МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе No4

Специальность ПО11(о)

Выполнил
К. А. Головач,
студент группы ПО11

Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«26» апрель 2025 г.

Брест 2025

Вариант 6

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

Общее задание: 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

6. Анализ вклада разработчика в open-source на GitHub

Условие:
Напишите Python-скрипт, который:
1. Запрашивает у пользователя имя пользователя GitHub.
2. Получает все публичные репозитории, в которые этот пользователь вносил вклад (не только его собственные, но и форки, pull requests).
3. Для каждого репозитория определяет:
 Количество коммитов пользователя
 Количество открытых pull requests
 Количество закрытых pull requests
 Количество issues, созданных пользователем
4. Вычисляет уровень активности пользователя, например, на основе взвешенной формулы:
Активность = (Коммиты * 1) + (Открытые PR * 2) + (Закрытые PR * 3) + (Issues * 1.5)
5. Определяет самый активный проект, в котором пользователь работал.
6. Сохраняет данные в github_contribution.json.

Выполнение:
Код программы:

lab_4:
```
import requests
import json
from getpass import getpass
import matplotlib.pyplot as plt
import pandas as pd
from jinja2 import Environment, FileSystemLoader

# ==== 1. Базовые настройки ====
GITHUB_API_URL = "https://api.github.com"

def get_headers(token=None):
    headers = {"Accept": "application/vnd.github.v3+json"}
    if token:
        headers["Authorization"] = f"token {token}"
    return headers


# ==== 2. Функции получения данных из GitHub REST API ====
```

```
def get_user_repos(username, token):
    repos = []
    page = 1
    while True:
        url = f"{GITHUB_API_URL}/users/{username}/repos?per_page=100&page={page}"
        response = requests.get(url, headers=get_headers(token))
        if response.status_code != 200 or not response.json():
            break
        repos.extend(response.json())
        page += 1
    return repos


def get_commit_count(username, repo_full_name, token):
    url = f"{GITHUB_API_URL}/repos/{repo_full_name}/commits"
    page = 1
    total_commits = 0
    while True:
        params = {"author": username, "per_page": 100, "page": page}
        response = requests.get(url, headers=get_headers(token), params=params)
        if response.status_code != 200:
            break
        commits = response.json()
        total_commits += len(commits)
        if len(commits) < 100:
            break
        page += 1
    return total_commits


def get_prs(username, repo_full_name, token, state=None):
    url = f"{GITHUB_API_URL}/search/issues"
    query = f"is:pr author:{username} repo:{repo_full_name}"
    if state:
        query += f" state:{state}"
    page = 1
    total = 0
    while True:
        params = {"q": query, "per_page": 100, "page": page}
        response = requests.get(url, headers=get_headers(token), params=params)
        data = response.json()
        total += data.get("total_count", 0)
        items = data.get("items", [])
        if len(items) < 100:
            break
        page += 1
    return total


def get_issues(username, repo_full_name, token):
    url = f"{GITHUB_API_URL}/search/issues"
    query = f"is:issue author:{username} repo:{repo_full_name}"
    page = 1
    total = 0
    while True:
        params = {"q": query, "per_page": 100, "page": page}
        response = requests.get(url, headers=get_headers(token), params=params)
        data = response.json()
```

```
        total += data.get("total_count", 0)
        items = data.get("items", [])
        if len(items) < 100:
            break
        page += 1
    return total


# ==== 3. Анализ активности пользователя ====

def analyze_github_user(username, token=None):
    print(f"\nАнализ вклада пользователя: {username}")

    all_repos = get_user_repos(username, token)
    contribution_data = {
        "username": username,
        "repositories": [],
        "total_commits": 0,
        "total_open_prs": 0,
        "total_closed_prs": 0,
        "total_issues": 0,
        "activity_score": 0
    }

    max_activity_repo = {"name": "", "score": 0}

    for repo in all_repos:
        repo_name = repo["full_name"]
        print(f"Обработка репозитория: {repo_name}")

        commits = get_commit_count(username, repo_name, token)
        open_prs = get_prs(username, repo_name, token, state="open")
        closed_prs = get_prs(username, repo_name, token, state="closed")
        issues = get_issues(username, repo_name, token)

        activity = (commits * 1) + (open_prs * 2) + (closed_prs * 3) + (issues * 1.5)

        repo_data = {
            "repository": repo_name,
            "commits": commits,
            "open_prs": open_prs,
            "closed_prs": closed_prs,
            "issues": issues,
            "activity_score": activity
        }

        contribution_data["repositories"].append(repo_data)

        contribution_data["total_commits"] += commits
        contribution_data["total_open_prs"] += open_prs
        contribution_data["total_closed_prs"] += closed_prs
        contribution_data["total_issues"] += issues
        contribution_data["activity_score"] += activity

        if activity > max_activity_repo["score"]:
            max_activity_repo["name"] = repo_name
            max_activity_repo["score"] = activity

    contribution_data["active_repository"] = max_activity_repo
```

```
    with open("github_contribution.json", "w", encoding="utf-8") as f:
        json.dump(contribution_data, f, indent=4, ensure_ascii=False)

    return contribution_data


# ==== 4. Визуализация графика активности ====

def visualize_activity(data):
    repos = [item['repository'] for item in data['repositories']]
    activity_scores = [item['activity_score'] for item in data['repositories']]

    df = pd.DataFrame({
        'Репозиторий': repos,
        'Активность': activity_scores
    })

    df = df.sort_values(by='Активность', ascending=False).head(10)

    plt.figure(figsize=(10, 6))
    plt.barh(df['Репозиторий'], df['Активность'], color='skyblue')
    plt.xlabel('Оценка активности')
    plt.title(f'Топ-10 репозиториев ({data["username"]})')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.savefig("github_activity_chart.png")
    plt.close()
    print("График активности сохранён как github_activity_chart.png")


# ==== 5. Генерация HTML-отчёта ====

def generate_html_report(data):
    env = Environment(loader=FileSystemLoader("."))
    template = env.get_template("template.html")

    repositories = [repo['repository'] for repo in data['repositories']]
    activity_scores = [repo['activity_score'] for repo in data['repositories']]

    rendered = template.render(
        username=data['username'],
        total_commits=data['total_commits'],
        total_open_prs=data['total_open_prs'],
        total_closed_prs=data['total_closed_prs'],
        total_issues=data['total_issues'],
        activity_score=round(data['activity_score'], 2),
        active_repo=data['active_repository']['name'],
        repositories=repositories,
        activity_scores=activity_scores
    )

    with open("github_contribution_report.html", "w", encoding="utf-8") as f:
        f.write(rendered)

    print("HTML-отчет создан как github_contribution_report.html")


# ==== 6. Основная функция запуска ====
```

```
def main():
    username = input("Введите имя пользователя GitHub: ")
    use_token = input("Хотите использовать персональный токен? (y/n): ").lower() == 'y'
    token = None
    if use_token:
        token = getpass("Введите ваш GitHub Personal Access Token: ")

    result = analyze_github_user(username, token)

    print(f"\nПользователь {username} внес вклад в {len(result['repositories'])} репозиториев.")
    print(f"Общее количество коммитов: {result['total_commits']}")
    print(f"Открытых pull requests: {result['total_open_prs']}")
    print(f"Закрытых pull requests: {result['total_closed_prs']}")
    print(f"Созданных issues: {result['total_issues']}")
    print(f"Активность: {result['activity_score']:.1f} баллов")
    print(f"Самый активный проект: {result['active_repository']['name']} "
          f"(оценка активности: {result['active_repository']['score']:.1f})")
    print("Результаты сохранены в github_contribution.json")

    visualize_activity(result)
    generate_html_report(result)


# ==== 7. Точка входа ====
if __name__ == "__main__":
    main()
```

template.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>GitHub Активность {{ username }}</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
    <h1>Анализ вклада пользователя {{ username }} в GitHub</h1>

    <ul>
        <li><strong>Коммиты:</strong> {{ total_commits }}</li>
        <li><strong>Открытые PR:</strong> {{ total_open_prs }}</li>
        <li><strong>Закрытые PR:</strong> {{ total_closed_prs }}</li>
        <li><strong>Issues:</strong> {{ total_issues }}</li>
        <li><strong>Общий рейтинг активности:</strong> {{ activity_score }}</li>
        <li><strong>Самый активный проект:</strong> {{ active_repo }}</li>
    </ul>

    <h2>Активность по репозиториям</h2>
    <div id="chart" style="width:90%; max-width:1000px; height:600px;"></div>

    <script>
        var data = [{
            x: {{ activity_scores|tojson }},
            y: {{ repositories|tojson }},
            type: 'bar',
```

```
            orientation: 'h'
        }];
        Plotly.newPlot('chart', data);
    </script>
</body>
</html>
```

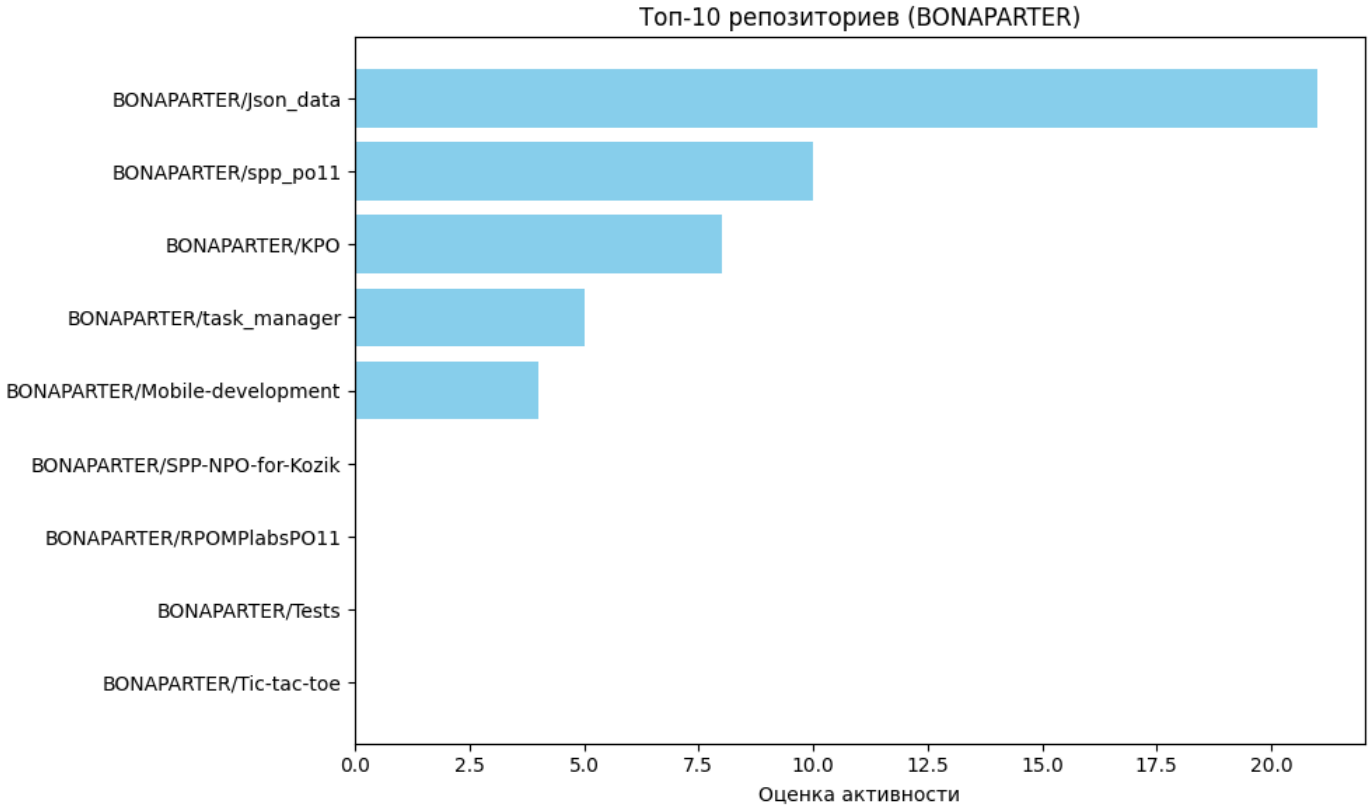
Результаты работы программы:

```
C:\Users\kirja\OneDrive\PythonProject\.venv\Scripts\python.exe C:\Users\
Введите имя пользователя GitHub: BONAPARTER
Хотите использовать персональный токен? (y/n): n

Анализ вклада пользователя: BONAPARTER
Обработка репозитория: BONAPARTER/Json_data
Обработка репозитория: BONAPARTER/KPO
Обработка репозитория: BONAPARTER/Mobile-development
Обработка репозитория: BONAPARTER/RPOMPlabsPO11
Обработка репозитория: BONAPARTER/SPP-NPO-for-Kozik
Обработка репозитория: BONAPARTER/spp_po11
Обработка репозитория: BONAPARTER/task_manager
Обработка репозитория: BONAPARTER/Tests
Обработка репозитория: BONAPARTER/Tic-tac-toe

Пользователь BONAPARTER внес вклад в 9 репозиториев.
Общее количество коммитов: 48
Открытых pull requests: 0
Закрытых pull requests: 0
Созданных issues: 0
Активность: 48.0 баллов
Самый активный проект: BONAPARTER/Json_data (оценка активности: 21.0)
Результаты сохранены в github_contribution.json
График активности сохранён как github_activity_chart.png
HTML-отчет создан как github_contribution_report.html

Process finished with exit code 0
```



Вывод: научился работать с Github API, а также визуализировать активность пользователя в каждом репозитории.