Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4

По дисциплине «Современные платформы программирования»

Выполнил:
Студент 3 курса
Группы ПО-11
Микулич М. И.
Проверил:
Козик И. Д.

Брест 2025

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Python.

Вариант 14

Задание 1:

Напишите Python-скрипт, который:

1. Запрашивает у пользователя ключевое слово или тему (например, "machine learning", "web development", "blockchain").

Крощенко А.А., Современные платформы программирования, ЛР4, 2025

2. Использует GitHub API для поиска 100+ самых популярных репозиториев по этому ключевому слову.
3. Для каждого найденного репозитория собирает:
- Язык программирования
- Количество звезд
- Количество форков
- Количество открытых issues
- Дату последнего обновления
4. Анализирует какие технологии чаще всего используются в данной области, строя рейтинг языков программирования.
5. Визуализирует результаты:
- Диаграмму популярных языков программирования (matplotlib, seaborn, plotly)
- График популярности репозиториев (по звёздам и активности)
- График "старения" репозиториев (когда последний коммит)

Код программы:

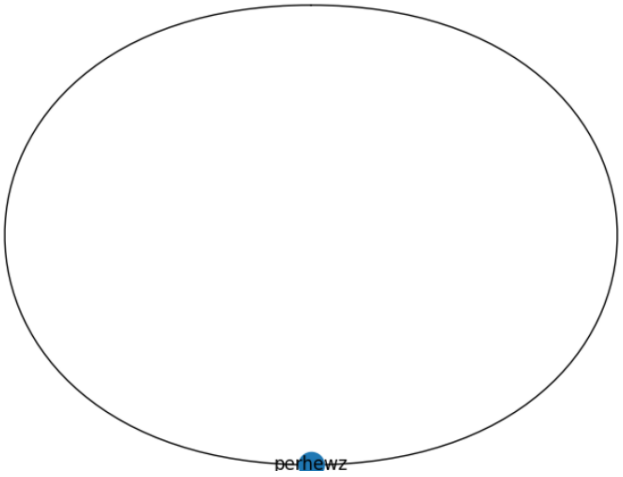
```
import requests
import networkx as nx
import matplotlib.pyplot as plt

# Функция для запроса к GraphQL API GitHub
def query_github_api(username, token):
    headers = {
        "Authorization": f"Bearer {token}",
        "Content-Type": "application/json"
    }

    query = """
        query ($username: String!) {
            user(login: $username) {
                contributionsCollection {
                    repositoryContributions(first: 100) {
                        edges {
                            node {
                                repository {
                                    name
                                    owner {
                                        login
                                    }
                                }
                            }
                        }
                    }
                }
                pullRequests(first: 100) {
                    edges {
                        node {
                            repository {
                                name
                                owner {
```

```
                                    login
                                }
                            }
                            author {
                                login
                            }
                        }
                    }
                }
                issues(first: 100) {
                    edges {
                        node {
                            repository {
                                name
                                owner {
                                    login
                                }
                            }
                            author {
                                login
                            }
                        }
                    }
                }
                starredRepositories(first: 100) {
                    edges {
                        node {
                            name
                            owner {
                                login
                            }
                        }
                    }
                }
            }
        }
    """

    variables = {"username": username}
    response = requests.post("https://api.github.com/graphql",
                             headers=headers,
                             json={"query": query, "variables": variables})

    return response.json()

# Функция для сбора взаимодействий
def collect_interactions(data, username):
    interactions = set()

    # Коммиты
    for contribution in data["data"]["user"]["contributionsCollection"]["repositoryContributions"]["edges"]:
        repo = contribution["node"]["repository"]
        interactions.add(repo["owner"]["login"])

    # Pull Requests
    for pr in data["data"]["user"]["pullRequests"]["edges"]:
        repo = pr["node"]["repository"]
        interactions.add(pr["node"]["author"]["login"])
        interactions.add(repo["owner"]["login"])

    # Issues
    for issue in data["data"]["user"]["issues"]["edges"]:
        repo = issue["node"]["repository"]
        interactions.add(issue["node"]["author"]["login"])
        interactions.add(repo["owner"]["login"])

    # Звёзды
    for star in data["data"]["user"]["starredRepositories"]["edges"]:
        repo = star["node"]
        interactions.add(repo["owner"]["login"])

    return interactions

# Функция для построения графа
def build_graph(interactions, username):
    G = nx.Graph()
    G.add_node(username)

    for interaction in interactions:
        G.add_node(interaction)
        G.add_edge(username, interaction)
```

```
    return G

# Основная функция
def main():
    username = input("Введите имя пользователя GitHub: ")
    token = input("Введите токен GitHub: ")

    data = query_github_api(username, token)
    interactions = collect_interactions(data, username)

    print(f"Анализируем взаимодействия пользователя {username}...")
    print(f"Найдено {len(interactions)} связанных разработчиков.")

    G = build_graph(interactions, username)

    # Визуализация графа
    nx.draw(G, with_labels=True)
    plt.savefig("github_network.png")

    # Сохранение графа в JSON
    nx.write_gexf(G, "github_network.gexf")

    print("Граф сохранён в github_network.gexf")
    print("Визуализация графа сохранена в github_network.png")

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

Результат работы:

```
Введите имя пользователя GitHub: perhewz
Введите токен GitHub: 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
Анализируем взаимодействия пользователя perhewz...
Найдено 1 связанных разработчиков.
Граф сохранён в github_network.gexf
Визуализация графа сохранена в github_network.png
```



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