Automação para Telegram: publicação de notícias a partir de um site acadêmico

Gustavo Moura da Costa [1], **Daniela Marques** [1]

[1]Instituto Federal de Educação, Ciência e Tecnologia– Campus Hortolândia (IFSP)
Avenida Thereza Ana Cecon Breda, N.o 1896, Vila São Pedro, Hortolândia- SP

gustavomourac2@gmail.com, marquesdaniela@ifsp.edu.br

Abstract. *Currently, the Hortolândia campus has a website where news are posted and made available in a list. Since it does not contain a notification to its users, it is difficult to access information. Therefore, this article describes the development of a system for disseminating the news available on the IFSP Campus Hortolândia website in an instant messaging application, like Telegram. The system was developed with the aim of automating and facilitating the dissemination of content posted on the campus website.*

Resumo. *Atualmente, o campus Hortolândia possui um site onde as notícias são postadas e disponibilizadas em uma lista. Por não conter uma notificação aos seus usuários, é difícil o acesso às informações. Portanto, este artigo descreve o desenvolvimento de um sistema para divulgação das notícias disponibilizadas no site do IFSP Campus Hortolândia em um aplicativo de mensagens instantâneas, o Telegram. O sistema foi desenvolvido com o intuito de automatizar e facilitar a divulgação do conteúdo postado no site do campus.*

1. Introdução

Atualmente, muitas das informações ou notícias relevantes para o IFSP Campus Hortolândia estão centralizadas e são postadas ou atualizadas, exclusivamente, por meio do site. E, do modo atual, não há notificação alguma para interessados que querem ver resultado de chamadas ou eventos. Os próprios alunos para acompanharem processos como bolsas de ensino, assistência estudantil, oportunidades de estágio, por exemplo, não possuem alternativa senão ter que checar periodicamente o site.

De acordo com um artigo do [DiárioPopular 2021], "os aplicativos de mensagens instantâneas se tornaram parte crucial na rotina diária de bilhões de pessoas". Não há dúvidas que estes aplicativos, como Telegram, WhatsApp ou Facebook Messenger, entre outros, são altamente utilizados e, ao menos um deles estará instalado no celular da população. Para reforçar essa afirmação, de acordo com a pesquisa feita em 2021, houve cerca de 2,91 bilhões de usuários desses aplicativos ao redor do mundo em 2020, com uma previsão de um aumento de 6,1% para 2021 [Enberg 2021].

Segundo a notícia postada no site Agência Brasil, de acordo com o IBGE, 95,7% dos brasileiros que têm acesso à internet usam a rede para enviar ou receber mensagens de texto, voz ou imagens por aplicativos de mensagens, como o WhatsApp, Telegram, entre outros [AgenciaBrasil 2020].

Identificado o avanço existente do uso de aplicativos de mensagens e pensando na dificuldade ao acessar as informações disponibilizadas no site do IFSP - campus Hortolândia, pensou-se na criação de um sistema para automatizar o processo de divulgações das notícias postadas no site do campus.

O objetivo deste trabalho é o desenvolvimento de um sistema de notificação automatizado das notícias postadas no site do IFSP de Hortolândia para um aplicativo de mensagens, nesse caso o Telegram, facilitando o acesso das notícias aos usuários. Para realizar este objetivo foi utilizado *Web Scraping*, uma forma de mineração de dados, para extrair os dados das notícias do site do campus Hortolândia e enviar a notícia extraída por um aplicativo de mensagens, a fim de aproveitar sua funcionalidade de notificação.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico dos conceitos que foram estudados para o desenvolvimento do projeto; a Seção 3 mostra trabalhos correlatos; a Seção 4 apresenta a metodologia, descrevendo os passos seguidos para conseguir escolher o aplicativo e o meio para produção da *API* que aplicaria o *Web Scraping* e o consumidor para enviar as informações para o aplicativo de mensagens; na Seção 5 temos o desenvolvimento do sistema, o passo a passo do desenvolvimento do código e da extração dos dados da página. Por fim, na Seção 6 são feitas as conclusões finais sobre o projeto.

2. Referencial Teórico

Esta seção apresenta o referencial teórico dos conceitos que foram estudados para o desenvolvimento do projeto.

2.1. Site do Campus Hortolândia

O site que foi aplicada a coleta dos dados é a página de Últimas notícias do IFSP Campus Hortolândia[1], que pode ser acessada a partir da página principal do campus[2] ao clicar em "Acessar a lista completa", marcado em amarelo na Figura 1. A página de Últimas notícias possui um modelo padrão utilizado pelos campus da rede do IF, apresentado na Figura 2, podendo ter alguns detalhes que diferem de campus para campus, tanto visualmente quanto no código fonte.

Dessas diferenças, foi visto no site de outros campi, principalmente, a presença de imagens e a quantidade de notícias que aparecem na página que, no caso do campus de Hortolândia, não há imagens na página. O que seria interessante possuir, já que hoje, ao mandar um *link* por um aplicativo, alguns identificam e podem apresentar o título da página ou vídeo, uma prévia do texto ou descrição e uma imagem da página ou do vídeo. E a quantidade de notícias pode influenciar num histórico maior dependendo dos requisitos definidos para a aplicação.

Apesar das diferenças, como páginas web, dificilmente possuem algum sistema de notificação e caso a página possua um, geralmente é atrelado ao navegador. Isto faz com que, para acompanhar as notícias, tenha que acessar toda vez o site do campus para observar se houve alguma atualização. Para isso foi aplicado *Web Scraping* na página de últimas notícias.

[1]https://hto.ifsp.edu.br/institucional/*index*.php/ultimas-noticias

[2]https://hto.ifsp.edu.br/institucional/

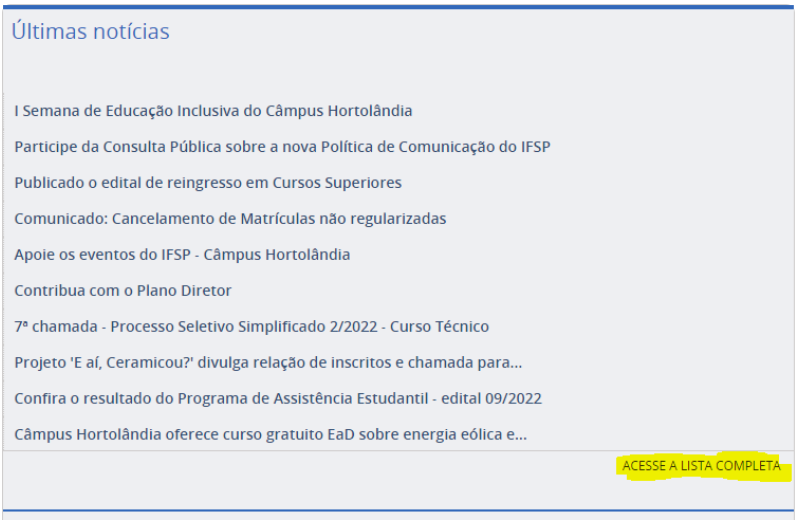

Figura 1. *Card* das Últimas Notícias encontrado na página inicial do site.

2.2. *Web Scraping*

Web Scraping, coleta de dados *web* ou raspagem *web*, é o processo de extração de uma representação estrutural de dados de um site. A linguagem de formatação usada para configurar os dados nas páginas da *web* pode apresentar variabilidade do HTML, pois as técnicas existentes para raspagem da *web* são baseadas em marcação. Uma alteração no HTML pode levar à coleta de dados incorretos [Aydin 2018].

Há diversos meios de aplicar *Web Scraping* numa página. Durante as pesquisas foram encontradas bibliotecas em Python e em Node que convertem todo conteúdo da página, seu código fonte, em um objeto manipulável. Com um conhecimento em HTML, realiza-se a leitura do código fonte da página para identificar o conteúdo que se deseja extrair e os componentes do HTML, as denominadas *tags*. Estas são representadas com <*tag* ...> e seus atributos podem auxiliar a extração da informação requerida, sendo eles: *class, id, name*, entre outros.

2.3. *API*

Uma *API* é um conjunto de definições e protocolos usado no desenvolvimento e na integração de aplicações. Às vezes, as APIs são descritas como um contrato entre um provedor e um usuário de informações, estabelecendo o conteúdo exigido pelo consumidor (a chamada) e o conteúdo exigido pelo produtor (a resposta). Por exemplo, uma *API* de filmes, que solicita que o usuário forneça um nome de algum filme e esse produtor retorna em diversas partes, contendo por exemplo: sinopse, avaliação, duração, categoria, entre outros. Ou que o desenvolvedor deseje retornar para aquele recurso [RedHat 2020].

3. Trabalhos Correlatos

Esta Seção apresenta dois trabalhos correlatos encontrados que divulgam as notícias por mensagens instantâneas. Os códigos fontes desses trabalhos não foram encontrados ou não se encontram disponíveis.

Figura 2. Imagem do site de Últimas Notícias

3.1. IFSP - notícias geral

Há um canal, feito no Telegram, de notícias que compila o conteúdo divulgado nas seções "Últimas notícias e Notícia - Servidores"[3], apresentando notícias e eventos divulgadas no site do campus de São Paulo.

Ao realizar uma análise nas notícias e principalmente nas datas entre o publicado originalmente no site (Figura 3) e a notícia divulgada pelo Telegram (Figura 4), conclui-se que há um atraso entre a publicação da notícia e o envio da mensagem no Telegram. Este atraso é bem variável podendo ser de alguns minutos ou horas, notou-se durante os testes diferenças de 4 minutos até 3 horas.

As mensagens seguem um padrão tendo na primeira linha da mensagem um título apresentando de onde foi retirada a notícia, se é do "Últimas notícias" ou "Últimas Notícia - Servidores". Em seguida, tem-se o título da notícia em que foi anexado um *link* a ele. Por fim, logo abaixo tem-se uma prévia do conteúdo do site da qual o *link* se refere, funcionalidade essa encontrada em diversos aplicativos, na qual apresenta o título da notícia novamente e uma foto encontrada na página.

Não há como afirmar se é feito o uso de alguma automação para a divulgação das notícias, como no objetivo deste trabalho, ou se é feito manualmente.

[3]https://www.ifsp.edu.br/

Figura 3. Imagem do canal retirado da versão mobile do Telegram.

Figura 4. Imagem das seções da onde as notícias são retiradas pelo IFSP - notícias geral.

3.2. Canal do portal de notícias do G1

A plataforma de notícias, G1[4], disponibiliza um canal no Telegram para disseminar seu conteúdo. Contudo, neste canal, diferentemente do trabalho correlato descrito anteriormente existe uma variedade de conteúdo, não aparentando retirar as notícias de uma seção específica. Além disso, a plataforma divulga seu conteúdo de diversas formas, como em formato de áudio e de resumo diário das principais notícias do dia, como pode ser visto na Figura 5.

[4]https://g1.globo.com

Assim como o trabalho correlato descrito anteriormente, haviam diferenças entre o momento em que o artigo foi publicado e quando foi divulgado pelo aplicativo, apresentando tempos variados. Entretanto, como é uma grande plataforma de notícias tendo um volume bem maior do que a página de notícia de um campus, são escolhidas apenas algumas notícias para serem enviadas pelo aplicativo. Geralmente são enviados por volta de meio dia, e apresentam notícias que foram postadas antes desse período. Com isso em mente, não é possível acompanhar todas as notícias.

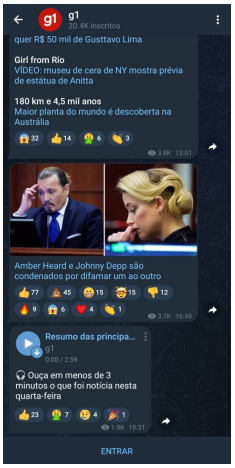

Figura 5. Imagem da visão do canal g1 retirado da versão mobile do Telegram.

4. Metodologia

Inicialmente foi feita uma pesquisa comparando diferentes aplicativos de mensagem instantânea, visando escolher um deles para o desenvolvimento do sistema. Em seguida, outra pesquisa foi realizada para entendimento de como implementar o rastreamento de notícias na página *web* do campus Hortolândia.

Finalizados esses estudos, foi iniciado o desenvolvimento dos códigos junto a exploração de códigos e documentações para o auxílio da implementação, utilizando a metodologia incremental para realizar o desenvolvimento.

5. Desenvolvimento

5.1. Pesquisa de Aplicativos

Durante a pesquisa sobre qual aplicativo de mensagens seria utilizado para divulgação, encontrou-se uma variedade no mercado. Por este motivo foi realizado uma análise dos aplicativos mais usados atualmente, dentre eles: WhatsApp, Messenger, Telegram, Direct (do Instagram).

Destes aplicativos, o Messenger e o Direct estão mais atrelados a uma rede social, sendo mais simples e eficiente criar um grupo ou perfil na rede e divulgar como publicações em suas redes atreladas, Facebook e Instagram, retirando-os das opções para o desenvolvimento deste trabalho.

5.1.1. WhatsApp *versus* Telegram

De acordo com uma das pesquisas apresentadas no Panorama *Mobile Time/Opinion Box*, fruto de uma relação entre o site de notícias *Mobile Time* e a empresa de pesquisas *on-line Opinion Box*, o WhatsApp se encontra em primeiro lugar de popularidade. Em uma pesquisa de janeiro de 2022, 99% possuíam o WhatsApp instalado apresentando ser uma excelente escolha no quesito de divulgação, já o Telegram 60% [MobileTime and OpinionBox 2022]. Na Tabela 1 é apresentada uma comparação entre os aplicativos WhatsApp e Telegram. Nota-se que Whatsapp é mais popular, porém suas APIs são pagas e focadas no uso empresarial e existe uma limitação para o número de pessoas no grupo. O Telegram apesar de não ser tão popular, possui APIs oficiais e gratuitas e sem limitação no número de pessoas nos grupos.

Tabela 1. Comparação entre WhatsApp e Telegram

	WhatsApp	**Telegram**
Fama	+2 bilhões	+ 700 milhões
APIs	Possui, é paga, para empresas	Possui, é oficial e gratuita
Grupos	512* pessoas	200 mil/ilimitado

O Telegram apresenta estar presente cada vez mais nos *smartphones* dos brasileiros. Em um ano a proporção de *smartphones* nacionais com o Telegram instalado foi de 45% para 60% [MobileTime and OpinionBox 2022]. A causa disso vai pelo motivo de possuir algumas funcionalidades que os outros concorrentes não possuem, como os canais e grupos. De acordo com FAQ do site oficial do Telegram[5], esses grupos podem ter até 200 mil integrantes ou mais, com conteúdo controlado por alguns administradores. Uma função que permite que possa ser desenvolvido um sistema de poucos recursos já que é possível centralizar as requisições ao *bot* e reduzi-las drasticamente, isto supondo num cenário futuro.

Explorando formas e métodos para realizar essa automação no envio de mensagens por estes aplicativos, o WhatsApp apresentou ter uma versão oficial, mas direcionada à empresas e com custo de uso.

Há formas extraoficiais mas que não apresentam suporte e necessitam de utilizar o próprio usuário para o envio. Seus funcionamentos podem ser resumidos a uma sequência de ações que o programa irá realizar no navegador, apresentando um aspecto de baixa performance caso se enviada individualmente mensagens aos usuários. Para minimizar esse efeito pode-se criar grupos para centralizar os envios, em contrapartida esses grupos possuem uma capacidade limitada, podendo criar outros grupos a medida que vão enchendo.

Entretanto, esses métodos extraoficiais apresentam uma possibilidade do usuário ser bloqueado por estar utilizando essas automações. É possível diminuir os riscos reduzindo a performance, fazendo com que as interações sejam mais lentas para se aproximar ao um ser humano.

Com esses pontos levantados, o Telegram, mesmo não possuindo alta presença

[5]https://telegram.org/faq#q-how-is-telegram-different-from-whatsapp

nos *smartphones*, possui um método oficial e sem custos disponível no site do Telegram[6], possuindo até mesmo uma documentação bem clara sendo o escolhido como o meio de divulgação deste trabalho. Em um artigo da editora EXAME [EstadãoConteudo 2022], Patrícia Rossini, pesquisadora do departamento de Comunicação e Mídia da Universidade de Liverpool, listou como uma das principais particularidades do Telegram, as funcionalidades de código aberto que permitem a criação de automações, como a que será realizada neste trabalho, além da sua capacidade de disseminação de informações.

Pontuo que não foi realizado quaisquer levantamentos sobre o uso desses aplicativos entre os alunos e servidores.

5.2. Métodos de Desenvolvimento

Ao procurar por métodos para o desenvolvimento deste trabalho, diversas formas foram encontradas em tutoriais e fóruns pela *internet*, desde criação da *API*, como também algumas formas para o envio de mensagens pelo *bot* e as configurações de seus comandos, em Python e em PHP. Conteúdo esse encontrado desde blogs amadores à fóruns de ajuda, não tendo sites em específico. Além de que, tendo em vista a atualização das bibliotecas algumas informações nesses sites podem não funcionar mais.

Um dos métodos encontrados utilizava o Node.js, mais especificamente o *Node-RED*, realizando todo o procedimento por meio dele, desde a aquisição das informações de um site ao envio da mensagem pelo *bot*. Fazendo uso do MongoDB, software de banco de dados *NoSQL*, para o controle de repetições e atualizações, de uma forma bem simples. E com diversos exemplos de *flow* prontos que apresentam uma codificação visual do processo destes *nodes*.

Entretanto, um de seus Nodes, um *plugin* ou biblioteca, necessita de uma visão do site em XML, que o site do campus Hortolândia não possui. Mas ainda há outras formas de realizar a busca das notícias de um site qualquer utilizando o Node.js e assim, o sistema possa ser realizado pelo meio comentado.

Por outro lado, o Python apresentou similaridades com o Node.js também possuindo bibliotecas para realizar o *Web Scraping* e a *API*, mas para leigos nestas linguagens o Node.js apresenta estruturas mais complexas, podendo apresentar um grau maior de dificuldade no entendimento do código. Já o Python por conta de seus princípios é uma linguagem mais clara e agradável, o que auxilia o desenvolvedor que deseja utilizar-se do resultado deste trabalho para fazer atualizações ou modificações.

Portanto, com esses pontos levantados o Python foi escolhido para ser utilizado para o desenvolvimento das funções do *bot* para o Telegram, para criação da *API* e aplicação do *Web Scraping*.

5.2.1. Raspagem das Notícias do Site

Antes de seguir para a extração, é necessário definir a URL da qual as informações das notícias serão retiradas, da qual será realizado o processo de extração.

Na página inicial do site do IFSP Campus Hortolândia[7] há uma lista das últimas

[6]https://core.telegram.org/*API*

[7]https://hto.ifsp.edu.br/institucional/

dez notícias do mais recente ao mais antigo, conforme foi apresentado na Figura 1, mostrando o título e o link da notícia. Contudo, ao acessar a lista completa marcado em amarelo na Figura 1, o usuário é redirecionado a uma página em que é apresentando por padrão as mesmas dez últimas notícias com mais informações, como foi apresentado na Figura 2. Nesta página encontra-se as informações que serão agregadas ao que será enviado pelo *bot*, o título, descrição, data e hora enviada. Será visto posteriormente que há o nome do autor, que aparece somente no código fonte da página. Neste ponto do trabalho é recomendável que se tenha algum conhecimento básico em HTML para extrair exatamente a informação desejada.

Para o desenvolvimento desse método inicial para raspar os dados foi baseado nos diversos exemplos de uso de *Web Scraping* com Python que encontra-se em video tutoriais, blogs e fóruns. Na maioria dos resultados encontrados estavam entre eles palavras-chaves como: *Selenium*, *BeautifulSoup*, Pandas, Pyppeteer e *Requests*; todas bibliotecas com métodos e aplicações que podem auxiliar na extração de dados de um site. Dentre elas foi utilizado, neste projeto, somente o *BeautifulSoup* e o *Requests* para conseguir as informações das notícias do site do IFSP HTO.

Pode ser utilizado o *Selenium* no lugar do *Requests* para conseguir o código fonte do site, uma vez que os dois conseguem chegar no objetivo. Entretanto, o *Selenium* é mais recomendado para realizar requisições em páginas que tem um conteúdo dinâmico. Isto é, o conteúdo gerado além do conteúdo fixo da página, como por exemplo no Youtube, ao chegar ao fim da página, ele carrega mais vídeos aparentando não ter fim.

O *Selenium* simula a interação humana, portanto, ele irá abrir o navegador escolhido e permitirá interagir com o conteúdo do site, campo de pesquisa, caixa de seleção, entre outros. Mas nesse caso a biblioteca *Requests* é suficiente, já que página da onde as notícias serão retiradas possui um conteúdo estático, isto é, conteúdo que não será adicionado à página depois de finalizar o carregamento da mesma.

5.2.2. Análise da página de notícias

Ao analisar o código fonte da página de notícias. utilizando a opção "Inspecionar" do navegador, cujo o atalho no teclado é o F12, foram identificados os componentes/*tags* que compõem a página. Conforme a Figura 6, é possível ver à direita uma linha em azul, nela vemos que a *tag div* de *class="title-list-1"* contém todos as notícias da página, e dentro da lista tem seus itens, as notícias todas possuindo a mesma *tag* "*div*" e classe "titleItem".

Na Figura 7 observa-se o que está dentro das *tags div class="tileItem"*, as informações das notícias. Cada informação esta separada por outras *tags* podendo ser distinguidas pela classe por isso, mapear e guardar o nome das *tags* e o valor de seus "*class*" será muito importante para realizar extração da informação posteriormente. Com isso em mente, podemos identificar que o título e parte do *link* da página estão na classe "*tileHeadline*" e a descrição se encontra na *tag div* de classe "*description*".

Foi encontrado também o autor da notícia, algo que não é apresentado na visualização da página e, por fim, encontramos a data e hora em que o artigo foi postado. Estes últimos não possuem uma *tag* com classe própria o que dificulta extrair a informação explicitamente. Entretanto, será apresentada uma forma de obter essas informações.

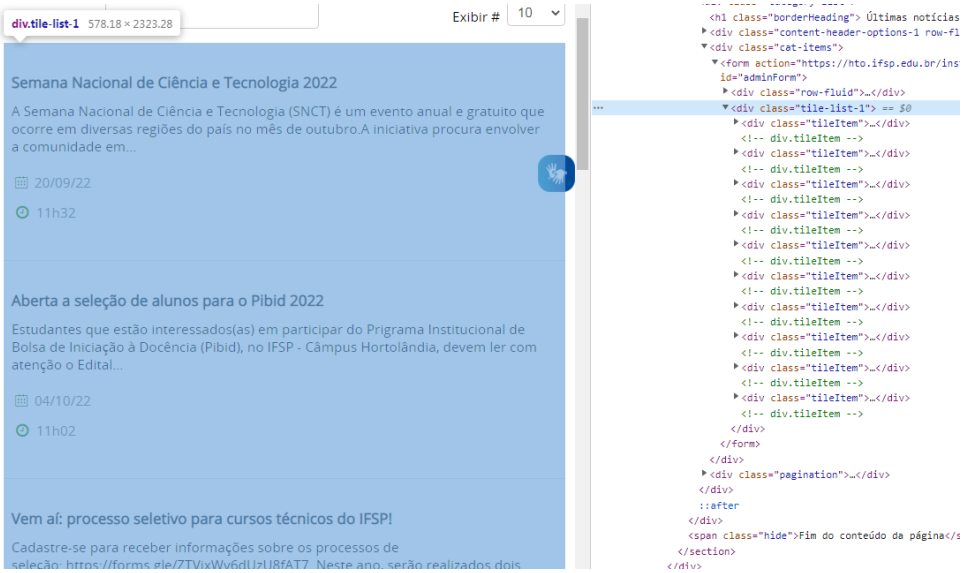

Figura 6. Código fonte do site apresentando a *tag* relacionada a lista de artigos apresentados.

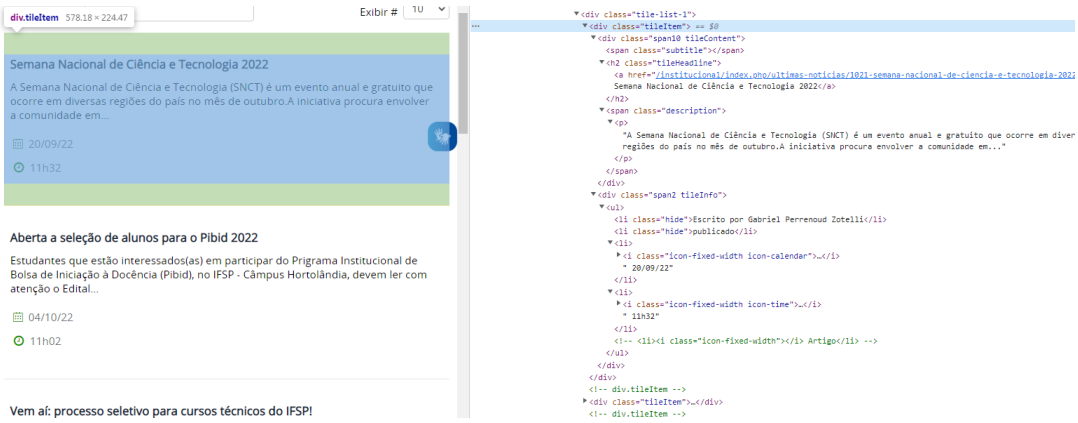

Figura 7. Código fonte do site apresentando a *tag* relacionada ao artigo apresentado.

5.2.3. Arquitetura

A arquitetura foi desenvolvida da seguinte forma. Da página alvo será utilizada uma *API* que terá a lógica de extrair as notícias da página e disponibilizá-las por *endpoints*, ou rotas. Para então, serem consumidas pelo programa consumidor que enviará uma requisição a um endereço correspondente a uma funcionalidade da API do Telegram. Que por sua vez, enviará a mensagem para o Telegram por meio de um *bot*, conforme na Figura 8.

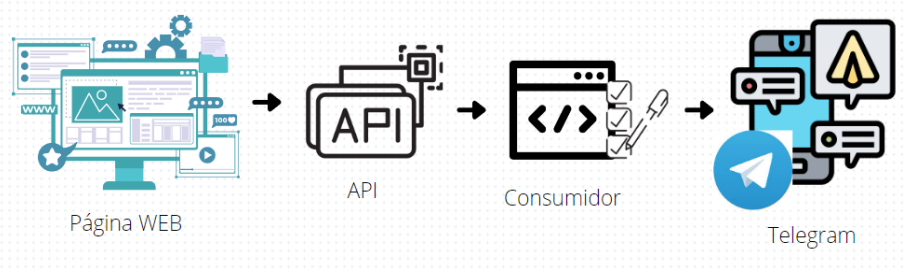

Figura 8. Arquitetura do projeto.

5.2.4. *API* v1

Com a identificação das *tags* e suas classes contendo as informações desejadas podemos criar uma implementação básica do *Web Scraping* no Python.

Conforme o código apresentado na Figura 9, é utilizado nas linhas 3 e 4, a biblioteca *requests*, para fazer a requisição do endereço da página das notícias (linha 6). A biblioteca bs4 (*BeautifulSoup*), utilizada para extrair os dados de arquivos HTML e XML, o que irá transformar o conteúdo dessa página, adquirido com o *requests*, para um objeto mapeado (linha 8), permitindo realizar buscas no conteúdo da página.

```
1  # -*- coding: utf-8 -*-
2
3  import requests
4  from bs4 import BeautifulSoup
5
6  response = requests.get("https://hto.ifsp.edu.br/institucional/index.php/ultimas-noticias")
7
8  site = BeautifulSoup(response.content, 'html.parser')
9
10 # HTML da notícia
11 noticia = site.find('div', attrs={'class', 'tileItem'})
12
13 # Título da notícia
14 titulo = noticia.find('a')
15 print("Titulo: " + titulo.text)
16 print("URL: " + "https://hto.ifsp.edu.br" + titulo.get('href'))
17
18 # Descrição da notícia
19 descricao = noticia.find('span', attrs={'class', 'description'})
20 print("Descrição: " + descricao.text)
21
22 # Data da notícia
23 data = noticia.findAll('li')[-2]
24 print("Data: " + data.text)
25
26 # Hora da notícia
27 hora = noticia.findAll('li')[-1]
28 print("Hora: " + hora.text)
29
```

Figura 9. Código fonte de uma implementação básica de *Web Scraping* na página de notícias, pegando todas as notícias da página.

Na linha 11, utilizando o método *findAll()*, procura-se todas as *div* cujo o atributo seja da "*class titleItem*", que é onde fica localizado o trecho na página que contém a notícia e, observando o código fonte da página. Com a extração dessa lista de *div* continua-se

extraindo as informações. Em seguida é feito um *for* para iterar com cada item da lista contida na variável "noticia"para pegar, em sequência, as informações de cada uma.

O título da notícia encontra-se numa *tag* <a> e, nesse caso, como é única dentro do *titleItem* é possível pegá-la somente passando o nome da *tag*, como é visto na linha 14. Para obter o título usa-se "*.text*" (Linha 15), pois é o conteúdo da *tag* <a> e esse "*.text*" é uma forma de obter o atributo de um objeto no Python. Para obter a parte do endereço do artigo é necessário obter o atributo da *tag* <a>, o href, usando "*.get('href')*". Juntando o endereço usado na linha 6 e do href é adquirido um *link* direto para o artigo. Nessa e em outras informações utilizou-se o método "*find()*", que pega a primeira ocorrência da *tag* passada e seus atributos.

A descrição da notícia encontra-se na *tag* <*span*> com "*class description*" (Linha 19). Da mesma maneira que o título usa-se "*.text*" para obter o conteúdo da *tag* <*span*>. Para a extração da data e hora da notícia, foi utilizado o "*findAll()*" (Linhas 23 e 27), para obter uma lista de todas as ocorrências das *tags* e, dessa lista foi passado um índice negativo para ordenar do final para o início.

Dessa forma, na hora foi passado o índice "[-1]" pois a informação se encontra na última ocorrência da *tag* . O mesmo foi feito em relação a data, porém a data se encontra na penúltima ocorrência ou posição na lista, portanto passando "[-2]", e para obter o conteúdo "*.text*". Com isso extraiu-se os dados da primeira notícia, a mais recente, que é apresentada no site das "Últimas Notícias".

5.2.5. *API* v2

Para obter essas notícias e então apresentá-las no Telegram foram feitas algumas alterações no código apresentado na Figura 9. Essas alterações tiveram como objetivo transformar a extração num *endpoint* que poderá ser utilizado para trabalhos futuros.

Na Figura 10 é exibido um trecho depois dos ajustes feitos. Primeiramente foi necessário importar a biblioteca *Flask* (Linha 6), um *microframework* destinado principalmente a pequenas aplicações com requisitos simples, como por exemplo, a criação de um *site* básico ou, nesse caso, uma *API* básica de notícias.

Na sequência foi inciada uma instância do *Flask* (Linha 17). A mesma recebe por parâmetro a variável "__nome__" (interna no Python) cujo objetivo é avaliar o nome do módulo atual. Após a instância definiu-se uma rota URL com *route* (Linha 20) a qual irá acionar a função descrita abaixo.

O nome da função é irrelevante para o funcionamento, por exemplo foi colocado *homepage*, mas poderia ser *index* ou algo que remeta o endereço raiz que é o "/". Foi criado um método para possibilitar colocar informações ou instruções ao acessar o endereço raiz, neste caso ele só escreve "*on*" na tela.

Também foi extraído para a função getSite(), a requisição da página (Linha 27) e sua conversão para um objeto *BeaultifulSoap* (Linha 28) para não duplicar código.

Para definição de uma rota pode ser utilizado o ".route()" ou o ".get()". O ".get()" é um endereço que utiliza-se do método HTTP GET, sendo assim, o *Flask* adiciona al-

```python
# -*- coding: utf-8 -*-

import requests
import json
from bs4 import BeautifulSoup
from flask import Flask


class NoticiaModel:
    def __init__(self, titulo, link, descricao, dataHora):
        self.titulo = titulo
        self.link = link
        self.descricao = descricao
        self.dataHora = dataHora


app = Flask(__name__)


@app.route('/')
def homepage():
    return 'on'


def getSite():
    response = requests.get(
        "https://hto.ifsp.edu.br/institucional/index.php/ultimas-noticias")
    return BeautifulSoup(response.content, 'html.parser')

```

Figura 10. Parte 1: Código fonte implementando uma *API* com o *microframework Flask*.

guns suportes. Na documentação oficial do *Flask*[8] é possível verificar que existe suporte para outros métodos HTTP, tais como POST, PUT, DELETE, entre outros. Salienta-se que neste projeto não houve a necessidade de aplicar filtros para o consumo de qualquer *endpoint* de notícias.

Na Figura 11 é exibido o restante do código contendo o *endpoint* para retornar as notícias. Das linhas 31 a 61, observa-se o código já descrito, implementando o *Web Scraping* e pegando todas as notícias da página de Últimas Notícias.

A fim de retornar essas notícias num formato JSON, formato padrão nesse tipo de sistema, implementou-se a classe NoticiaModel (Linhas 8 a 21 na Figura 10) para adicionar as informações extraídas em um objeto no Python. Com isso foi instanciada uma lista (Linha 37) antes do "for" para que a cada iteração fosse adicionado dentro da lista um objeto da classe NoticiaModel (Linhas 54 a 58). Além disso, antes de adicionar o objeto à lista, o objeto era convertido em um dicionário, um formato chave-valor (JSON). Para converter o objeto usou-se uma espécie de atalho no Python, o ".__dict__" (Linha 58).

Para finalizar o retorno das notícias, importou-se a biblioteca "json" (Linha 4 na Figura 10). Essa biblioteca permite codificar e decodificar o formato JSON. Utilizou-se o método ".dumbs()" (Linha 58 na Figura 11), para converter a lista criada em um formato JSON e, incluiu-se um parâmetro adicional do dumbs, o "ensure_ascii=False" (Linha 58), para que os acentos contidos no site não fossem convertidos em código ASCII.

[8]https://*Flask*.palletsprojects.com/en/2.2.x/

```
30
31  @app.get('/noticias')
32  def get_noticias():
33      site = getSite()
34
35      listaNoticias = []
36
37      # HTML da notícia
38      noticias = site.findAll('div', attrs={'class', 'tileItem'})
39      for noticia in noticias:
40          # Título da notícia
41          titulo = noticia.find('a')
42
43          # Descrição da notícia
44          descricao = noticia.find('span', attrs={'class', 'description'})
45
46          # Data da notícia
47          data = noticia.findAll('li')[-2]
48
49          # Hora da notícia
50          hora = noticia.findAll('li')[-1]
51
52          listaNoticias.append(
53              NoticiaModel(titulo.text,
54                           "https://hto.ifsp.edu.br" + titulo.get('href'),
55                           descricao.text, data.text + "" + hora.text).__dict__)
56
57      #ensure_ascii=False para impedir que troque por caracteres por \u00f8 (ASCII code)
58      return json.dumps(listaNoticias, ensure_ascii=False)
59
60
61  @app.get('/ultima_atualizacao')
62  def get_ultima_atualizacao():
63      site = getSite()
64
65      ultima_atualizacao = site.find('span', attrs={'class', 'documentModified'})
66
67      return json.loads('{ "ultima_atualizacao": "' + ultima_atualizacao.text + '" }')
68
69
70  app.run(host='0.0.0.0', port=7000)
71
```

Figura 11. Parte 2: Código fonte implementando uma *API* com o *microframework Flask*.

Durante o desenvolvimento do projeto observou-se a necessidade de extrair a informação relacionada a data da última atualização da notícia. Essa informação é utilizada para identificar se houve alguma alteração na página. Assim como foi feito para as notícias, definiu-se uma rota e método para requisitar essa informação localizados entre as linhas 61 a 67. A informação estava contida numa *tag* <*span*> de *class documentModified* e, para retornar essa informação foi utilizado outro método da biblioteca json, *loads*, que irá converter o texto passado num formato json.

Por fim, é iniciada a instância do *Flask* utilizando o ".run()" passando como parâmetros o endereço e porta selecionados para executar o programa. Utilizou-se a ferramenta Replit, a qual disponibiliza um *link* com base no nome do projeto e no nome do usuário podendo ser acessado por qualquer um. Existe um limite de recursos no seu uso porém a ferramenta é gratuita. Após finalizada essa etapa, o próximo passo foi a criação do *bot* no Telegram e a implementação do código do *bot*.

5.2.6. *bot* no Telegram

Como primeiro passo para realizar a integração com o aplicativo de mensageria, cria-se um *bot* no próprio Telegram. O Telegram tem um método oficial para criação e gerenciamento dos seus *bots* criados para o aplicativo, o BotFather.

Esse *bot* pode ser acessado ao pesquisá-lo pelo ícone de busca encontrado na parte superior da tela inicial. Ao entrar no chat inicia-se a conversa com "/start". Em seguida é

apresentada uma lista de ações podendo clicar no comando da lista ou digitar para iniciar a ação.

A ação para criar o *bot* é "/newbot". Será solicitado que insira um nome para o *bot* duas vezes, sendo a segunda terminada com "bot" ou "_bot". Após inserir o que é solicitado, será apresentado o *link* de acesso ao chat do seu *bot* e, o mais importante, o *token* para acessar a *API* do *bot*. Essa chave deve ser mantida em segurança e não deve ser compartilhada, através dela pode-se receber e enviar mensagens para os usuários que interagem com o *bot*. O procedimento descrito pode ser observado na Figura 12.

Figura 12. Imagem do procedimento de criação de um *bot* no Telegram pelo *BotFather*.

5.2.7. Consumidor

Nessa seção será descrito o programa que irá consumir a *API* de notícias do campus descrita nos tópicos anteriores e as enviará para um canal do Telegram. O programa em questão pode ser visualizado nas Figuras 13 (parte 1) e 14 (parte 2). Na Figura 13, linhas 3 e 4, tem-se as importações dos módulos *requests* e *schedule* para o programa. O módulo *requests* permite que o programa faça solicitações HTTP e obtenha dados da web, no caso irá obter da *API* desenvolvida deste trabalho, enquanto o módulo *schedule* permite que o programa agende a execução de funções, que permitirá com que tenha-se uma execução periódica desse programa.

Da linha 6 a 17, há dois blocos *"try/except"* que tentam abrir os arquivos "ultima_atualizacao.txt" e "titulo_ultima_noticia.txt", respectivamente. Se esses arquivos não existirem, eles serão criados usando o modo "x" (que cria um novo arquivo para escrita) por meio da função "*open*". Esses arquivos são usados para que, caso o programa

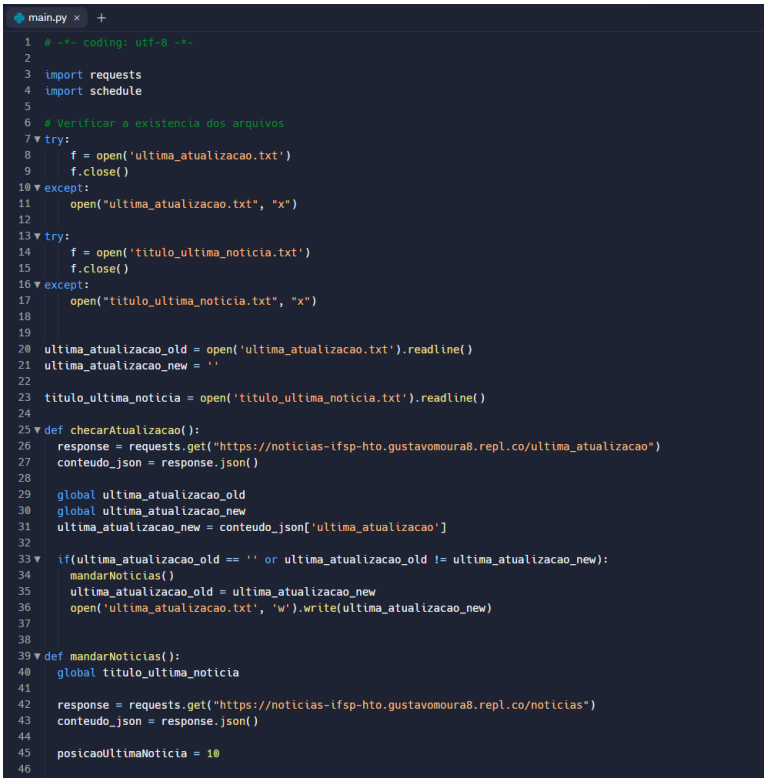

Figura 13. Parte 1: Código fonte do programa consumidor.

pare por algum motivo, ele consiga identificar qual foi a última notícia a ser enviada e não enviará novamente as notícias anteriores.

Em sequência, são declaradas algumas variáveis de controle (Linhas 20 a 23). Nessas linhas é lido o conteúdo da primeira linha dos arquivos “ultima_atualizacao.txt” e “titulo_ultima_noticia.txt” e armazenam o resultado em variáveis correspondentes. A variável “ultima_atualizacao_old” é inicializada com o valor lido de “ultima_atualizacao.txt” e a variável “ultima_atualizacao_new” é inicializada com uma *string* vazia.

Seguindo o código, foi criado a função “checarAtualizacao()”, essa função faz uma solicitação HTTP para obter a data e hora da última atualização da *API* e compara a data e hora obtida com a data e hora armazenadas no arquivo “ultima_atualizacao.txt”. Se a data e hora forem diferentes, a função “mandarNoticias()” será chamada e o conteúdo da variável “ultima_atualizacao_new” será gravado no arquivo “ultima_atualizacao.txt”.

Na linha 39 temos a função “mandarNoticias()”, responsável por buscar as notícias mais recentes e enviar as informações para o grupo do Telegram. A primeira linha da função usa o comando global para definir a variável “titulo_ultima_noticia” como global, ou seja, permitindo que a variável seja acessada dentro e fora da função. Na segunda linha, é feita uma requisição HTTP para a *API*, solicitando o *endpoint* “/noticias”, com o objetivo de obter o conteúdo da página em formato JSON.

Na linha 47 (Figura 14), é criado um laço for que percorre o conteúdo da página JSON. A variável “posicaoUltimaNoticia” é inicializada com o valor 10, lembrando que, 10 é a quantidade máxima de notícias que podem vir da *API*. A ideia é que, a partir desse

índice, identifique-se as notícias que já tenham sido enviadas ao grupo do Telegram.

```
47  for i in range(len(conteudo_json)):
48    noticia = conteudo_json[i]
49    if (noticia['titulo'] == titulo_ultima_noticia):
50      posicaoUltimaNoticia = i
51      break
52
53  for i in reversed(range(posicaoUltimaNoticia)):
54    noticia = conteudo_json[i]
55
56    mensagem = '<b><a href="'+noticia['link']+'">' + noticia['titulo'] + '</a></b>\n'
57    mensagem += noticia['descricao'] + "\n\n"
58    mensagem += "Publicado " + noticia['dataHora'] + "\n\n"
59
60    url = "https://api.telegram.org/bot5793339883%3AAAGzMbtI9obUxID6TbuZBYYdDvB17gRiJ4k/sendMessage"
61
62    payload = {
63        "text": ""+ mensagem +"",
64        "parse_mode": "html",
65        "disable_web_page_preview": False,
66        "disable_notification": False,
67        "reply_to_message_id": None,
68        "chat_id": "-1001851441625"
69    }
70    headers = {
71        "accept": "application/json",
72        "User-Agent": "Noticias IFSP Telegram Bot",
73        "content-type": "application/json"
74    }
75
76    response = requests.post(url, json=payload, headers=headers)
77    print(response.text)
78
79  ultima_noticia = conteudo_json[0]
80  if (titulo_ultima_noticia == '' or titulo_ultima_noticia != ultima_noticia['titulo']):
81    titulo_ultima_noticia = ultima_noticia['titulo']
82    open('titulo_ultima_noticia.txt', 'w').write(ultima_noticia['titulo'])
83
84
85  schedule.every(1).minute.do(checarAtualizacao)
86
87  while 1:
88    schedule.run_pending()
89
```

Figura 14. Parte 2: Código fonte do programa consumidor.

Em seguida, é feita uma verificação das notícias já enviadas, comparando o título da última notícia do site com o valor da variável “titulo_ultima_noticia”, que representa o título da última notícia enviada ao grupo do Telegram. Se o título da notícia atual for igual ao título da última notícia enviada, a variável “posicaoUltimaNoticia” recebe o índice da notícia atual e o laço é interrompido.

A partir da linha 53, começa outro laço for, que percorre as notícias mais recentes a partir da posição “posicaoUltimaNoticia”. Para cada notícia, é criada uma mensagem com título, descrição e data de publicação, no formato HTML. É possível utilizar de algumas *tags* do HTML para personalizar o texto, no código utilizei da *tag* “<*strong*>” para deixar em negrito o título da notícia ao enviar ao Telegram.

Na linha 60 é definida a URL da *API* do Telegram para envio da mensagem, com o *token* do *bot*, criado na Seção 5.2.6, e o ID do grupo de destino. Em seguida, é criado um dicionário *payload* com os dados da mensagem, como texto, modo de *parse*, pré-visualização de páginas desabilitada e notificação desabilitada.

Na linha 76, é feita uma requisição HTTP com o método POST, utilizando a URL da *API*, o dicionário *payload* como dados da mensagem e os cabeçalhos HTTP definidos no dicionário *headers*. O retorno da requisição é impresso na tela, para poder visualizar o envio da mensagem.

Por fim, na linha 80, é atualizada a variável “titulo_ultima_noticia” com o título da última notícia do site, caso a variável “titulo_ultima_noticia” esteja vazia ou o título

seja diferente da última notícia enviada. O título da última notícia é armazenado em um arquivo de texto chamado “titulo_ultima_noticia.txt”, sobrescrevendo todo o conteúdo do arquivo caso tenha algo.

Na sequência, há um loop infinito que executa o método “*run_pending*()” do módulo *schedule*, que faz a verificação do agendamento de execução da função “checarAtualizacao()” de 1 em 1 minuto. Como apresentado, essa função é responsável por verificar periodicamente a existência de novas notícias e chamar a função “mandarNoticias()” para enviar as novas informações ao grupo do Telegram.

6. Conclusão

Este trabalho apresentou o desenvolvimento de uma automação de notícias em que sua divulgação se dá por meio de um aplicativo de mensagens, o Telegram. Através dessa automação é possível se obter uma outra forma de divulgação que ganha mais atenção com o sistema de notificação que os aplicativo de mensageria realizam no geral. No período de testes final, em que os 2 programas ficaram rodando por quase 2 dias, teve um total de 4 notícias e as mesmas foram enviadas dentro do intervalo de 1 minuto, apresentando uma resposta rápida. Na Figura 15, é possível ver o resultado final da notícia enviada no Telegram.

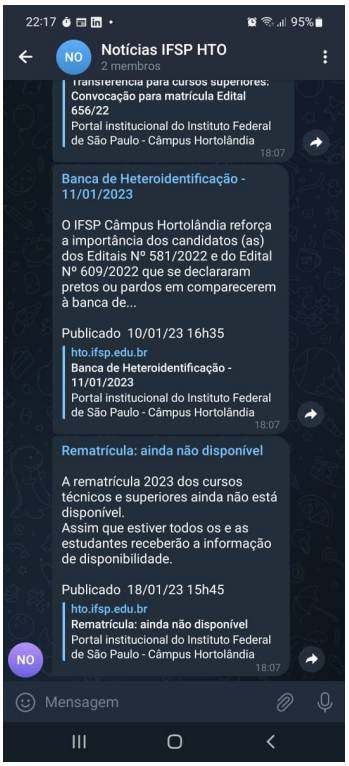

Figura 15. Imagem da visão do canal g1 retirado da versão mobile do Telegram.

O trabalho apresenta limitações, tais como: só será extraído do site as últimas 10 notícias que são apresentadas por padrão, entretanto, esse ponto pode se apresentar inválido visto que, a medida que o tempo passa e as notícias vão sendo enviadas, vai se formando um histórico de mensagens no próprio Telegram. Mas ainda não seria possível

acessar as notícias anteriores do momento em que o sistema for iniciado ou um fila de mais de 10 artigos que forma publicados de uma vez no sistema.

Este trabalho foi desenvolvido para que as notícias sejam enviadas para um único canal (que foi escolhido), onde os interessados teriam que ingressar para que recebam as notícias e sejam notificados, portanto é obrigatória a inscrição no canal escolhido. E por fim, obviamente, a automação criada depende da disponibilidade do site da qual extrai as notícias.

Além do estudo da linguagem da qual não tinha prática, todos os conhecimentos desenvolvidos durante o curso de Análise e Desenvolvimento de Sistemas foram exercitados a durante o desenvolvimento, enfatizando conteúdos de lógica de programação, linguagem de programação e estrutura de dados.

7. Trabalhos Futuros

Esta seção apresenta sugestões de trabalhos futuros relacionas a melhorias, expansão e alternativas de uso, algumas sugestões:

Realizar um levantamento sobre o uso dos aplicativo de mensageria no campus entre os alunos e servidores.

Este trabalho foi projetado com as especificidades do site IFSP HTO, ainda assim, este sistema pode servir como um modelo base para serem utilizados em sites de outros campi, tendo que realizar ajustes de acordo com as particularidades dos códigos fontes das páginas de outros campi.

Aprofundar na biblioteca do Telegram, a fim de permitir o uso individual do *bot*. Neste trabalho foi necessário somente enviar a notícia, mas por meio da biblioteca do Telegram no Python é possível receber mensagens que contém o código do remetente e, dessas mensagens pode-se definir comandos para realizar determinada ação, abrindo um leque de possibilidades.

A *API* foi criada utilizando a biblioteca *Requests* que pega o conteúdo estático da página, fazendo com que possa extrair no máximo as 10 notícias que a página tem por padrão, podendo ser resolvido ao se utilizar outras bibliotecas, como o *Selenium*, para que possa interagir com o campo que determina a quantidade de notícias mostrada na página, além de possibitar o uso da barra de busca da página.

E não precisa se limitar ao objetivo de enviar notícias, pode se mapear e extrair informações de diversas páginas permitindo acesso rápido a certos conteúdos. Pode-se, por exemplo, criar um *endpoint* que retorna *links* de trabalhos de conclusão por ano, ou selecionando um determinado ano. Também pode-se utilizar somente a parte de extração de dados do sistema para alimentar um outro sistema, ou aplicativo mobile, e até usar como base para outras finalidades acadêmicas e pessoais.

Referências

AgenciaBrasil (2020). Celular é o principal meio de acesso à internet no país. `https://agenciabrasil.ebc.com.br/economia/noticia/2020-04/celular-e-o-principal-meio-de-acesso-internet-no-pais`. Acesso em: 2022-12-07.

Aydin, O. (2018). Introduction to web scraping. In *R Web Scraping Quick Start Guide*. Packt Publishing.

DiárioPopular (2021). Aplicativos de mensagens: um mercado que não para de crescer. https://www.diariopopular.com.br/tecnologia/aplicativos-de-mensagens-um-mercado-que-nao-para-de-crescer-164471/. Acesso em: 2022-06-02.

Enberg, J. (2021). Global mobile messaging forecast 2021. https://www.emarketer.com/content/global-mobile-messaging-forecast-2021#page-report. Acesso em: 2022-06-02.

EstadãoConteudo (2022). Telegram vs whatsapp: qual o melhor app de mensagens. https://exame.com/tecnologia/telegram-vs-whatsapp-qual-o-melhor-app-de-mensagens/. Acesso em: 2023-01-26.

MobileTime and OpinionBox (2022). Mensageria no brasil fevereiro de 2022. https://www.mobiletime.com.br/pesquisas/mensageria-no-brasil-fevereiro-de-2022/. Acesso em: 2022-07-14.

RedHat (2020). Api rest. https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 2022-06-02.

Documento Digitalizado Público

Anexo I - Entrega do artigo final

Assunto: Anexo I - Entrega do artigo final
Assinado por: Daniela Marques
Tipo do Documento: Projeto
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Daniela Marques**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 28/03/2023 21:40:34.

Este documento foi armazenado no SUAP em 28/03/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1281481
Código de Autenticação: 06978f281e