Análise de Repositórios GitHub por Linguagem de Programação

🌎 Introdução

Este projeto tem como objetivo criar um dataset rico e estruturado com informações dos repositórios mais relevantes do GitHub, organizados pelas 10 linguagens de programação mais populares em 2025. Com esses dados, podemos responder perguntas como:

- Qual linguagem tem os projetos mais estrelados?
- Há correlação entre o número de contribuidores e a atividade do repositório?
- Quais licenças são mais comuns em projetos open-source?
- Como a localização geográfica dos donos influencia a popularidade dos repositórios?

O dataset gerado pode ser usado para:

✅ Identificação de tendências no desenvolvimento de software

✅ Análise de comunidades open-source

✅ Tomada de decisões para contribuições ou adoção de tecnologias

```
# !pip install pandas
# !pip install requests
```

Imports

```
import requests
import pandas as pd
from datetime import datetime, timedelta
import time
import random
```

🔁 Rotação de Tokens

Para evitar limites da API, usamos múltiplos tokens com seleção aleatória:

```
TOKENS = [
            # "TOKENS"
    ]

def get_headers():
    return {"Authorization": f"token {random.choice(TOKENS)}"}
```

🔧 Configuração Inicial

Definir as linguagens para análise e parâmetros de paginação para extração eficiente dos dados.

```
LANGUAGES = ["Python", "JavaScript", "Java", "C#", "C++", "TypeScript", "Go", "Rust", "Kotlin", "Swift"]
PER_PAGE = 100
PAGES = 15
```

👤 Obter Informações do Dono do Repositório

- Acessa o perfil do dono via API
- Trata 3 informações principais: tipo, quantidade de repositórios e localização
- Implementa tratamento robusto de erros

```
def get_owner_info(owner_url):
    try:
        response = requests.get(owner_url, headers=get_headers())
        if response.status_code == 200:
            owner_data = response.json()
            return {
                "owner_type": owner_data.get("type", "User"),
                "owner_public_repos": owner_data.get("public_repos", 0),
```

```
                "owner_location": owner_data.get("location", None)
            }
    except Exception as e:
        print(f"Erro ao buscar owner: {e}")
    return {}
```

📈 Coletar Estatísticas do Repositório

- Paginação automática para contagem de contribuidores
- Filtro temporal para issues fechadas
- Delay entre requisições para evitar rate limits

```
def get_repo_stats(owner, repo_name):
    stats = {
        "subscribers_count": 0,
        "last_year_commits": 0,
        "contributors": 0,
        "closed_issues": 0,
        "pull_requests": 0
    }

    try:
        subscribers = requests.get(
            f"https://api.github.com/repos/{owner}/{repo_name}/subscribers",
            headers=get_headers()
        )
        stats["subscribers_count"] = len(subscribers.json()) if subscribers.status_code == 200 else 0

        participation = requests.get(
            f"https://api.github.com/repos/{owner}/{repo_name}/stats/participation",
            headers=get_headers()
        )
        if participation.status_code == 200:
            stats["last_year_commits"] = sum(participation.json().get("all", [])[-52:])

        contributors = []
        page = 1
        while True:
            url = f"https://api.github.com/repos/{owner}/{repo_name}/contributors?page={page}&per_page=100"
            response = requests.get(url, headers=get_headers())
            if response.status_code == 200:
                page_contributors = response.json()
                if not page_contributors:
                    break
                contributors.extend(page_contributors)
                page += 1
                time.sleep(1)
            else:
                break
        stats["contributors"] = len(contributors)

        since_date = (datetime.now() - timedelta(days=180)).isoformat()
        closed_issues = requests.get(
            f"https://api.github.com/repos/{owner}/{repo_name}/issues?state=closed&since={since_date}",
            headers=get_headers()
        )
        stats["closed_issues"] = len(closed_issues.json()) if closed_issues.status_code == 200 else 0

        prs = requests.get(
            f"https://api.github.com/repos/{owner}/{repo_name}/pulls?state=all",
            headers=get_headers()
        )
        stats["pull_requests"] = len(prs.json()) if prs.status_code == 200 else 0

    except Exception as e:
        print(f"Erro ao buscar stats: {e}")

    return stats
```

🔍 Função Principal: Busca por Linguagem

- Rotação automática de tokens
- Intervalos entre requisições
- Consolidação em DataFrame

```
def fetch_repos_by_language(language):
    all_repos = []

    for page in range(1, PAGES + 1):
        try:
            url = f"https://api.github.com/search/repositories?q=language:{language}&sort=stars&page={page}&per_page={PER_PAGE}"
            response = requests.get(url, headers=get_headers())

            if response.status_code == 403:
                print("Rate limit excedido, trocando token...")
                time.sleep(60)
                continue

            response.raise_for_status()

            for repo in response.json()["items"]:
                repo_info = {
                    "name": repo["name"],
                    "owner": repo["owner"]["login"],
                    "stars": repo["stargazers_count"],
                    "forks": repo["forks_count"],
                    "language": repo["language"],
                    "created_at": repo["created_at"],
                    "updated_at": repo["updated_at"],
                    "size_kb": repo["size"],
                    "watchers_count": repo["watchers_count"],
                    "open_issues": repo["open_issues_count"]
                }

                repo_info.update(get_owner_info(repo["owner"]["url"]))
                repo_info.update(get_repo_stats(repo["owner"]["login"], repo["name"]))

                all_repos.append(repo_info)

                time.sleep(1)

        except Exception as e:
            print(f"Erro na página {page}: {e}")
            continue

    return pd.DataFrame(all_repos)
```

🚀 Execução e Exportação

- Arquivo github_repos_completos.csv com todas as linguagens
- Estrutura padronizada para análise

```
def main():
    all_data = pd.DataFrame()

    for lang in LANGUAGES:
        print(f"Coletando dados para {lang}...")
        df = fetch_repos_by_language(lang)
        if not df.empty:
            all_data = pd.concat([all_data, df], ignore_index=True)
            print(f"{len(df)} repositórios de {lang} coletados!")
            time.sleep(300)

    if not all_data.empty:
        all_data.to_csv("github_repos_completos.csv", index=False, encoding='utf-8')
        print(f"Todos os dados salvos em 'github_repos_completos2.csv'! Total: {len(all_data)} repositórios")
    else:
        print("Nenhum dado foi coletado.")

if __name__ == "__main__":
    main()
```

📋 Estrutura do Dataset (Colunas Extraídas)

Coluna	Tipo de Dado	Descrição	Exemplo
Informações Básicas			
`name`	string	Nome do repositório	`tensorflow`
`owner`	string	Login do usuário/organização dono	`google`
`language`	string	Linguagem principal do projeto	`Python`

Coluna	Tipo de Dado	Descrição	Exemplo
Estatísticas			
`stars`	integer	Número de estrelas	`175000`
`forks`	integer	Número de forks	`85000`
`watchers_count`	integer	Usuários acompanhando o repositório	`3200`
`subscribers_count`	integer	Inscritos no repositório (diferente de stars)	`1500`
Atividade			
`open_issues`	integer	Issues abertas no momento	`42`
`closed_issues`	integer	Issues fechadas nos últimos 6 meses	`128`
`pull_requests`	integer	Total de PRs (abertos + fechados)	`75`
`last_year_commits`	integer	Commits realizados nos últimos 12 meses	`890`
`contributors`	integer	Número de contribuidores únicos	`35`
Metadados			
`created_at`	datetime	Data de criação do repositório (UTC)	`2015-11-09T23:25:38Z`
`updated_at`	datetime	Data da última atualização	`2024-03-15T08:12:45Z`
`size_kb`	integer	Tamanho aproximado do repositório em KB	`10240`
Informações do Dono			
`owner_type`	string	Tipo do dono (`User` ou `Organization`)	`Organization`
`owner_public_repos`	integer	Quantidade de repositórios públicos do dono	`250`
`owner_location`	string	Localização geográfica declarada no perfil (opcional)	`Mountain View, California`

Repositorio

https://github.com/LucasjsSilva/data-set-repositorios