

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №4

Специальность ПО11

Выполнил
Зайченко С.В.
студент группы ПО11

Проверил
Крощенко А. А.
ст. преп. кафедры ИИТ

Брест 2025

Цель работы: научиться работать с Github API, приобрести практические навыки написания программ для работы с REST API или GraphQL API

Ход Работы

Напишите Python-скрипт, который анализирует самых активных контрибьюторов в указанном GitHub-репозитории за определённый период.

Что нужно сделать:

1. Запросить у пользователя:

❑ Репозиторий (owner/repo, например, tensorflow/tensorflow).

❑ Период анализа (за неделю, за месяц, за год).

❑ (Опционально) Минимальное количество коммитов для попадания в рейтинг.

2. Использовать GitHub API для получения списка контрибьюторов и их активности:

❑ Количество коммитов.

❑ Количество добавленных и удалённых строк кода.

❑ Количество открытых и закрытых pull requests.

❑ Количество открытых и закрытых issues.

❑ Количество комментариев к PR и issues.

3. Определить ТОП-5 самых активных контрибьюторов.

4. Построить график активности (matplotlib, seaborn).

```
from typing import Dict, List, Tuple
import matplotlib.pyplot as plt
import pandas as pd
import requests
```

```
class GitHubAnalyzer:
```

```
    def __init__(self, token: str):
        self.token = token
        self.headers = {
            'Authorization': f'token {token}',
            'Accept': 'application/vnd.github.v3+json'
        }
        self.base_url = 'https://api.github.com'
```

```
    def get_repo_contributors(self, owner: str, repo: str) -> List[Dict]:
        url = f'{self.base_url}/repos/{owner}/{repo}/contributors'
        response = requests.get(url, headers=self.headers)
        if response.status_code == 200:
```

```

        return response.json()
        raise RuntimeError(f"Ошибка при получении контрибьюторов:
{response.status_code}")

def get_user_activity(self, owner: str, repo: str, username: str) -> Dict:
    # Получаем коммиты
    commits_url = f'{self.base_url}/repos/{owner}/{repo}/commits'
    commits_params = {'author': username}
    commits_response = requests.get(commits_url, headers=self.headers,
params=commits_params)
    commits_count = len(commits_response.json()) if commits_response.status_code ==
200 else 0

    # Получаем PR
    pr_url = f'{self.base_url}/repos/{owner}/{repo}/pulls'
    pr_params = {'state': 'all', 'creator': username}
    pr_response = requests.get(pr_url, headers=self.headers, params=pr_params)
    prs = pr_response.json() if pr_response.status_code == 200 else []

    open_prs = len([pr for pr in prs if pr['state'] == 'open'])
    closed_prs = len([pr for pr in prs if pr['state'] == 'closed'])

    # Получаем issues
    issues_url = f'{self.base_url}/repos/{owner}/{repo}/issues'
    issues_params = {'creator': username}
    issues_response = requests.get(issues_url, headers=self.headers,
params=issues_params)
    issues = issues_response.json() if issues_response.status_code == 200 else []

    open_issues = len([issue for issue in issues if issue['state'] == 'open'])
    closed_issues = len([issue for issue in issues if issue['state'] == 'closed'])

    # Получаем дату последней активности
    last_activity = None
    if commits_count > 0:
        last_commit = commits_response.json()[0]
        last_activity = last_commit['commit']['author']['date']

    return {
        'commits': commits_count,
        'open_prs': open_prs,
        'closed_prs': closed_prs,
        'open_issues': open_issues,

```

```

        'closed_issues': closed_issues,
        'last_activity': last_activity
    }

def analyze_repository(self, owner: str, repo: str) -> List[Tuple[str, Dict]]:
    print(f"Анализируем вклад контрибьюторов в \"{owner}/{repo}\"...")

    contributors = self.get_repo_contributors(owner, repo)
    contributor_stats = []

    for contributor in contributors:
        username = contributor['login']
        stats = self.get_user_activity(owner, repo, username)
        contributor_stats.append((username, stats))

    # Сортируем по общему вкладу
    contributor_stats.sort(
        key=lambda x: x[1]['commits'] + x[1]['open_prs'] + x[1]['closed_prs'] +
            x[1]['open_issues'] + x[1]['closed_issues'],
        reverse=True
    )

    return contributor_stats

def plot_contributor_activity(self, contributor_stats: List[Tuple[str, Dict]], repo_name:
str):
    # Подготавливаем данные для графика
    top_5 = contributor_stats[:5]
    usernames = [stat[0] for stat in top_5]

    data = {
        'Коммиты': [stat[1]['commits'] for stat in top_5],
        'Открытые PR': [stat[1]['open_prs'] for stat in top_5],
        'Закрытые PR': [stat[1]['closed_prs'] for stat in top_5],
        'Открытые Issues': [stat[1]['open_issues'] for stat in top_5],
        'Закрытые Issues': [stat[1]['closed_issues'] for stat in top_5]
    }

    df = pd.DataFrame(data, index=usernames)

    # Создаем график
    plt.figure(figsize=(12, 6))
    df.plot(kind='bar', stacked=True)

```

```

plt.title(f'Активность контрибьюторов в {repo_name}')
plt.xlabel('Пользователь')
plt.ylabel('Количество')
plt.xticks(rotation=45)
plt.tight_layout()

# Сохраняем график
plt.savefig(f'{repo_name.replace("/", "_")}_contributors.png')
plt.close()

def main():
    # Запрашиваем токен GitHub
    token = input("Введите ваш GitHub токен: ")

    # Запрашиваем репозиторий
    repo_input = input("Введите репозиторий для анализа (owner/repo): ")
    owner, repo = repo_input.split('/')

    analyzer = GitHubAnalyzer(token)

    try:
        contributor_stats = analyzer.analyze_repository(owner, repo)

        # Выводим топ-5 контрибьюторов
        print("\nТОП-5 самых активных разработчиков:")
        for i, (username, stats) in enumerate(contributor_stats[:5], 1):
            print(f"{i}. {username} - {stats['commits']} коммитов, "
                  f"{stats['open_prs'] + stats['closed_prs']} PR, "
                  f"{stats['open_issues'] + stats['closed_issues']} issues")

        # Строим график
        analyzer.plot_contributor_activity(contributor_stats, repo_input)
        print(f"\nГрафики активности сохранены в \"{repo_input.replace('/',
        '_')}_contributors.png\"")

    except (RuntimeError, ValueError) as e:
        print(f"Произошла ошибка: {str(e)}")

if __name__ == "__main__":
    main()

```

Рисунок с результатом работы программы:

```
Введите ваш GitHub токен: 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
Введите репозиторий для анализа (owner/repo): kroschenko/spp_po11
Анализируем вклад контрибьюторов в "kroschenko/spp_po11"...

ТОП-5 самых активных разработчиков:
1. kroschenko - 30 коммитов, 30 PR, 0 issues
2. MorozovOriginal - 29 коммитов, 30 PR, 1 issues
3. AntonyukMick - 17 коммитов, 30 PR, 1 issues
4. CicliGs - 17 коммитов, 30 PR, 0 issues
5. ILGurin - 10 коммитов, 30 PR, 1 issues
```

Вывод графика контрибьюторов:

