МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №4
По дисциплине “**Современные платформы программирования**”
Тема: “**Работа с Github API на языке Python**”

Выполнил:
студент группы ПО-11
Надежук А.Г.
Проверил:
Козик И. Д.

Брест 2025

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

Вариант 3

Задание. Автоматическое отслеживание новых релизов и обновлений в репозитории GitHub.

Ход работы

Условие: напишите Python-скрипт, который:

1. Запрашивает у пользователя список репозиториев GitHub для мониторинга (например, fastapi/fastapi, pallets/flask).
2. Использует GitHub API для получения информации о последних релизах и тегах в каждом репозитории.
3. Проверяет, появились ли новые версии или обновления с момента последнего запуска скрипта.
4. Если есть обновления, собирает следующую информацию:
    - Номер новой версии
    - Дата релиза
    - Список изменений (changelog)
    - Ссылку на страницу релиза

Код программы:

```
import requests
import json
import os
from datetime import datetime, timezone
import logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

TRACKED_REPOS_FILE = "tracked_repos.json"
GITHUB_TOKEN = os.getenv("GITHUB_TOKEN")

if GITHUB_TOKEN is None:
    logging.error("Нужно установить переменную окружения GITHUB_TOKEN")
    exit()

def load_tracked_repos():
    try:
        with open(TRACKED_REPOS_FILE, "r") as f:
            return json.load(f)
    except FileNotFoundError:
        return {}
    except json.JSONDecodeError:
        logging.error(f"Ошибка при чтении файла {TRACKED_REPOS_FILE}. Файл может быть поврежден.
Создаётся пустая конфигурации.")
        return {}

def save_tracked_repos(repos_data):
    with open(TRACKED_REPOS_FILE, "w") as f:
        json.dump(repos_data, f, indent = 2, default = str)

def check_repo_updates(owner, repo, repos_data):
    repo_name = f"{owner}/{repo}"
    logging.info(f"Проверяем обновления для {repo_name}...")

    url = f"https://api.github.com/repos/{owner}/{repo}/releases"
    headers = {
        "Authorization": f"token {GITHUB_TOKEN}",
        "Accept": "application/vnd.github+json"
    }

    try:
        response = requests.get(url, headers = headers)
        response.raise_for_status()
        releases = response.json()

        nowTime = datetime.now(timezone.utc)
```

```
        now_time_str = nowTime.strftime("%Y-%m-%dT%H:%M:%SZ")

        if repo_name in repos_data and repos_data[repo_name].get("last_version"):
            last_version = repos_data[repo_name]["last_version"]
            last_release_date_str = repos_data[repo_name].get("release_date", None)
            last_release_url = repos_data[repo_name].get("release_url", "Нет URL")
            last_release_body = repos_data[repo_name].get("release_body", "Нет описания")

            if last_release_date_str:
                 last_release_date = datetime.strptime(last_release_date_str, "%Y-%m-
%dT%H:%M:%SZ")
                 formatted_last_release_date = last_release_date.strftime("%Y-%m-%d")
            else:
                formatted_last_release_date = "Неизвестно"
        else:
            last_version = None
            formatted_last_release_date = None
            last_release_url = None
            last_release_body = None

        if not releases:
            logging.info(f"Нет релизов для {repo_name}.")
            if last_version:
                print(f"Нет новых релизов для {repo_name}.")
                print(f"Последний известный релиз: {last_version}
({formatted_last_release_date})")
                print(f"URL: {last_release_url}")
                print(f"Основные изменения: {last_release_body}")
            else:
                print(f"Нет информации о релизах для {repo_name}.")
            return

        latest_release = releases[0]
        release_name = latest_release.get("tag_name", latest_release.get("name", "Нет имени"))
        release_date_str = latest_release["published_at"]
        release_date_time = datetime.strptime(release_date_str, "%Y-%m-
%dT%H:%M:%SZ").replace(tzinfo = timezone.utc)
        formatted_release_date = release_date_time.strftime("%Y-%m-%d")
        release_url = latest_release["html_url"]
        release_body = latest_release["body"]

        if repo_name not in repos_data:
            repos_data[repo_name] = {"last_checked": None, "last_version": None, "release_date":
None, "release_url": None, "release_body": None}

        last_checked = repos_data[repo_name].get("last_checked", None)
        if last_checked is not None:
            last_checked_time = datetime.strptime(last_checked, "%Y-%m-
%dT%H:%M:%SZ").replace(tzinfo=timezone.utc)
        else:
            last_checked_time = datetime.now(timezone.utc).strftime("%Y-%m-%dT%H:%M:%SZ")

        if last_checked is None or release_date_time > last_checked_time:
            print(f"Найден новый релиз: {release_name} ({formatted_release_date})")
            print(release_url)
            print(f"Основные изменения: {release_body}")

            repos_data[repo_name]["last_checked"] = now_time_str
            repos_data[repo_name]["last_version"] = release_name
            repos_data[repo_name]["release_date"] = release_date_str
            repos_data[repo_name]["release_url"] = release_url
            repos_data[repo_name]["release_body"] = release_body
        else:
            logging.info(f"Нет новых релизов для {repo_name}.")
            print(f"Последний известный релиз: {last_version} ({formatted_last_release_date})")
            print(f"URL: {last_release_url}")
            print(f"Основные изменения: {last_release_body}")
            repos_data[repo_name]["last_checked"] = now_time_str

    except requests.exceptions.RequestException as e:
```

```
        logging.error(f"Произошла ошибка при запросе API для {repo_name}: {e}")
    except Exception as e:
        logging.error(f"Произошла непредвиденная ошибка при обработке {repo_name}: {e}")

if __name__ == "__main__":
    repos_data = load_tracked_repos()
    repo_list_str = input("Введите репозитории для отслеживания (через запятую, например:
requests/requests, fastapi/fastapi, pallets/flask): ")
    repo_list = [repo.strip() for repo in repo_list_str.split(",")]

    for repo_name in repo_list:
        try:
            owner, repo = repo_name.split("/")
            check_repo_updates(owner, repo, repos_data)
        except ValueError:
            logging.error(f"Неверный формат репозитория: {repo_name}. Ожидается формат
owner/repo.")

    save_tracked_repos(repos_data)
```

Результаты работы программы:

Введите репозитории для отслеживания (через запятую, например: requests/requests, fastapi/fastapi, pallets/flask): pallets/flask
2025-04-05 10:40:37,823 - INFO - Проверяем обновления для pallets/flask...
2025-04-05 10:40:38,707 - INFO - Нет новых релизов для pallets/flask.
Последний известный релиз: 3.1.0 (2024-11-13)
URL: https://github.com/pallets/flask/releases/tag/3.1.0
Основные изменения: This is the Flask 3.1.0 feature release. A feature release may include new features, remove previously deprecated code, add new deprecations, or introduce potentially breaking changes. We encourage everyone to upgrade, and to use a tool such as [pip-tools](https://pypi.org/project/pip-tools/) to pin all dependencies and control upgrades. Test with warnings treated as errors to be able to adapt to deprecation warnings early.

PyPI: https://pypi.org/project/Flask/3.1.0/
Changes: https://flask.palletsprojects.com/en/stable/changes/#version-3-1-0
Milestone: https://github.com/pallets/flask/milestone/33?closed=1

- Drop support for Python 3.8. #5623
- Update minimum dependency versions to latest feature releases. Werkzeug >= 3.1, ItsDangerous >= 2.2, Blinker >= 1.9. #5624, #5633
- Provide a configuration option to control automatic option responses. #5496
- `Flask.open_resource`/`open_instance_resource` and `Blueprint.open_resource` take an `encoding` parameter to use when opening in text mode. It defaults to `utf-8`. #5504

Вывод: научился работать с Github API.