriptografaram o banco de dados acessando fisicamente o dispositivo móvel que está registrado na conta. O objetivo deste trabalho está em prover contexto aos grupos públicos de WhatsApp para que pesquisadores possam compreender quais dados podem ser coletados e como podem ser utilizados. O trabalho conclui que:

> (...) como um popular meio de comunicação em muitas partes do mundo, nós argumentamos que o WhatsApp deve receber uma atenção equivalente a outros serviços de mídia social, *e.g.* Twitter. Nós esperamos que esse trabalho, e as ferramentas associadas a ele, possam servir como uma plataforma para outras pesquisas construírem sobre. (GARIMELLA; TYSON, 2018, p. 387, tradução nossa)

Em Machado et al. (2019) encontra-se um importante trabalho de análise sobre dados coletados do Whatsapp. Além de achados que reforçam o tom da importância da rede social nas eleições brasileiras de 2018, o estudo apresentou um método para classificação do conteúdo coletado. A classificação foi feita de acordo com a origem da

informação: conteúdo profissional de notícia, profissional de política e de polarização e conspiração; e de acordo com a afinidade política. Outras conclusões do trabalho:

> (1) No Brasil, WhatsApp apresenta um nímero extremamente pequeno de conteúdo político profissional e um alto número de conteúdo enganoso; (2) A propagação da informação no WhatsApp depende de uma disseminação intensa de arquivos de mídia, que não usa a mesma retórica que fontes de notícias enganosas, não tentam simular uma autoridade para creditar a informação; (3) Estratégias de disseminação de conteúdo em grupos de WhatsApp frequentemente recorre a discurso de ódio e engano para alcançar uma disseminação viral. Nossa investigação indica que metáforas visuais estão sendo pesadamente utilizadas dentro dos grupos de WhatsApp para distorcer informação e manipular os usuários (MACHADO et al., 2019, p. 1017, tradução nossa).

Usando uma estratégia de coleta de dados semelhante à proposta por Resende et al. (2018), o estudo de Caetano et al. (2018) se caracteriza pelo o que afirma acreditar ser, na literatura científica, a primeira análise significante do comportamento de grupos no WhatsApp. O trabalho introduz um *framework* e métricas para caracterizar o comportamento de grupos de comunicação em aplicativos de troca de mensagem móveis como o WhatsApp. Diferente de outros trabalho, esse não se preocupa com o conteúdo em si, mas sim com dados como a frequência de mensagens, nível de atividade de usuários em grupos, proporção de mensagens trocadas em forma de arquivo de mídia, emoji e textual, dentre outros. O estudo exemplifica a utilidade das métricas as utilizando para comparar grupos políticos a não políticos.

> Vale a pena mencionar que, apesar do cenário desse artigo ser contrastar grupos públicos políticos e não políticos de WhatsApp, nós acreditamos que nossa metodologia pode ser aplicável em diversos cenários e também em outras plataformas de mensagem instantânea. (...)
>
> Nós esperamos que os achados desse artigo possam contribuir para clarear a forma que o WhatsApp funciona e reduzir a opacidade de serviços moder-

> nos da infraestrutura global de comunicação de informação. (CAETANO et al., 2018, p. 1013, tradução nossa).

Todos os trabalhos citados aqui propõem metodologias de extração ou análise de dados onde, apesar de muito úteis tanto para pesquisadores quanto para jornalistas, há a necessidade de conhecimentos de programação para a realização da etapa de coleta. Seja para a implementação de um *script* ou para o uso daqueles disponibilizados publicamente. O presente trabalho, diferente dos anteriores, introduz uma ferramenta de fácil instalação e uso por não programadores para a extração de mensagens e outras informações de grupos públicos de WhatsApp na tentativa de promover ainda mais dinamismo e incentivar o aumento da quantidade de pesquisas sobre a plataforma.

3.3 Proposta

Sua natureza privada e pessoal manteve o WhatsApp fora do foco de acadêmicos por muito tempo, com trabalhos se limitando a estudos qualitativos com uso de voluntários (GARIMELLA; TYSON, 2018). Para manter a segurança e a privacidade de seus usuários, o aplicativo mantém suas mensagens criptografadas de ponta a ponta. Além disso, o WhatsApp não disponibiliza nenhuma forma oficial para a coleta de informações para pesquisadores, diferente do Facebook e Twitter.

Tendo em vista cenário apresentado, estudar o aplicativo vem se mostrando cada vez mais essencial para que se possa compreender a sociedade e os eventos que nela ocorrem. Mesmo sendo uma tarefa especialmente complexa dada a sua natureza, a quantidade de estudos sobre o WhatsApp vem crescendo e reforçam a importância de que mais trabalhos sejam feitos nessa direção. Para que esses trabalhos pudessem ser feitos, foi preciso contornar as limitações que a plataforma impõe utilizando técnicas de programação. Tal requisito pode ser um impeditivo para que outros pesquisadores façam seus trabalhos.

Com o objetivo de facilitar pesquisas sobre as interações interpessoais dentro dos grupos públicos de WhatsApp, o presente trabalho se propõe a disponibilizar uma ferramenta de fácil uso e livre do requisito de conhecimento programação para

extração de mensagens e outras informações relevantes desses grupos. Para isso, tal ferramenta deverá contar com mecanismos de interface gráfica e poder ser usada em um computador pessoal.

O trabalho também se propõe a implementar a aplicação de forma que sua API interna seja desacoplada de seu código de interface. Sendo assim, também será disponibilizado a API em um repositório separado para que possa ser implementada as mais diversas e úteis formas de interface que se faça necessário. Tais interfaces, *e.g* linha de comando, Rest, *Socket*, poderão ser implementadas posteriormente por qualquer um que desejar. Dessa forma, as pessoas interessadas poderão concentrar seus esforços em implementar a interface de usuário desejada e também poderão contribuir para a melhoria da API caso haja interesse. Tal como a ferramenta de interface gráfica, a API será concentrada na extração de mensagens trocadas por uma conta de WhatsApp.

No método apresentado por Resende et al. (2018), foi utilizado a ferramenta *WebWhatsAppAPI*[4] para que pudesse ser feito a coleta das mensagens contornando a criptografia. Tal ferramenta, apesar de eficiente, necessita de muitas etapas de configuração de ambiente e sua manipulação é inteiramente feita via linha de comando ou via codificação de *scripts*. Garimella e Tyson (2018), por sua vez, apresenta uma metodologia em que em uma etapa as mensagens são coletadas usando um *smartphone*, com o WhatsApp instalado, deixando este com uma conta autenticada e devidamente inscrita nos grupos em que há interesse em coletar as mensagens. Em uma etapa posterior é usado um programa que é capaz de acessar o banco de dados do WhatsApp e fazer a descriptografia do mesmo, conectando esse *smartphone* a um computador. Em face das constantes atualizações de segurança do WhatsApp, a realização dessa segunda etapa pode se tornar um impeditivo, além de também necessitar que usuário possua um conjunto de habilidades ligados à tecnologia da informação. Necessidade essa que o presente trabalho se propõe a contornar.

A metodologia para a coleta e extração das mensagens proposta no presente trabalho, no nível mais alto, se assemelha ao trabalho de Garimella e Tyson (2018)

[4]https://github.com/mukulhase/WebWhatsapp-Wrapper

pois também faz uso da estratégia de duas etapas. Na primeira etapa, de coleta das mensagens, o usuário deverá configurar um dispositivo móvel com uma conta de WhatsApp e subscrever em todos os grupos que deseja coletar mensagens. Uma vez que o aplicativo estiver devidamente configurado, ele começará a receber as mensagens e guardá-las localmente em um banco de dados. Na segunda etapa, de extração das mensagens coletadas, que é onde o presente trabalho se concentra, deverá ser usada a ferramenta aqui proposta, que será capaz de extrair todas as mensagens coletadas na primeira etapa e seu uso se dará através de uma interface gráfica de usuário em um computador pessoal. Nos próximos parágrafos será descrito com mais detalhes as características, funcionalidades desse *software*.

O WppScrapper GUI, ferramenta que será implementada e apresentada no próximo capítulo, tem como seu principal objetivo prover ao usuário uma interface através da qual ele será capaz de extrair todas as mensagens trocadas por uma conta de WhatsApp que ele possua. O usuário aqui idealizado é aquele que possui o interesse em extrair essas informações do WhatsApp mas não possui conhecimentos de programação requeridos para fazê-lo usando interfaces de programação ou de linha de comando disponíveis. Tendo em vista esse usuário, deve-se projetar uma ferramenta mais simples e objetiva possível e onde todo seu uso seja através da interface gráfica. Esse software deverá poder ser usado em um computador pessoal qualquer.

Acreditando ser um impeditivo menor, a ferramenta a ser implementada não visa automatizar a etapa de inscrição automática nos grupos públicos de WhatsApp, nem mesmo a coleta dos links dos mesmos. Tais etapas poderão continuar sendo realizadas das formas propostas por Resende et al. (2018) ou Garimella e Tyson (2018), tal como qualquer outro método, sem causar impacto no funcionamento da ferramenta.

O usuário do *software* aqui proposto deverá poder utilizá-lo para extrair todas as mensagens de WhatsApp enviadas e recebidas por uma determinada conta que, no aplicativo, é identificada por um número de celular. As mensagens extraídas deverão estar devidamente ordenada pelo sistema e a elas deve ser atribuído informação

temporal do momento de envio ou recebimento da mesma. Além disso, também deverá para cada mensagem, conter atributos que identifique quem enviou, um valor identificador da mensagem em si, o nome ou identificador da conversa ou grupo onde a mensagem foi enviada e, caso seja uma resposta a outra mensagem, o identificador dessa. O procedimento de coleta de mensagens também deverá coletar informações sobre os grupos onde as mensagens foram trocadas, *e.g.* nome do grupo, descrição e identificador de todos os membros, caso tenham sido trocadas em um grupo.

Todas essas informações deverão ficar armazenadas em local de fácil acesso pelo usuário e em um formato que possa ser facilmente lido e manipulado por outras ferramentas. Esses dados deverão poder ser usados em manipulação direta do usuário usando editores de texto ou ferramentas de planilha, *e.g. LibreOffice Calc*, como dados de entrada para programas de análise de dados, aprendizado de máquina ou outros programas escritos exclusivamente para a finalidade desejada, dentre outras vastas possibilidade de utilidades. Para atender esses requisitos, propõe-se que todas as informações estejam em formato CSV (*Comma-separated values*[5]). Cada conversa terá seu próprio arquivo de mensagens e, caso seja um grupo, também haverá outros dois arquivos adicionais, um com a lista de membros e outro com as demais informações.

Devido à técnica utilizada para fazer a extração das mensagens, a ferramenta proposta no presente trabalho precisará de uma etapa na qual a conta que o usuário está usando para fazer a coleta das mensagens deverá ser autenticada junto ao servidor do WhatsApp. Para cumprir com esse requisito, a ferramenta deverá, sempre que iniciada, verificar se uma sessão prévia existe e se é possível recuperar a conexão usando-a ou se é necessário que o usuário autentique a conta de WhatsApp usada para a coleta das mensagens. Caso a autenticação se faça necessária, a ferramenta deverá recuperar junto ao servidor do WhatsApp o *QRCode* e apresentá-lo ao usuário através da interface gráfica. O usuário, por sua vez, deverá usar o aplicativo do WhatsApp em seu *smartphone*, com a conta devidamente autenticada, para escanear o *QRCode* apresentado pela ferramenta. A ferramenta estará pronta para receber a informação de que a conta foi devidamente autenticada pelo servidor do WhatsApp

[5]https://tools.ietf.org/html/rfc4180

e deverá salvar essa sessão para que possa ser usada novamente no futuro, pulando a etapa de autenticação. Após um usuário estar com a conta autenticada junto à ferramenta, com a sessão ativa, a ferramenta deverá apresentar para ele, através da interface gráfica, a possibilidade de finalizar encerrar a sessão. Com a sessão encerrada o usuário poderá autenticar uma nova conta seguindo os mesmos passos descritos anteriormente neste parágrafo.

O usuário, uma vez autenticado, terá na interface a opção de iniciar a extração das mensagens. O sistema deverá responder iniciando a extração e informando ao usuário que a extração foi iniciada. O sistema também deverá desabilitar a opção para iniciar a coleta e habilitar a opção de pausar ou cancelar a extração. O usuário poderá esperar até que a extração termine. O sistema, depois de todas as mensagens serem extraídas, deverá expor essa informação ao usuário e uma opção para o usuário confirmar. Após o usuário confirmar, o sistema deverá voltar a apresentar a opção por iniciar a extração das mensagens e desabilitar as opções de pausar e cancelar. Uma vez que já possuem dados extraídos no caminho de destino, caso a extração seja novamente iniciada, o sistema deverá extrair novamente os mesmos dados e, uma vez extraídos, substituir os já existentes.

Caso o usuário, enquanto o sistema estiver realizando a extração das mensagens, opte por pausar a extração, o sistema deverá responder interrompendo o processo e apresentando ao usuário a possibilidade do mesmo reiniciar a extração. Os dados já extraídos no momento da interrupção poderão ser acessados pelo usuário, mas sem garantias de integridade. O sistema, depois do usuário optar por reiniciar a operação, deverá continuar a extração sem que seja perdido nenhuma mensagens já coletada, mas não é necessário ter garantias de que a extração recomece exatamente de onde parou, podendo recomeçar em outra conversa ou grupo.

No caso do usuário fechar a aplicação durante a execução ou algum erro inesperado ocorra que cause a interrupção abrupta, o sistema, ao ser reiniciado, deverá apresentar a opção de reiniciar a extração tal como se o usuário tivesse optado por pausar e também deverá apresentar a opção por iniciar novamente a extração.

No intuito de implementar o sistema descrito nos parágrafos anteriores seguindo

as melhores práticas de programação, será implementada também uma API que possua as principais lógicas de domínio descritas. Essa API será também apresentada como um resultado do presente trabalho. Ela será disponibilizada em um repositório separado. Fazendo uso de tal código, os autores do presente trabalho ou outros programadores poderão implementar outros tipos de interface que possua diferentes benefícios. Poderá ser implementada, por exemplo, uma interface de linha de comando de tal sorte que essa poderá ser instalada em um servidor remoto, com mais poder computacional ou espaço. Será possível também a implementação de um sistema web o qual, instalado em um servidor remoto, poderá prover uma interface gráfica através do navegador. Essas são apenas algumas das formas adicionais de interface que poderão ser implementadas por qualquer programador que se interesse.

Ao fazer uso dessa API, o programador terá o benefício de não precisar reimplementar diversas lógicas do domínio da aplicação principal e poder se concentrar no código de interface. Dentre essas lógicas estarão a ordenação das mensagens, os tratamentos necessários para pausar uma extração e continuar novamente no mesmo ponto, os tratamentos necessários para recuperar um QRCode de autenticação e a realização da recuperação de uma sessão salva. A formatação do arquivo CSV, sua criação, destruição, adição de novas mensagens, cabeçalho, dentre outras manipulações necessárias ao arquivo também deverão se encontrar encapsuladas pela API. Além das funções mais importantes, que serão as que inicia a operação de extração e a que finaliza, a API também deverá dispor de métodos para que o código que a consome possa receber informações como a lista de todos os chats e qual estado (extraindo, extraído ou esperando) atual do mesmo. Também poderá receber a informação de que a extração terminou tal como retornos de erro.

Capítulo 4

WppScrapper

Com a intenção cumprir com os objetivos descritos na seção 3.3, o presente trabalho propõe a criação de uma aplicação onde o usuário poderá configurar uma conta de WhatsApp e realizar a coleta de mensagens. Tal aplicação poderá ser instalada em um computador doméstico simples para ser utilizada pelo usuário e terá uma interface visual gráfica. A essa ferramenta dá-se o nome de *WppScrapperGUI*.

Como parte da proposta também consta a criação de uma API capaz das mesmas tarefas, mas que deve ser implementada de forma agnóstica a como será apresentada para o usuário ou a sua interface de usuário. A essa API dá-se o nome de *WppScrapper*.

No presente capítulo é possível encontrar descrição das decisões técnicas por trás da implementação do *WppScrapper*, tal como sua interface de programação e a utilização da mesma na implementação da *WppScrapperGUI*. Também está presente aqui exemplos de uso da *WppScrapperGUI*.

4.1 Implementação da WppScrapper

O primeiro e principal problema a ser revolvido para que a proposta desse trabalho possa ser entregue está na técnica pra extrair as mensagens trocadas através do WhatsApp. Como já mostrado na seção 3.2, os trabalhos relacionados procuram resolver esse problema de duas formas. Uma delas é extraindo as mensagens

diretamente do dispositivo móvel, acessando o banco de dados e descriptografando-o, e a outra consiste em extraí-las da página do *WhatsApp Web* utilizando de técnicas de *Web Scrapping*.

Para a implementação deste trabalho ficou decido utilizar uma API de terceiro que provê uma reimplementação da API do WhatsApp Web. Essa API, a *GOWhatsApp*[1], consiste na implementação da engenharia reversa da API do WhatsApp Web de forma que tal implementação é capaz de se comunicar diretamente com o servidor do WhatsApp Web e prover uma interface de programação igual, ou próxima, à interface *WebSocket* usada internamente pelo WhatsApp Web.

O uso da *GOWhatsApp* é justificado pela facilidade que a mesma provê para que se possa obter os dados que a aplicação aqui proposta necessita. Mas, apesar de mais fácil, ainda se fez necessário encapsular essa API e prover uma outra que seja mais direta ao suprir os objetivos do trabalho atual, como extrair as mensagens para um arquivo *CSV*. Além disso, esse encapsulamento também se justifica pela necessidade de que a API *WppScrapper* não dependa demais dessa API de terceiro e que a forma de obtenção dos dados possa ser trocada sem que a interface mude. Ou seja, os programas que utilizarem a *WppScrapper* não precisarão mudar sua implementação caso o *WppScrapper* mude a forma de acessar os dados internamente; assim, o *GO WhatsApp* poderá ser substituído futuramente sem impactar a interface gráfica, que será apresentada mais adiante, e nem nenhum outro programa que possa surgir futuramente.

A API *GO WhatsApp* é escrita usando a linguagem de programação *GO*[2]. Esse fato não torna mandatório que o resto do projeto aqui desenvolvido seja feito usando essa mesma linguagem, pois existem ferramentas da linguagem que possibilitam que suas funções sejam executadas de outras linguagens. Mas ficou decidido implementar a API WppScrapper também nesta linguagem devido às facilidades e suas características.

O presente trabalho objetiva implementar também um programa com interface gráfica de usuário que utiliza a API que está sendo descrita no presente capítulo e

[1]https://github.com/Rhymen/go-whatsapp
[2]https://golang.org/

apresenta ao usuário suas funções. A linguagem GO permitirá que esse programa possa ser executado em qualquer sistema operacional com facilidade, pois ao realizar a exportação do executável, o compilador da linguagem embute todas as dependências e tem como resultado um único arquivo. Esse compilador é construído para conseguir exportar para todos os principais sistemas operacionais modernos e seus programas serem executados sem a necessidade de instalação ou configuração de nenhuma dependência adicional. Esse objetivo é herdado da linguagem para as demais ferramentas e APIs construídas usando-a. Dessa forma, mesmo que haja a necessidade do uso de um *framework* ou uma biblioteca específica, dificilmente a mesma limitará as plataformas que será possível executar o programa final ou adicionará necessidades que dificultem sua instalação e uso.

Além disso, também foi avaliado o fato de ser uma linguagem de código aberto, ter uma comunidade ativa dando suporte não só na manutenção e desenvolvimento da linguagem em si, mas também a uma sorte enorme de APIs e ferramentas feitas com e para a linguagem, uma documentação bem escrita e uma sintaxe limpa e familiar.

Como visto anteriormente, a interface da *WppScrapper* precisará ser sólida e agnóstica a detalhes de implementação interna. No diagrama da figura 4.1 está ilustrado o que está aqui idealizado: o programa *WppScrapperGUI* precisará referenciar a implementação concreta do *WppScrapper* apenas no momento de criar a instância, diminuindo o impacto de mudanças internas. Tal como o *WppScrapperGUI* o mesmo deve ser verdade para qualquer programa que use a API *WppScrapper* para extrair mensagens trocadas por meio do WhatsApp.

A figura 4.2 descreve um diagrama de classes com a definição das interfaces de programação resultado de todos os requisitos descritos na seção 3.3. O usuário da API poderá instanciar uma implementação concreta da interface *IWppScrapper* e através dela ter acesso às funcionalidades desejadas. Esse usuário também poderá implementar as interfaces que possuem sufixo *Listener* e, usando a instância de *IWppScrapperEventHandler* que é provida pela implementação de *IWppScrapper* através da função *GetAppScrapperEventHandler*, adicionar e remover *Listeners* para

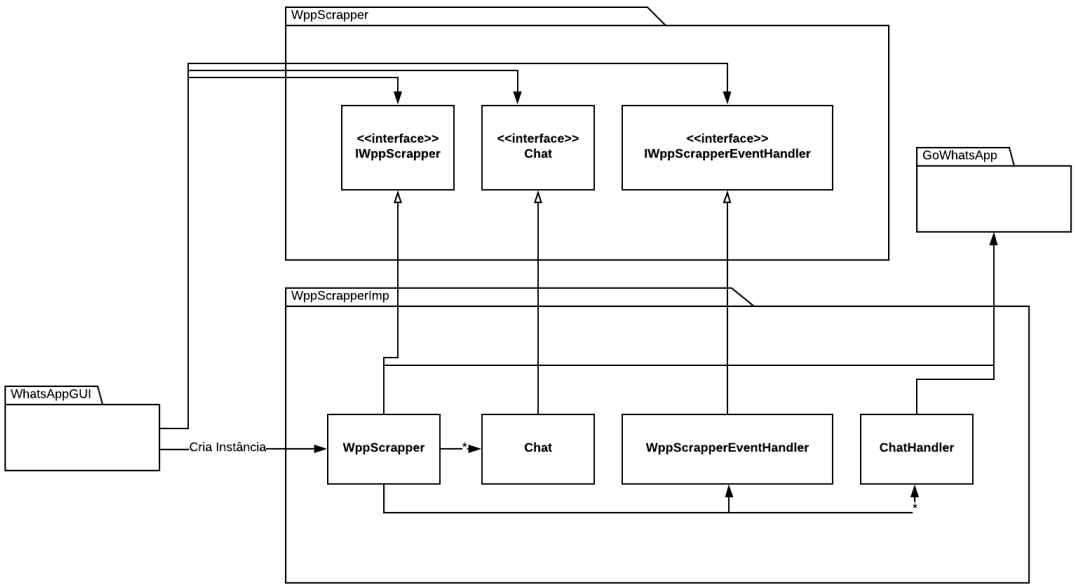

Figura 4.1: Diagrama de pacotes simplificado com diagrama de classes simplificado que demonstra como o *WppScrapper* se relaciona com o *WppScrapperImp*, *WppScrapperGUI* e com a API *GoWhatsApp*.

receber os eventos desejados e poder responder a eles.

É de responsabilidade da implementação concreta implementar o *IWppScrapperEventHandler* e disparar os eventos no momento correto. A implementação do *WppScrapper* também possui a responsabilidade de implementar todas as outras funções de *IWppScrapper* tal como uma função adicional para criar a instância da classe concreta que implementa tal interface. Na figura 4.1 é possível identificar como as referências se dão. É possível ver que o módulo que contém as interfaces principais não possuem qualquer referência para fora do módulo e qualquer dependência da API *GO WhatsApp* é isolada dentro da implementação concreta e pode ser substituída sem causar grandes impactos na interface ou nos usuários da API *WppScrapper*.

Para que haja acesso às informações do WhatsApp, é necessária a autenticação com o uso do *QRCode*. O *WppScrapper* é capaz de recuperar esse *QRCode* através da *GO WhatsApp*. Como é possível ver no diagrama de sequência da figura 4.3, onde está ilustrado o processo de autenticação, o programa recebe de forma assíncrona as informações do *QRCode* e deve passar adiante para o usuário. Uma vez que

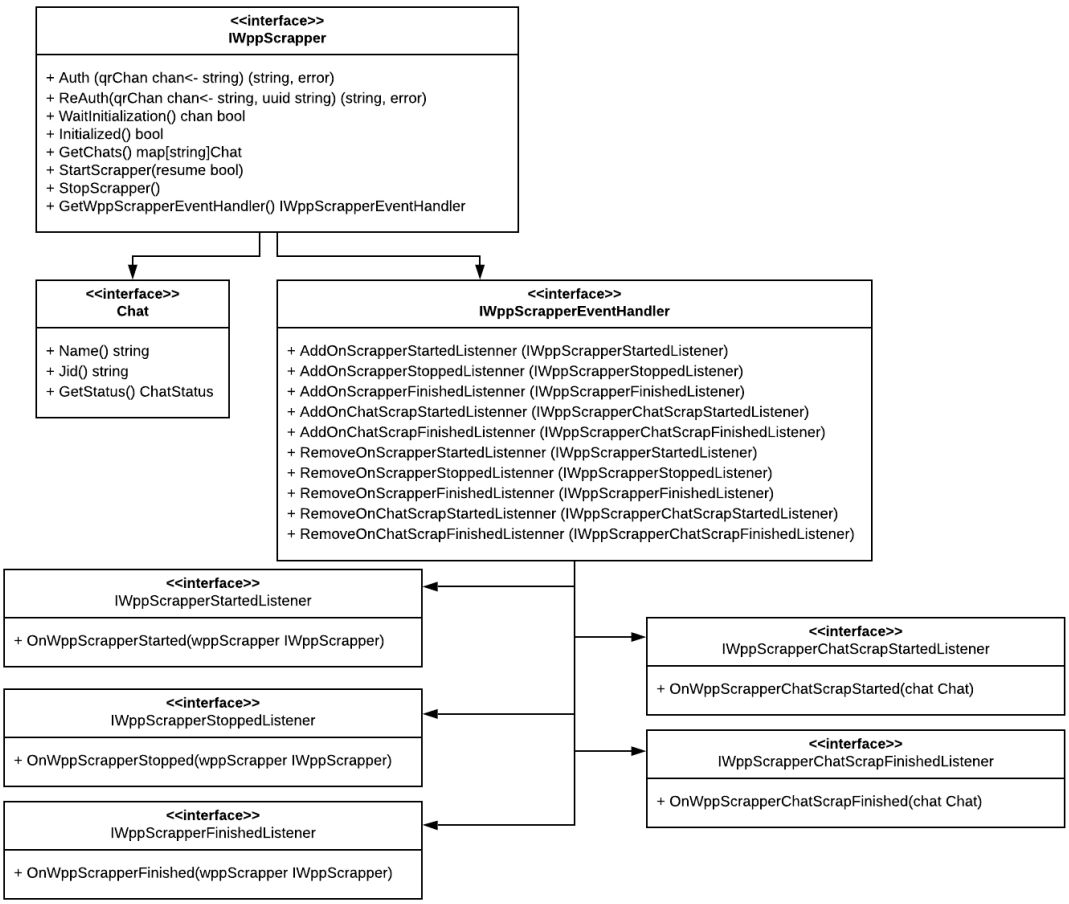

Figura 4.2: Diagrama de Classes com as interfaces que compõem o *WppScrapper*

o usuário realize o escaneamento do *QRCode*, a função é capaz de retornar com sucesso. Não está ilustrado na figura, mas o usuário pode demorar mais que um tempo determinado; caso isso, ocorra a função irá retornar com erro de *timeout*.

Além das funções necessárias para que se requisite a execução de determinadas ações, como a autenticação (*Auth*), inicialização da rotina de extração das mensagens (*StartScrapper*), pausa dessa rotina (*StopScrapper*) etc, o *WppScrapper* também define uma série de eventos necessários para que a aplicação usuária possa responder adequadamente. Por exemplo, para saber quando a extração terminou, a aplicação usuária deverá implementar a interface *IWppScrapperFinishedListerner* e registrar esse objeto junto ao objeto do tipo *IWppScrapperEventHandler* provido pelo *WppScrapper* através da função *GetWppScrapperEventHandler*. Uma vez que esse objeto estiver registrado, ele terá a função *OnWppScrapperFinished* chamada quando a

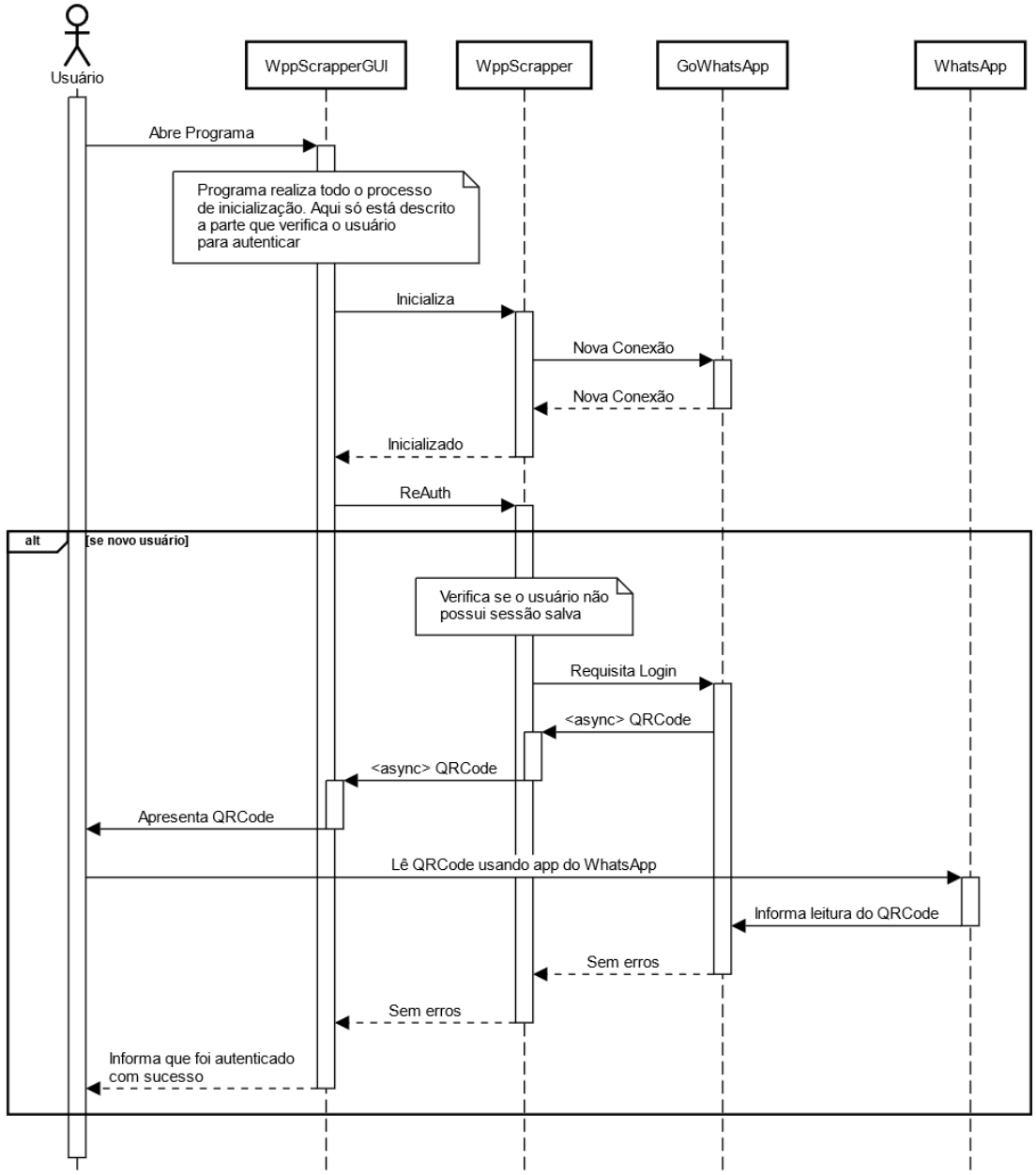

Figura 4.3: Diagrama da sequência que demonstra o processo de autenticação de um usuário sem sessão salva.

extração foi concluída. A figura 4.4 exemplifica esse processo. De maneira análoga, ocorre com os demais eventos, todos descritos no diagrama de classes da figura 4.2 com o sufixo *Listener*.

Na próxima sessão, estará descrito o processo de implementação da *WppScrapperGUI*. Junto a isso será possível encontrar mais exemplos de uso da *WppScrapper*.

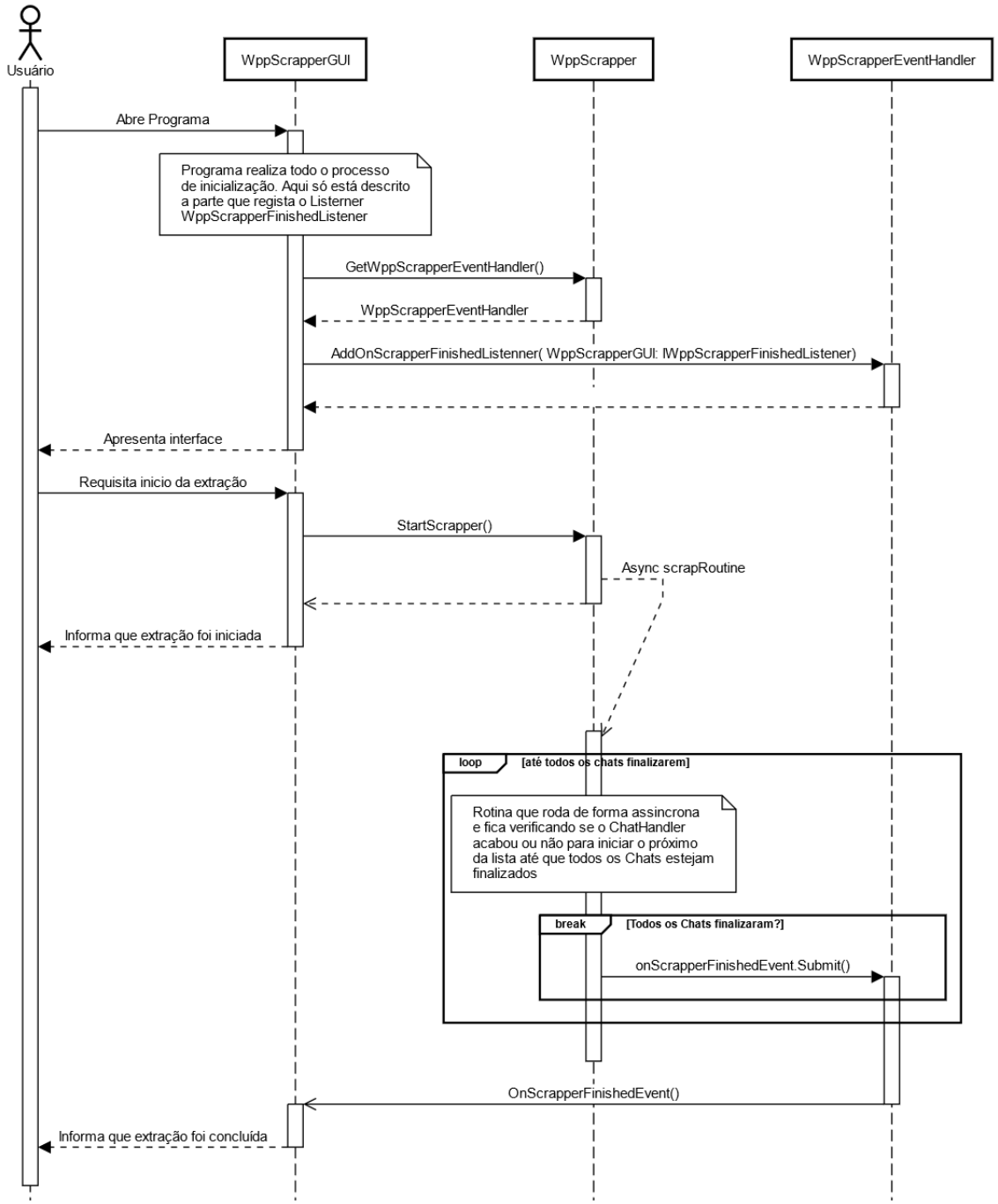

Figura 4.4: Diagrama da sequência que demonstra o processo de registrar o *Listerner IWppScrapperFinishedListerner* até o momento que o mesmo é acionado.

4.2 Implementação da WppScrapperGUI

Para realizar a implementação do *WppScrapperGUI*, o primeiro desafio está na escolha de uma biblioteca de construção interface gráfica escrita na linguagem GO

que atenda as necessidades. Ou seja, essa biblioteca deveria permitir que o programa resultante seja executado nos diferentes sistemas operacionais, (*e.g. Windows*, *Linux*), que seja estável, tenha uma interface de programação bem documentada e estruturada, possua o conjunto de funcionalidades e elementos visuais e interativos necessários, (*e.g Botão, Texto, Tabela, Imagem*) e que a interface gráfica gerada por ela seja agradável aos olhos do usuário.

Dentre as bibliotecas estudadas e testadas, aquela se mostrou melhor para atender as necessidades do projeto foi a *Fyne.io*[3]. A facilidade para gerar os programas executáveis para diferentes plataformas, a simplicidade da biblioteca de construção da interface gráfica, boa documentação, o *design* padrão aceitável da interface gerada e ser de código aberto foram as características que foram decisivas para que tenha sido a escolhida.

Para atender aos requisitos definidos no capítulo 3.3, a interface de usuário implementa duas telas. A primeira é a tela de autenticação, onde o QRCode é apresentado para o usuário e a segunda tela é onde, uma vez autenticado, o usuário poderá iniciar a extração das mensagens. O *WppScrapperGUI* ainda apresenta uma terceira tela de carregamento. Nela não é possível nenhuma interação e é apresentada enquanto alguma operação de transição entre as telas está sendo realizada.

Na figura 4.5 está apresentado o trecho do código responsável pela iniciação do programa. É iniciado como variáveis globais as referências às partes principais da biblioteca do *Fyne.io*, pelas quais será possível alterar o conteúdo da janela visível ao usuário, e com uma referência a uma instância do *IWppScrapper*, que foi explicado no capitulo 4.1. Na função *main* é configurado o tamanho da janela e então chamado a função *ShowAndRun* do *Fyne.io*, que é quando uma janela é aberta, apresentada ao usuário e toda a biblioteca do *Fyne.io* passa a ser executada.

Ainda na figura 4.5 é possível ver que há a chamada da função *showInitialLoadingView*. Essa função somente apresenta na janela um texto informando ao usuário que o programa está carregando. Nesse mesmo trecho de código tem a chamada assíncrona para função *initializeWppScrapper*, que está inteiramente ilustrada na

[3]https://fyne.io/

```
1 var application = app.NewWithID("br.ufrrj.wppscrappergui")
2 var wdw = application.NewWindow("WppScrapper GUI")
3
4 var wppScrapper = wppscrapperimp.InitializeConnection().(wppscrapper.IWppScrapper)
5
6 func main() {
7     application.SetIcon(theme.FyneLogo())
8
9     showInitialLoadingView()
10
11    wdw.Resize(fyne.NewSize(640, 460))
12    go initializeWppScrapper()
13    wdw.ShowAndRun()
14 }
```

Figura 4.5: Trecho de inicialização do *WppScrapperGUI*.

figura 4.6. Essa função é responsável por requisitar o *QRCode* para o *WppScrapper* e então passá-lo para a função que irá apresentá-lo na janela. Essa função está ilustrada na figura 4.7 e a janela resultado da função na figura 4.8. A função ainda trata que, caso ocorra algum erro, ela será chamada novamente de forma recursiva para que um novo *QRCode* seja requisitado apresentado ao usuário.

Caso não ocorra erros, ou seja, o usuário realize a leitura do *QRCode* corretamente, é apresentado ao usuário a janela de *carregando* novamente enquanto o WppScrapper não termina de inicializar. Uma vez inicializado, o programa chama a função *showMainView*, que é responsável por requisitar a apresentação da segunda tela, a tela onde é possível iniciar a extração das mensagens.

A função *showMainView* apenas cria uma nova instancia do tipo *MainView* e requisita que a mesma apresente seu conteúdo através da função *Show*. Esse tipo é responsável por construir a tela que permite ao usuário iniciar a extração do conteúdo, pausar a mesma e recomeçar, caso deseje. Nessa janela o usuário também tem acesso ao estado atual do programa, *e.g.* executando a extração, esperando, finalizado, tal como informação do estado de cada *Chat*.

Nessa parte do programa é possível encontrar bons exemplos de uso da API *WppScrapper*. Na função *buildHeader*, parcialmente apresentada na figura 4.9, é possível ver o código executado quando os botões são acionados pelo usuário. Já na

```go
func initializeWppScrapper() {

    qrCode := make(chan string)
    go func() {
        showQrCodeView(<-qrCode)
    }()

    _, err := wppScrapper.ReAuth(qrCode, application.UniqueID())
    if err != nil {
        //TODO: tratar erro
        fmt.Println("Error trying to auth", err)

        initializeWppScrapper()
        return
    }

    showInitialLoadingView()

    if !wppScrapper.Initialized() {
        <-wppScrapper.WaitInitialization()
    }

    showMainView()
}
```

Figura 4.6: Trecho de código que trata a autenticação do usuário.

```go
func showQrCodeView(qrCode string) fyne.CanvasObject {

    title := canvas.NewText("Scan the QRCode using your WhatsApp app", theme.ForegroundColor())
    qrImage := getQrCodeImage(qrCode)
    qrImage.FillMode = canvas.ImageFillContain
    qrImage.Refresh()
    cont := container.NewBorder(title, nil, nil, nil, qrImage)

    wdw.SetContent(cont)
    return cont
}

```

Figura 4.7: Trecho de código que apresenta o QRCode ao usuário.

figura 4.10 encontramos exemplo de código que implementa os listeners definidos na API *WppScrapper* e usa a mesma API para assinar a instancia e passar a 'escutar' caso algum desses eventos ocorra. Assim o programa pode responder e esconder determinados botões a depender do estado do programa, como por exemplo esconde

Figura 4.8: Janela que apresenta o QRCode ao usuário.

os botões que iniciam a extração das mensagens quando o programa já está extraindo as mensagens e apresentar apenas o botão que pausa essa execução.

4.3 Demonstração do WppScrapperGUI

Com intuito de realizar uma pequena demonstração do *WppScrapperGUI* em ação, essa secção descreve um procedimento bastante simplificado de coleta de mensagens de grupos públicos de WhatsApp. Os *links* para acesso aos grupos foram encontrados disponíveis no *Google* e no *Facebook*. Para a realização da operação, foi utilizada uma conta de WhatsApp criada apenas para essa finalidade.

O primeiro passo realizado foi a inscrição dessa conta nos grupos públicos. Como a intenção era realizar apenas uma pequena amostra, foi feita a inscrição em 10 grupos de temas variados, *e.g.* futebol, jogos. As inscrições foram feitas às 20h de um dia e deixou-se coletando as mensagens até às 13h do dia seguinte, quando foi feito o uso do *WppScrapperGUI* para realizar a extração das mensagens.

Nas figuras 4.11, 4.12 e 4.13 encontra-se exemplo do programa aberto em seus três diferentes estados. No primeiro o programa acabara de ser aberto e tem listado todas as conversas encontradas em estado de espera, apresentado na figura 4.11. Em seguida, após acionado pelo usuário a realizar a extração, o programa modifica parte da sua interface para indicar seu estado, apresentar as opções de ação possível e indicar a conversa que está sendo extraído no momento, tal como seu indicador

```
1 func (m *MainView) buildHeader() *fyne.Container {
2 ...
3     m.btnRestartScrap = &widget.Button{
4         Text: "Start Scrapper",
5         Icon: theme.DownloadIcon(),
6         OnTapped: func() {
7             m.scrappedChatsCount = 0
8             wppScrapper.StartScrapper(false)
9         },
10    }
11    m.btnStartScrap = &widget.Button{
12        Text: "Restart Scrapper",
13        Icon: theme.DownloadIcon(),
14        OnTapped: func() {
15            wppScrapper.StartScrapper(true)
16        },
17    }
18
19    m.btnStopScrap = &widget.Button{
20        Text: "Stop Scrapper",
21        Icon: theme.CancelIcon(),
22        OnTapped: func() {
23            wppScrapper.StopScrapper()
24        },
25    }
26 ...
27 }
```

Figura 4.9: Parte da função *buildHeader* do tipo *MainView* que cria os botões que inicia a extração, pausa a mesma e reinicia. Parte da função foi omitida.

de progresso passa a indicar uma porcentagem referente a quantidade de conversas extraídas frente ao total de conversas a se extrair, ilustrado na figura 4.12. Por ultimo o programa apresenta a tela de finalizado, onde todas as conversas já foram extraídas e se encontram em formato CSV no computador do usuário na mesma pasta em que se encontra o executável do programa, ilustrado na figura 4.13.

Neste exemplo, foi possível coletar e extrair um total de 1472 mensagens de 10

```
1
2  func (m *MainView) Show() {
3
4      eventHandler := wppScrapper.GetWppScrapperEventHandler()
5      eventHandler.AddOnChatScrapFinishedListener(m)
6      eventHandler.AddOnScrapperFinishedListener(m)
7      eventHandler.AddOnScrapperStartedListener(m)
8      eventHandler.AddOnScrapperStoppedListener(m)
9
10     m.buildView()
11 }
12
13 func (m *MainView) OnWppScrapperStarted(wppScrapper wppscrapper.IWppScrapper) {
14     m.updateButtons(true)
15     m.lblStatus.SetText("Status: Running")
16 }
17
18 func (m *MainView) OnWppScrapperStopped(wppScrapper wppscrapper.IWppScrapper) {
19     m.updateButtons(false)
20     m.lblStatus.SetText("Status: Stopped")
21 }
22 func (m *MainView) OnWppScrapperFinished(wppScrapper wppscrapper.IWppScrapper) {
23     m.updateButtons(false)
24     m.lblStatus.SetText("Status: Finished")
25 }
26
27 func (m *MainView) OnWppScrapperChatScrapFinished(chat wppscrapper.Chat) {
28     m.scrappedChatsCount++
29     progress := float64(m.scrappedChatsCount) / float64(len(wppScrapper.GetChats()))
30     m.cnvProgressBar.SetValue(progress)
31     wdw.Content().Refresh()
32 }
33
```

Figura 4.10: Trecho de código que define funções para o tipo *MainView*. Nesse trecho é possível encontrar a implementação das interfaces da API *WppScrapper*.

diferentes grupos que possuem, no total, 1311 usuários não necessariamente distintos. Na tabela 4.1 é possível encontrar esses dados descritos para cada grupo tal como seu nome.

A seguir temos exemplos de arquivos extraídos, em todas as imagens os identificadores, que possuem o número de celular em sua composição, foram borrados para preservar privacidade daqueles que enviaram as mensagens e que participam do grupo em questão. Na figura 4.14 é possível observar, entre outras informações, as mensagens enviadas no grupo "Free Fire Campeonato". Com dados do mesmo grupo, na figura 4.15 está ilustrado as informações do grupo em si, como o identificador do seu criador e a descrição do grupo. O arquivo com os membros deste grupo está ilustrado na imagem 4.14.

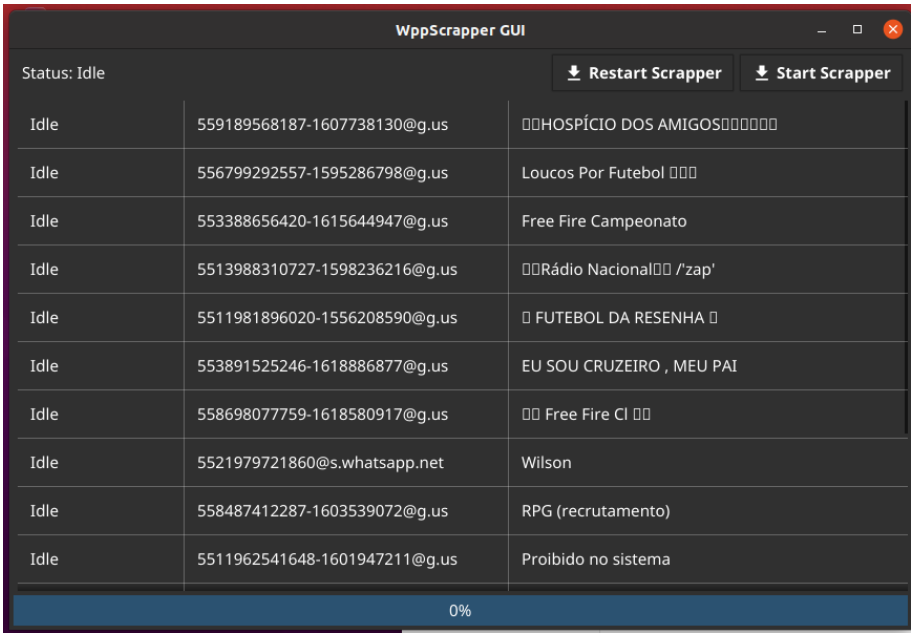

Figura 4.11: Captura de tela do *WppScrapperGUI* em estado de espera logo após de ter o usuário validado e carregar todas as informações.

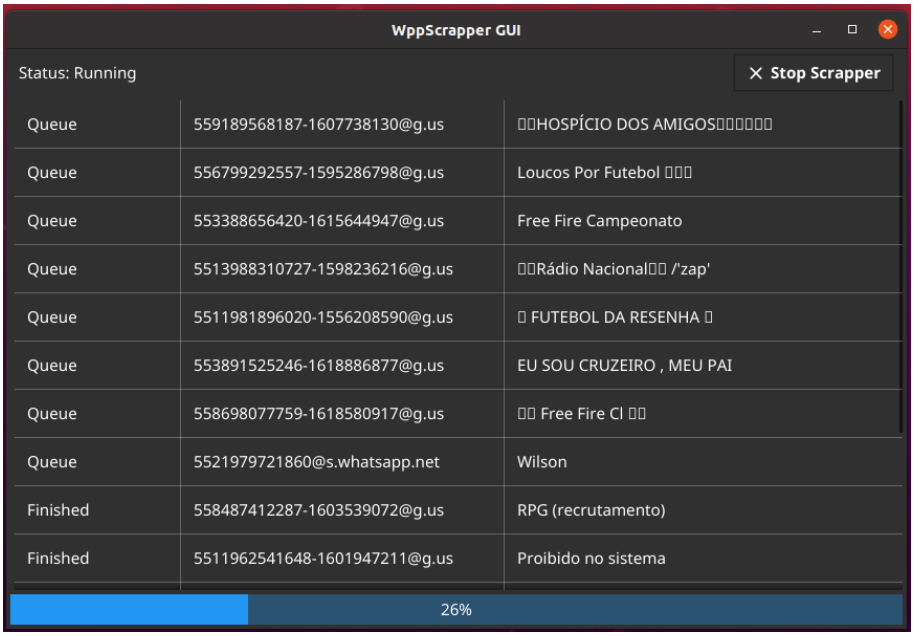

Figura 4.12: Captura de tela do *WppScrapperGUI* rodando a rotina de extração das mensagens, onde 26% das conversas já foram extraídas.

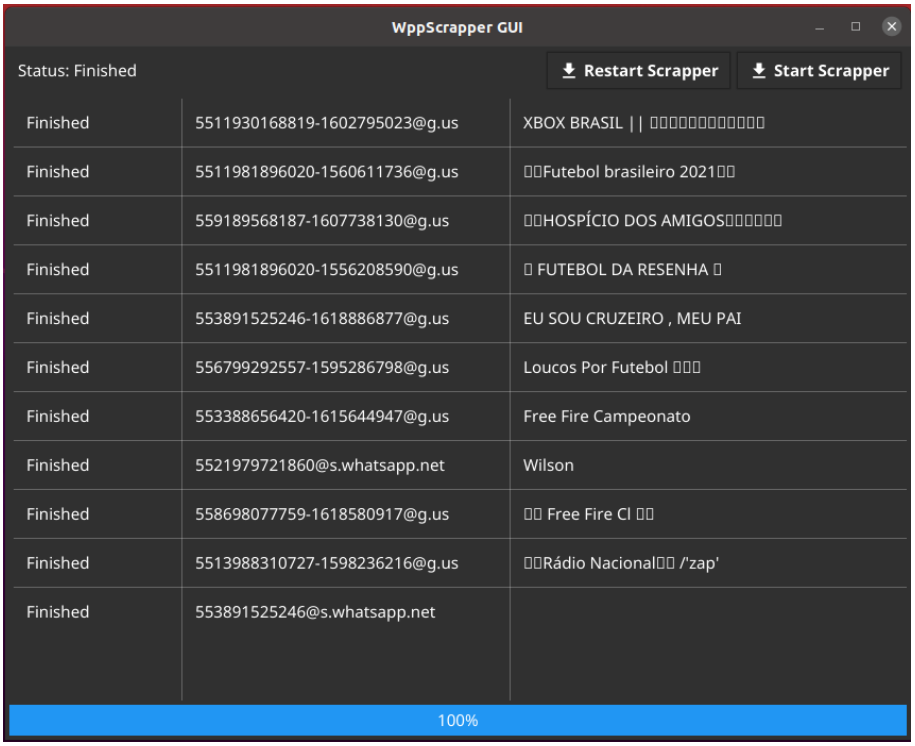

Figura 4.13: Captura de tela do *WppScrapperGUI* após finalizar de extrair todas as conversas.

Tabela 4.1: Tabela de nome dos grupos que tiveram mensagens extraídas com a quantidade de membros e a quantidade de mensagens extraídas por grupo.

Nome do Chat	Quantidade de Membros	Quantidade de Mensagens
Free Fire Campeonato	150	65
EU SOU CRUZEIRO , MEU PAI	10	27
Loucos Por Futebol	176	25
Free Fire Cl	61	32
HOSPÍCIO DOS AMIGOS	233	1301
XBOX BRASIL	71	19
FUTEBOL DA RESENHA	206	0
Futebol brasileiro 2021	206	0
Rádio Nacional	198	3

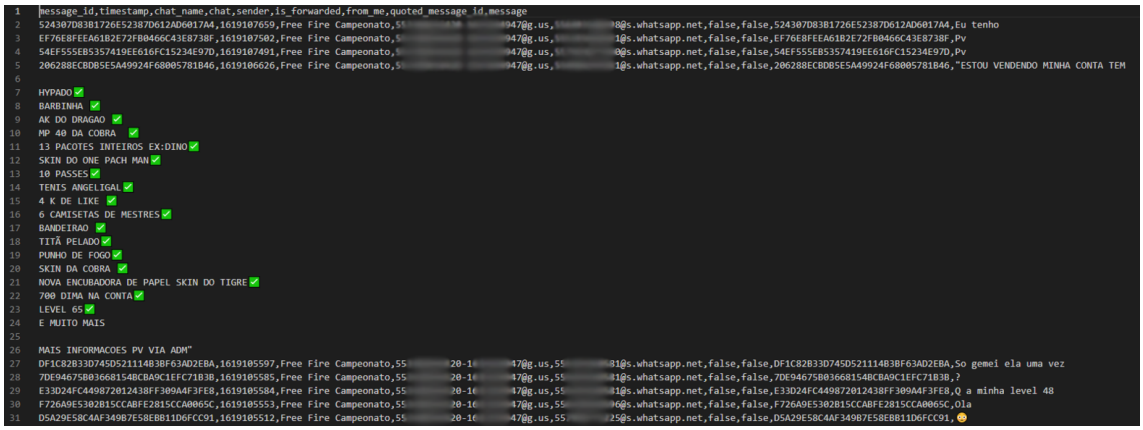

Figura 4.14: Captura de tela da parte inicial do conteúdo de um arquivo CSV de mensagens gerado pela extração.

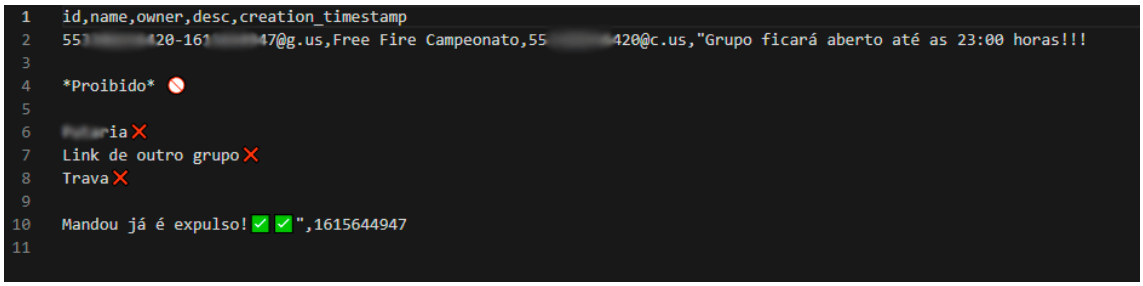

Figura 4.15: Captura de tela do conteúdo um arquivo *CSV* de descrição de um grupo gerado pela extração.

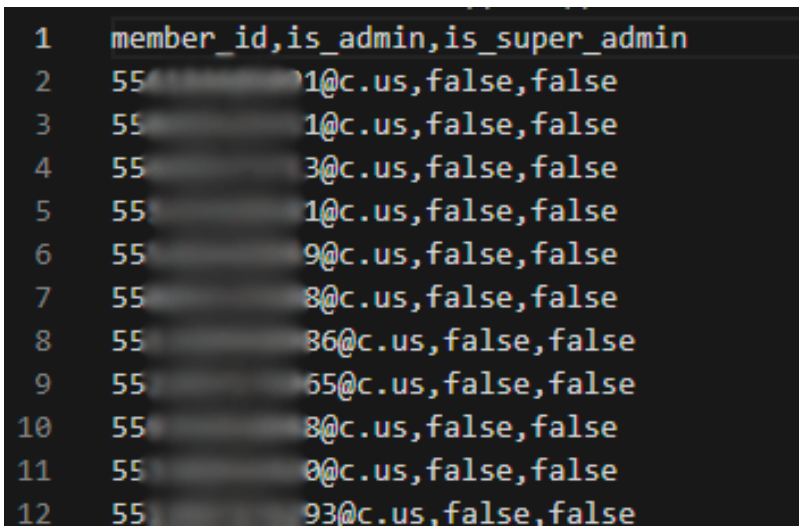

Figura 4.16: Captura de tela da parte inicial do conteúdo um arquivo *CSV* de lista de membros de um grupo gerado pela extração.

Capítulo 5

Conclusão

Estudar como se dá as interações entre os usuários do WhatsApp[1] se tornou uma ferramenta poderosa para que pesquisadores e jornalistas possam melhor compreender os eventos da sociedade. Apesar de o aplicativo de mensagens instantânea não disponibilizar nenhuma ferramenta ou *API* para que essas pessoas possam realizar seus estudos, o presente trabalho apresentou uma alternativa que pode ser usada.

No presente trabalho foi apresentado duas ferramentas, a API WppScrapper e a aplicação de interface gráfica para essa API chamada WppScrapperGUI. A aplicação se mostrou capaz de realizar a extração de mensagens de uma conta do WhatsApp, o que pode ser considerada uma tarefa essencial para realização de novos trabalhos acadêmicos que objetivem compreender essas mensagens. Se diferenciando de outras alternativas disponíveis, a aplicação possui uma interface gráfica, o que a torna mais acessível.

Enquanto outras ferramentas similares necessitam de conhecimentos de programação para instalar e utilizar, a aplicação apresentada nesse trabalho consegue ser usada apenas executando um arquivo binário que pode ser baixado e sua operação pode ser realizada inteiramente através de uma interface gráfica. A disponibilização da *API* WppScapper, que é análoga, porém agnóstica, a interface gráfica pode possibilitar que pesquisadores com conhecimento de programação a sua utilização em computadores

[1] www.whatsapp.com

remotos ou até a criação de novas formas de interface, *e.g. WebSocket*, *REST*, *CLI* etc.

Apesar de útil para coletar as mensagens textuais do WhatsApp, ambas as ferramentas apresentadas são limitadas e incapazes de coletar as imagens, mensagens de áudio ou de vídeo do WhatsApp. As ferramentas também não são capazes de extrair outras informações que podem ser úteis, *e.g.* imagem do grupo, foto de usuários.

Para trabalhos futuros, além de implementar melhorias na aplicação para que essas limitações não estejam mais presentes, seria interessante realização de estudos, que utilize das mensagens extraídas e de técnicas de mineração de texto, com o objetivo de extrair informações relevantes, *e.g* as tendências mais comentadas dentro dos grupos públicos de um determinado tópico.

Outras melhorias ainda podem ser feitas, no futuro, na aplicação e na API. A possibilidade de configurar um serviço armazenamento em nuvem, *e.g. Amazon S3*[2], para armazenar os arquivos gerados pela extração, pode ser uma funcionalidade interessante, tal como a possibilidade de instalar a ferramenta em uma hospedagem remota, podendo proporcionar mais velocidade e disponibilidade. A possibilidade de extrair as mensagens em outros formatos, (*e.g. Json*[3]) também pode ser desejada.

Tanto o *WppScrapper*[4] quanto o *WppScrapperGUI*[5] estão disponíveis no *Github* em código aberto e contribuições são muito bem vindas. A melhoria das práticas de integração e entrega contínua, já parcialmente implementadas fazendo uso do *Github Actions*[6], ou adição de testes unitários em ambas as ferramentas são apenas alguns exemplos de contribuições. Outros tipos de contribuição partindo de necessidades específicas de cada uso futuro também podem ser propostas e incorporadas ao código fonte.

Existe uma infinidade de estudos que podem ser feitos utilizando os dados

[2]https://aws.amazon.com/pt/s3/
[3]https://json.org/json-pt.html
[4]https://github.com/ribeiroferreiralucas/wpp-scrapper
[5]https://github.com/ribeiroferreiralucas/wpp-scrapper-gui
[6]https://github.com/features/actions

possíveis de serem extraídos da aplicação aqui apresentada e muitas melhorias possíveis de serem feitas nela. O presente trabalho espera ter contribuído para que mais pesquisadores possam, além de contribuir com a evolução dessa ferramenta, esclarecer mais o que acontece no ambiente interno da rede social e que a sociedade possa fazer bom uso de tal poder.

Referências

AGGARWAL, C. C.; ZHAI, C. *Mining text data*. [S.l.]: Springer Science & Business Media, 2012.

BANERJEE, R. Website scraping. *Happiest Minds. Np*, 2014.

BOEING, G.; WADDELL, P. New insights into rental housing markets across the united states: Web scraping and analyzing craigslist rental listings. *Journal of Planning Education and Research*, SAGE Publications Sage CA: Los Angeles, CA, v. 37, n. 4, p. 457–476, 2017.

CAETANO, J. A. et al. Analyzing and characterizing political discussions in whatsapp public groups. *arXiv preprint arXiv:1804.00397*, 2018.

CAMILO, C. O.; SILVA, J. C. d. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. *Universidade Federal de Goiás (UFC)*, p. 1–29, 2009.

CÔRTES, S. da C.; PORCARO, R. M.; LIFSCHITZ, S. *Mineração de dados-funcionalidades, técnicas e abordagens*. [S.l.]: PUC, 2002.

DASTIDAR, B. G.; BANERJEE, D.; SENGUPTA, S. An intelligent survey of personalized information retrieval using web scraper. *IJ Educ. Manag. Eng*, 2016.

DEOTTI, F. *Pesquisa Ipsos-Truckpad com caminhoneiros*. 2018. Disponível em: <https://www.ipsos.com/pt-br/pesquisa-ipsos-truckpad-com-caminhoneiros>.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI magazine*, v. 17, n. 3, p. 37–37, 1996.

GARIMELLA, K.; TYSON, G. Whatapp doc? a first look at whatsapp public group data. In: *Twelfth International AAAI Conference on Web and Social Media*. [S.l.: s.n.], 2018.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011.

MACHADO, C. et al. A study of misinformation in whatsapp groups with a focus on the brazilian presidential elections. In: ACM. *Companion Proceedings of The 2019 World Wide Web Conference*. [S.l.], 2019. p. 1013–1019.

MARFIANTO, A.; RIADI, I. Whatsapp messenger forensic analysis based on android using text mining method. *International Journal of Cyber-Security and Digital Forensics*, The Society of Digital Information and Wireless Communications, v. 7, n. 3, p. 319–327, 2018.

MITCHELL, R. *Web Scraping with Python: Collecting More Data from the Modern Web*. [S.l.]: "O'Reilly Media, Inc.", 2018.

NEWMAN, N. et al. *Reuters institute digital news report 2019*. [S.l.]: Reuters Institute for the Study of Journalism, 2019. v. 2019.

OLSON, D. L.; DELEN, D. *Advanced data mining techniques*. [S.l.]: Springer Science & Business Media, 2008.

RESENDE, G. et al. A system for monitoring public political groups in whatsapp. In: ACM. *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*. [S.l.], 2018. p. 387–390.

ROSSI, A. *Como o WhatsApp mobilizou caminhoneiros, driblou governo e pode impactar eleições*. 2018. Disponível em: <https://www.bbc.com/portuguese/brasil-44325458>.

SEVIT, D. *Mobile Messaging App Map – February 2018*. 2018. Disponível em: <https://www.similarweb.com/blog/mobile-messaging-app-map-2018>.

SHI, Z.; MA, H.; HE, Q. Web mining: Extracting knowledge from the world wide web. In: *Data Mining for Business Applications*. [S.l.]: Springer, 2009. p. 197–208.

TARDAGUILA, C. *Fotos (velhas) de universitários nus inundam WhatsApp para 'provar' a 'balbúrdia' apontada por Weintraub*. 2019. Disponível em: <https://epoca.globo.com/fotos-velhas-de-universitarios-nus-inundam-whatsapp-para-provar-balburdia-apontada-por-weintraub-23661819>.