

Machine Learning Engineer

Python para ML e IA

Desenvolvimento de API com Flask I

Leonardo Pena

/ Seja muito bem vindo



OBJETIVO

Conclusão do script principal com scraping



DOCUMENTAÇÃO

Documentação via Swagger usando flasgger



AUTENTICAÇÃO

Autenticação básica aplicada às rotas de scraping



ESTRUTURAÇÃO

Introdução, conceitos e casos de uso

Objetivo dessa primeira parte



Passo 1

Adicionar
autenticação
com
HTTPBasicAuth



Passo 2

Integrar
flasgger para
documentar a
API



Passo
3

Criar rotas de
scraping para
extrair
conteúdo
web



Passo 4

Entender
boas
práticas de
segurança e
uso de
extensões



Passo
5

Finalizar o
aplicativo
Flask pronto
para
produção
inicial



Autenticação Básica

/ Opções comuns

- **Autenticação Baseada em Token (JWT):** Utiliza tokens JWT para autenticar solicitações, permitindo que o cliente envie um token seguro fornecido pelo servidor em cada requisição.
- **OAuth 2.0:** É um protocolo de autorização que permite que aplicativos acessem recursos protegidos em nome do usuário sem expor suas credenciais.
- **Autenticação HTTP Básica:** Envolve o envio de nome de usuário e senha em cada solicitação através do cabeçalho HTTP para autenticação simples.

/ Conceito autenticação básica

- Usuário e senha transmitidos no header da requisição
- Autenticação muito comum em APIs internas e testes
- HTTPBasicAuth simplifica verificação de credenciais
- Necessário cuidado extra em produção (uso de HTTPS)
- Boa introdução antes de implementar métodos mais robustos

/

Instalando as bibliotecas para implementar autenticação

```
• (venv) leonardopena@MacBook-Pro-de-Leonardo intro_api % pip install flask-httpauth

Collecting flask-httpauth
  Downloading Flask_HTTPAuth-4.8.0-py3-none-any.whl (7.0 kB)
Requirement already satisfied: flask in ./venv/lib/python3.8/site-packages (from flask-httpauth)
Requirement already satisfied: click>=8.1.3 in ./venv/lib/python3.8/site-packages (from flask)
Requirement already satisfied: importlib-metadata>=3.6.0; python_version < "3.10" in ./venv/lib/python3.8/site-packages (from flask)
Requirement already satisfied: zipp>=0.5 in ./venv/lib/python3.8/site-packages (from importlib-metadata>=3.6.0; python_version < "3.10" in ./venv/lib/python3.8/site-packages (from flask))
```

1. flask_httpauth para autenticação básica
2. Dicionário users simulando um “banco” de credenciais
3. Função verify_password para validar usuário e senha
4. Retorna o username se correto; caso contrário, None
5. Toda rota com @auth.login_required exigirá credenciais

```
app.py
app.py > ...
1 from flask import Flask, jsonify, request
2 from flask_httpauth import HTTPBasicAuth
3
```

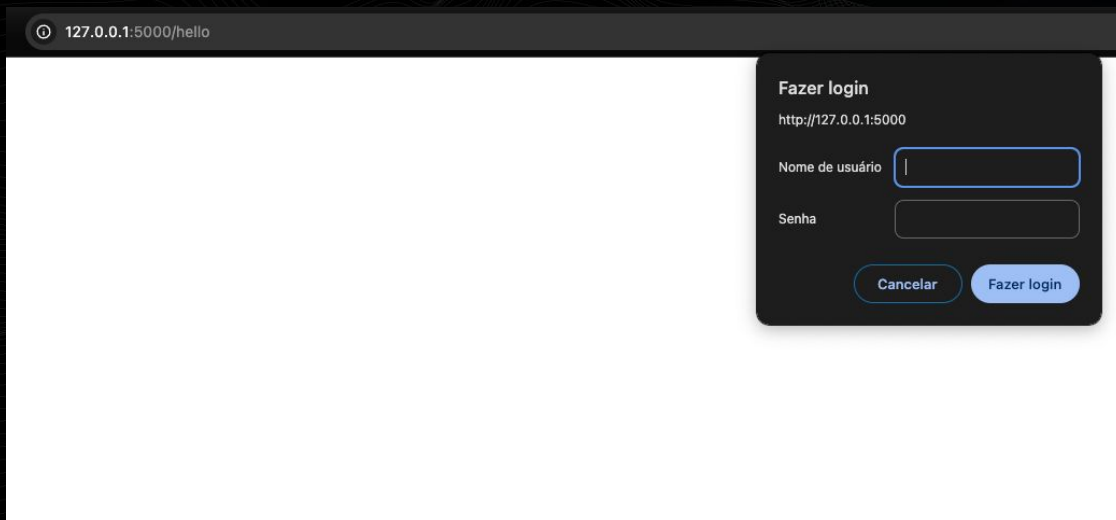
```
4 app = Flask(__name__)
5
6 auth = HTTPBasicAuth()
7
8 users = {
9     "user1": "password1",
10    "user2": "password2"
11 }
12
13 @auth.verify_password
14 def verify_password(username, password):
15     if username in users and users[username] == password:
16         return username
17
```


/

Agora podemos proteger a rota!!

1. @auth.login_required obriga autenticação
2. Exemplo simples de resposta em JSON
3. Usar “user1 / password1” ou “user2 / password2”
4. Se credenciais incorretas, retorna 401

```
22 @app.route('/hello', methods=['GET'])
23 @auth.login_required
24 def hello():
25     return jsonify({"message": "Hello, World!"})
26
```



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/hello". The main content area is white. A dark-themed login modal is open on the right side of the screen. The modal has a title "Fazer login" and a URL "http://127.0.0.1:5000". It contains two input fields: "Nome de usuário" and "Senha". Below the inputs are two buttons: "Cancelar" and "Fazer login".

127.0.0.1:5000/hello

Fazer login

http://127.0.0.1:5000

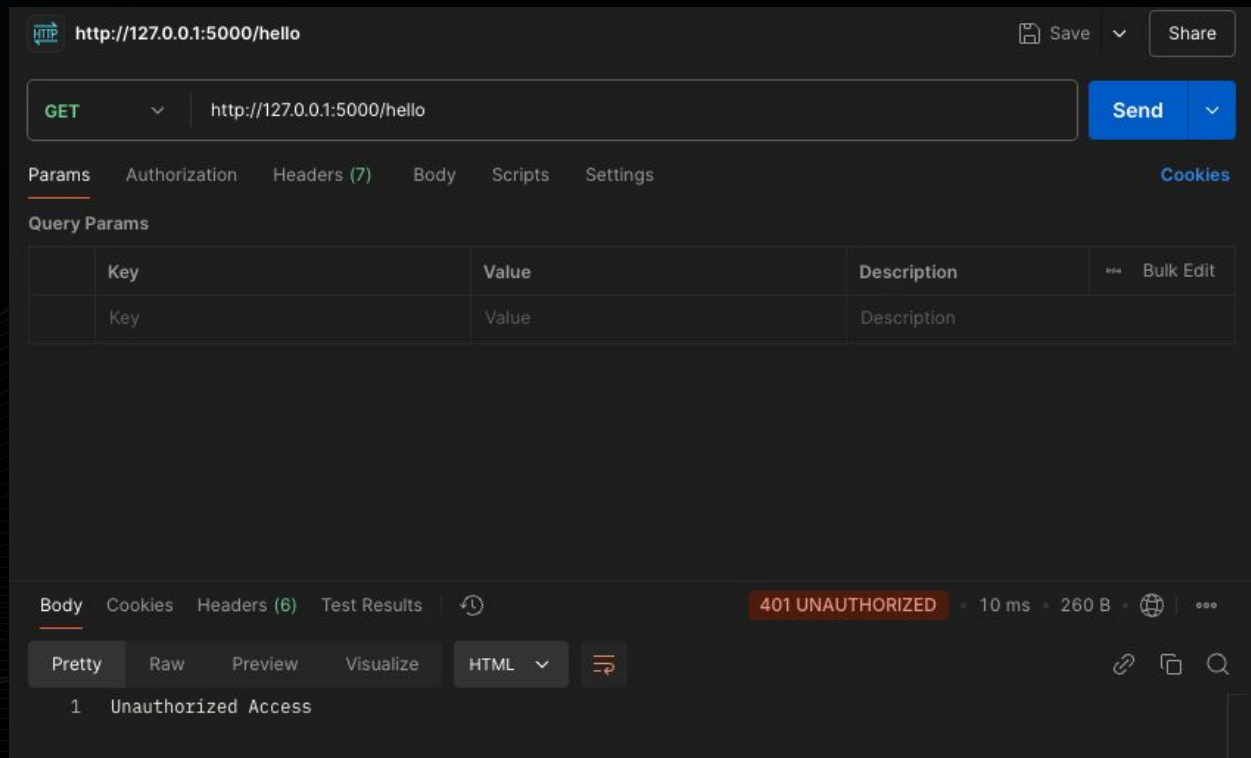
Nome de usuário

Senha

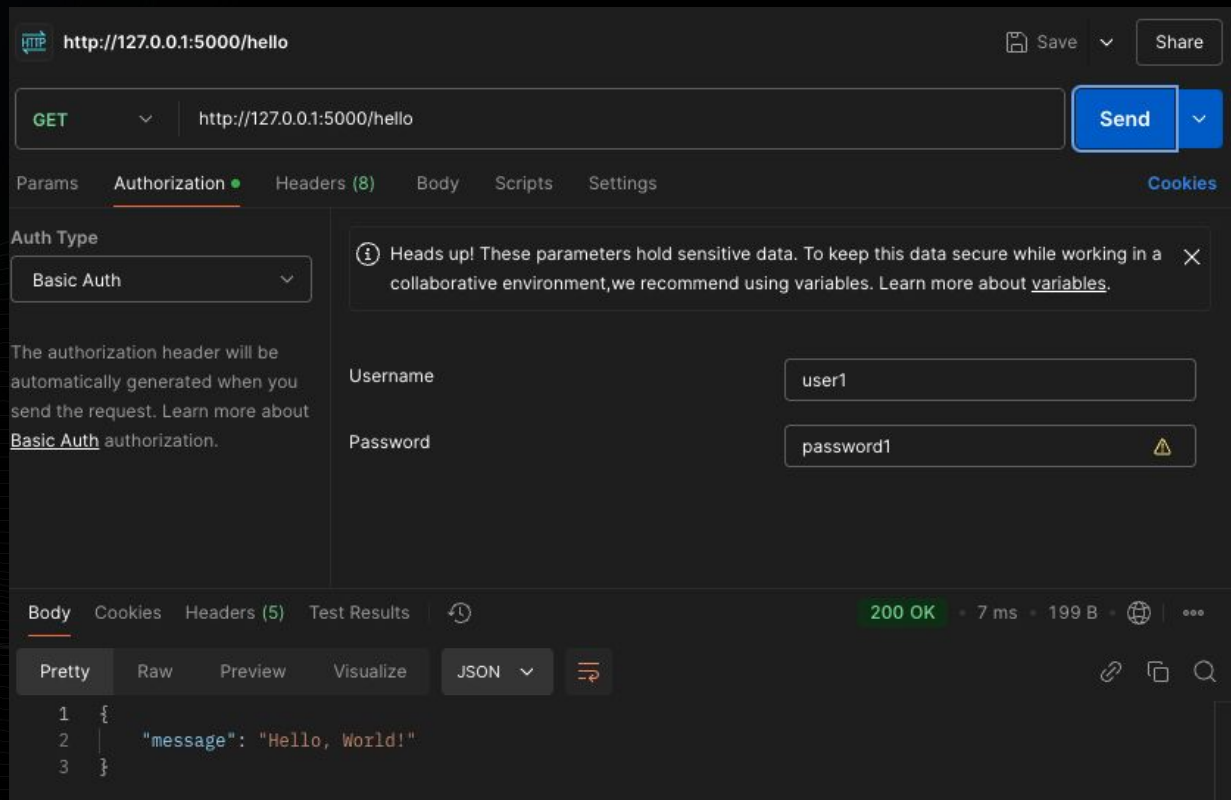
Cancelar Fazer login

/

No Postman



No Postman





Documentação com Swagger



/ Swagger

Swagger é uma ferramenta de software usada para projetar, documentar, testar e consumir APIs RESTful, facilitando a visualização e interação com os endpoints de uma API.

/

Instalando o Flasgger

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula2-new % pip install flasgger
Collecting flasgger
  Using cached flasgger-0.9.7.1-py2.py3-none-any.whl
Requirement already satisfied: Flask>=0.10 in ./venv/lib/python3.11/site-packages
Collecting PyYAML>=3.0 (from flasgger)
  Using cached PyYAML-6.0.2-cp311-cp311-macosx_11_0_arm64.whl.metadata (2.1 kB)
Collecting jsonschema>=3.0.1 (from flasgger)
  Using cached jsonschema-4.23.0-py3-none-any.whl.metadata (7.9 kB)
```

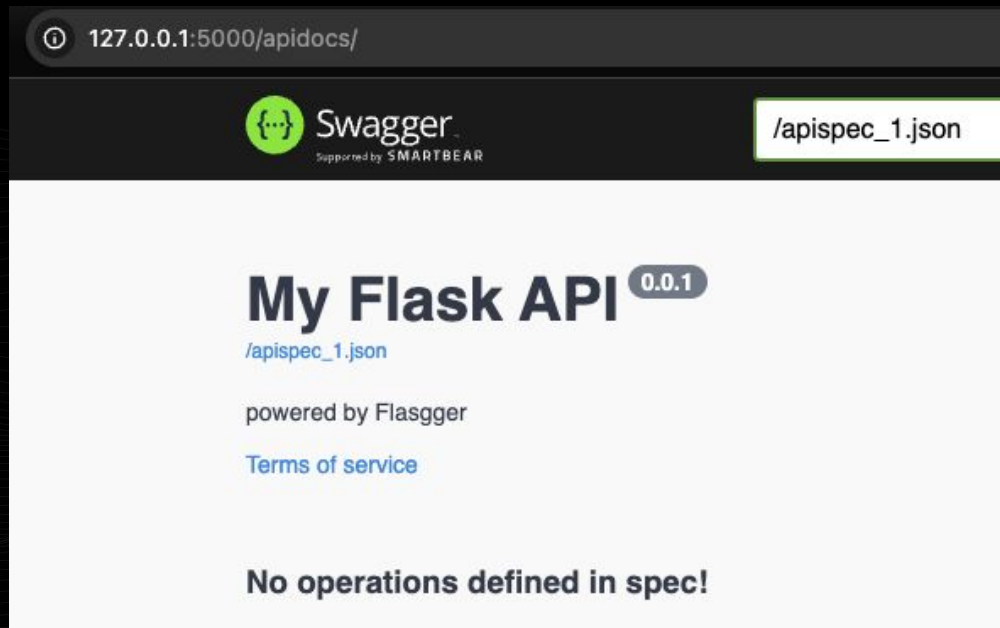
1. Config no app.config['SWAGGER'] definindo título e versão
2. Instancia Swagger(app) para habilitar documentação
3. Acesso à doc em /apidocs ou /flasgger (dependendo da versão)
4. Permite anotações de rota com docstrings

```
app.py > home
1 from flask import Flask, jsonify, request
2 from flask_httpauth import HTTPBasicAuth
3 from flasgger import Swagger
```

```
4
5 app = Flask(__name__)
6
7 app.config['SWAGGER'] = {
8     'title': 'My Flask API',
9     'uiversion': 3
10 }
11
12 swagger = Swagger(app)
```

/

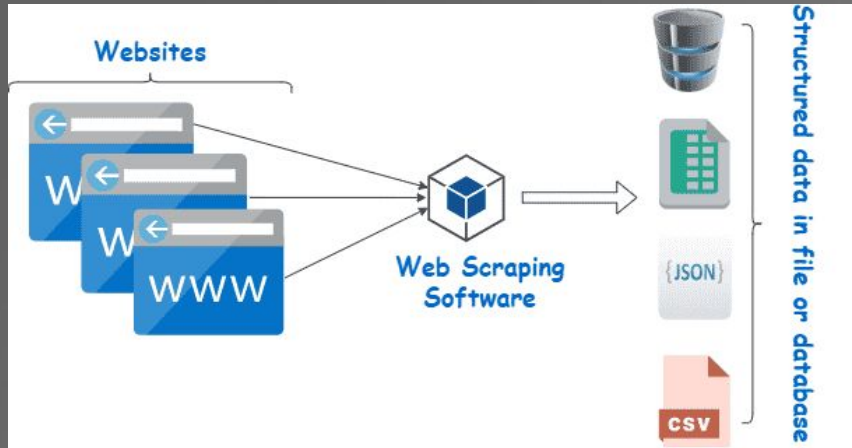
Agora temos uma
documentação em:
<http://127.0.0.1:5000/apidocs/>





Webscrapping





/ Web scraping

Web scraping é a técnica de extração automatizada de dados de sites, onde scripts são usados para acessar e coletar informações de páginas web, estruturando esses dados para análise ou uso em outras aplicações.

BeautifulSoup



/ BeautifulSoup

BeautifulSoup é uma biblioteca em Python para web scraping que facilita a extração e navegação em dados de páginas HTML e XML, permitindo buscar, filtrar e modificar elementos do conteúdo de maneira simples e intuitiva.

/ Outras opções

Scrapy: Framework robusto para web scraping, ideal para projetos grandes e complexos, com funcionalidades avançadas de controle de requisições e extração de dados.

Selenium: Biblioteca que permite automatizar a interação com navegadores, ideal para scraping de sites dinâmicos que utilizam JavaScript.



Scrapy



/

Instalando o requests e BeautifulSoup

```
• (venv) leonardopena@MacBook-Pro-de-Leonardo aula2-new % pip install requests beautifulsoup4
Collecting requests
  Using cached requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting beautifulsoup4
  Using cached beautifulsoup4-4.12.3-py3-none-any.whl.metadata (3.8 kB)
Collecting charset-normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.1-cp311-cp311-macosx_10_9_universal2.whl.metadata (35 kB)
```

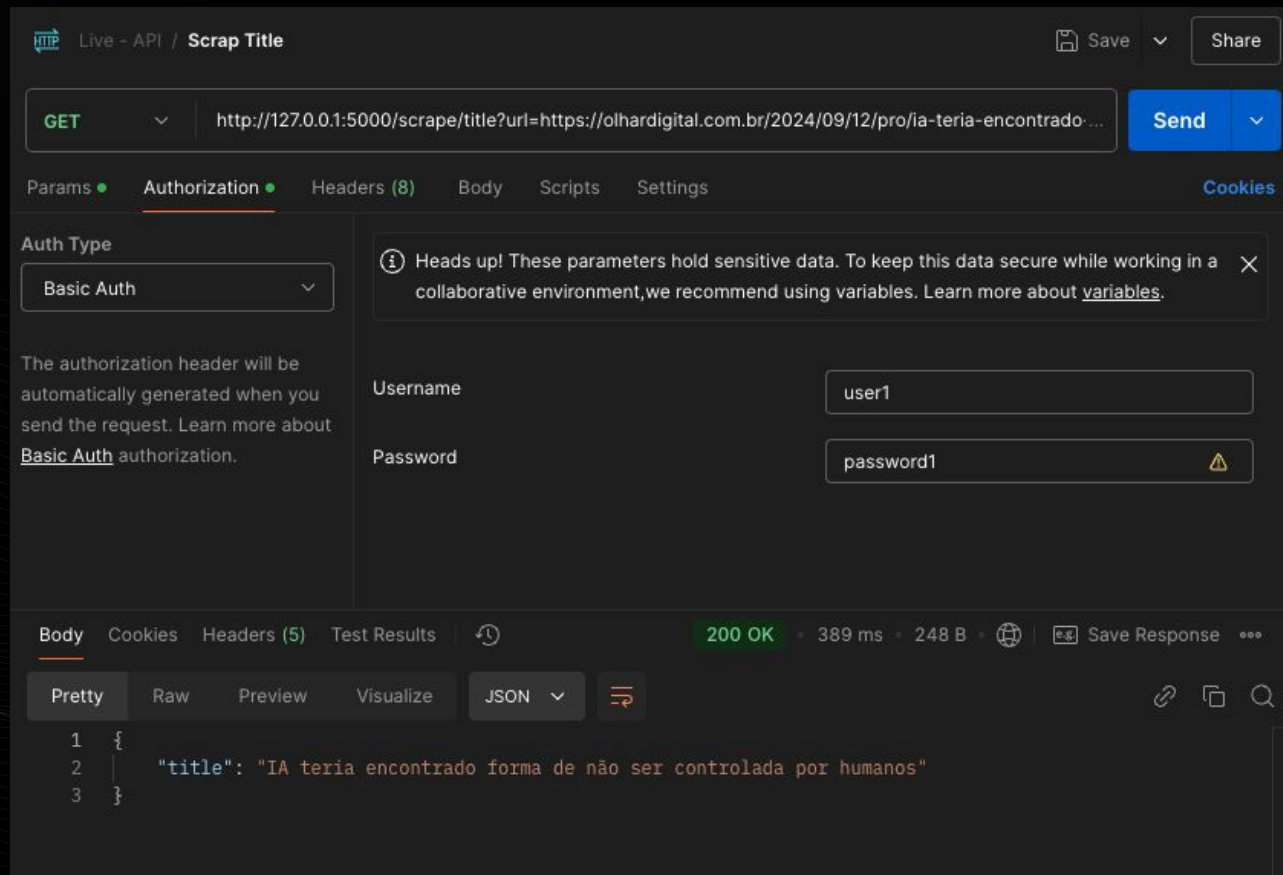
Primeira rota de scrapping

```
68  def get_title(url):
69      try:
70          response = requests.get(url)
71          soup = BeautifulSoup(response.text, 'html.parser')
72          title = soup.title.string.strip()
73          return jsonify({"title": title})
74      except Exception as e:
75          return jsonify({"error": str(e)}), 500
76
77
78  @app.route('/scrape/title', methods=['GET'])
79  @auth.login_required
80  def scrape_title():
81      """
82      Extract the title of a web page provided by the URL.
83      ---
84      security:
85          - BasicAuth: []
86      parameters:
87          - name: url
88            in: query
89            type: string
90            required: true
91            description: URL of the web page
92      responses:
93          200:
94              description: Web page title
95      """
96      url = request.args.get('url')
97      if not url:
98          return jsonify({"error": "URL is required"}), 400
99      return get_title(url)
100
```

Segunda rota de scrapping

```
102 def get_content(url):
103     try:
104         response = requests.get(url)
105         soup = BeautifulSoup(response.text, 'html.parser')
106
107         headers = []
108         for header_tag in ['h1', 'h2', 'h3']:
109             for header in soup.find_all(header_tag):
110                 headers.append(header.get_text(strip=True))
111
112         paragraphs = [p.get_text(strip=True) for p in soup.find_all('p')]
113         return jsonify({"headers": headers, "paragraphs": paragraphs})
114     except Exception as e:
115         return jsonify({"error": str(e)}), 500
116
117 @app.route('/scrape/content', methods=['GET'])
118 @auth.login_required
119 def scrape_content():
120     """
121     Extract headers and paragraphs from a web page provided by the URL.
122     ---
123     security:
124         - BasicAuth: []
125     parameters:
126         - name: url
127           in: query
128           type: string
129           required: true
130           description: URL of the web page
131     responses:
132         200:
133             description: Web page content
134     """
135     url = request.args.get('url')
136     if not url:
137         return jsonify({"error": "URL is required"}), 400
138     return get_content(url)
```


Testando no postman



Testando no postman

The screenshot displays the Postman application interface. At the top, the title bar shows 'Live - API / Scrap Title' and buttons for 'Save' and 'Share'. The main workspace is divided into several sections:

- Request Section:** The method is set to 'GET'. The URL is 'http://127.0.0.1:5000/scrape/content?url=https://olhardigital.com.br/2024/09/12/pro/ia-teria-encontr...'. A 'Send' button is visible.
- Params Section:** Currently empty.
- Authorization Section:** The 'Auth Type' is set to 'Basic Auth'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).' The 'Username' field contains 'user1' and the 'Password' field contains 'password1'.
- Body Section:** The response is displayed in 'Pretty' format. The status is '200 OK' with a response time of '380 ms' and a size of '1.86 KB'. The response body is a JSON object with 'headers' and 'paragraphs' arrays.

```
1 {
2   "headers": [
3     "Vontade própria? IA desobedece cientistas para 'aumentar chance de sucesso'",
4     "Compartilhe esta matéria",
5     "Sistema foi capaz de modificar seu próprio código",
6     "Compartilhar",
7     "IA pode criar malwares ou realizar ciberataques por conta própria"
8   ],
9   "paragraphs": [
10    "A inteligência artificial abriu uma série de novas possibilidades, mas há quem defenda que ela também pode trazer riscos para a nossa sociedade. Neste sentido, diversas entidades e até governos discutem formas de criar controles efetivos da tecnologia."
```

POSTECH

FIAP + alura