

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  
Н. А. Антонюк  
студент группы ПО11

Проверил  
А. А. Крощенко  
ст. преп. кафедры ИИТ,  
11.04.2025 г.

Брест 2025

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

**Задание 1. используя Github API, реализовать предложенное задание на языке Python. Выполнить визуализацию результатов, с использованием графика или отчета. Можно использовать как REST API (рекомендуется), так и GraphQL.**

### **Вариант 2: Анализ активности разработчиков в open-source проекте**

Условие:

Напишите Python-скрипт, который анализирует активность разработчиков в указанном публичном GitHub-репозитории и составляет рейтинг самых активных контрибьюторов.

1. Запрашивает у пользователя имя репозитория (например, pallets/flask или microsoft/vscode).
  2. Использует GitHub API для получения списка всех контрибьюторов.
  3. Для каждого контрибьютора собирает данные:
    - ☐ Количество коммитов
    - ☐ Количество открытых pull requests
    - ☐ Количество закрытых pull requests
    - ☐ Количество открытых issues
    - ☐ Количество закрытых issues
    - ☐ Дата последней активности
  4. Определяет самых активных разработчиков по суммарному вкладу.
  5. Строит график вклада разработчиков (matplotlib / seaborn). Введите репозиторий для анализа (owner/repo): fastapi/fastapi  
Анализируем вклад контрибьюторов в "fastapi/fastapi"...
- ТОП-5 самых активных разработчиков:
1. user1 - 150 коммитов, 40 PR, 20 issues
  2. user2 - 120 коммитов, 35 PR, 15 issues
  3. user3 - 110 коммитов, 25 PR, 10 issues
  4. user4 - 90 коммитов, 20 PR, 8 issues
  5. user5 - 80 коммитов, 15 PR, 5 issues
- Графики активности сохранены в "fastapi\_contributors.png"

Выполнение:

### **Код программы:**

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

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 Exception(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 Exception as e:
    print(f"Произошла ошибка: {str(e)}")

if __name__ == "__main__":
    main()

```

### Рисунок с результатом работы программы:

```

PS C:\Users\Nikita\Desktop\4\src> python SPP_Lab4_Task1.py
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 - 26 коммитов, 30 PR, 0 issues
2. MorozovOriginal - 23 коммитов, 30 PR, 0 issues
3. AntonyukMick - 10 коммитов, 30 PR, 0 issues
4. Cicligs - 10 коммитов, 30 PR, 0 issues
5. ILGurin - 8 коммитов, 30 PR, 0 issues

Графики активности сохранены в "kroschenko_spp_po11_contributors.png"
PS C:\Users\Nikita\Desktop\4\src> 

```

### Вывод графика активностей пользователей :



