Пояснительная записка

Сервис для вывода статистических данных о публичном репозитории github.com

Выполнил: программист, Миронов А.В.

(github.com username: skyshy0707)

2020

1. Руководство по использованию.
2. Назначение

Скрипт (далее – сервис) **gitrepo\_stat.py** предназначен для вывода статистических данных репозитория на экран. Позволяет вывести следующую численную информацию за заданный период времени:

1. Число открытых и число старых pulls. Старыми считаются те pulls, которые момента их создания не были закрыты в течении 30 дней и до сих пор открыты на заданном периоде времени.
2. Число закрытых pulls.
3. Число открытых и число старых issues. Старыми считаются те issues, которые с момента их создания не были закрыты в течении 14 дней и до сих пор открыты на заданном периоде времени.
4. Число закрытых issues.
5. Тридцать участников с числом их коммитов. Данные представлены в табличном виде. Участники отсортированы по убыванию числа их коммитов.
6. Системные требования

Для работы сервиса необходимо следующее:

Операционная система: Windows 10, 64-bit.

Установленное ПО: Python 3.8

1. Внешние зависимости

Требуется установить библиотеку prettytable для python.

Установка библиотеки prettytable:

1. Запустить cmd
2. pip install prettytable
3. Использование

**path>python** **gitrepo\_stat.py** [-h] --url URL --type TYPE [--state STATE] [--period PERIOD]

где path – абсолютный путь к скрипту.

URL – url-адрес публичного репозитория github.com. Обязательный аргумент. Допускается указывать url-адрес без указания протокола https.

Примеры корректного указания url-адреса:

--url github.com/user/repo

--url <https://github.com/user/repo>

TYPE – тип запроса. Может принимать значение из следующего множества {contributors, issues, pulls}

--type contributors – с данным значением выполняется поиск тридцати участников (contributors) c наибольшим числом коммитов на заданном периоде времени. Период времени задаётся аргументом period. Результат выполнения сервиса – вывод данных ***(участники: число коммитов за указанный период)*** в табличном виде.

--type issues – с данным значением выполняется поиск числа issues заданного статуса, заданным в соответствии со значением аргумента state. В том числе будет выведено число старых issues, если аргументу state задано значение open.

--type pulls – с данным значением выполняется поиск числа pulls заданного статуса, заданным в соответствии со значением аргумента state. В том числе будет выведено число старых pulls, если аргументу state задано значение open.

STATE – состояние искомых запросов. Необязательный аргумент. Может принимать значение из следующего множества {open, closed}. По умолчанию state=closed.

--state closed – с данным значением выполняется поиск pulls или issues, которые закрылись на заданном периоде времени.

--state open – с данным значением выполняется поиск pulls или issues, которые открыты на заданном периоде времени.

Период времени задаётся аргументом period.

PERIOD – интервал времени, на котором производится анализ репозитория. По результату анализа выводится статистика по учёту запросов в зависимости от значения аргументов type, state. Необязательный аргумент. По умолчанию period=’2008-01-01---YYYY-MM-DD’, где YYYY-MM-DD – условное обозначение даты текущего дня.

--period ‘-INF---YYYY-MM-DD’ – если только дата начала анализа не ограничена

--period ‘YYYY-MM-DD---INF’ – если только дата конца анализа не ограничена

--period ‘YYYY-MM-DD---YYYY-MM-DD’ – если нужно указать дату начала и конца анализа явно

Даты указываются в формате ISO.

-h – вызов справочного сообщения:

**path>python** **gitrepo\_stat.py** -h

1. Ограничения

Для вызова функций сервиса требуется авторизация пользователя (требуется наличие учётной записи в github.com) по заранее сгенерированному персональному токену. Как сгенерировать токен, читайте здесь:

<https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/creating-a-personal-access-token>

Убедитесь, что токен действителен.

Количество запросов не может превышать 5 000 в час, но не более 30 поисковых запросов в минуту для каждого пользователя с базовой идентификацией, независимо от того, сколько токенов у него есть. Каждый запрос проходит проверку подлинности.

Структура запросов к API github.com может поменяться в любой момент, поэтому не исключено, что сервис по какой-то из этих причин может оказаться неработоспособным.

Статистические результаты содержат данные тех запрашиваемых ресурсов, к которым у пользователя есть доступ:

<https://docs.github.com/en/free-pro-team@latest/rest/reference/search#access-errors-or-missing-search-results>

1. Организация CI/CD.
2. Общие требования, предъявляемые к разработке.

Каждый участвующий в поддержке или разработке данного сервиса может заниматься разработкой определённого интерфейса полностью или частично – в случае, если ему поручено разрабатывать для этого интерфейса отдельные методы или классы.

Разработчик пишет код на python 3.8 под Windows OS.

Разработчик переносит код с публичного репозитория к себе в локальный репозиторий на ПК и работает в нём, доводя код до надлежащего состояния по п. 2.

После этого, код отправляется в публичный репозиторий и тестируется на одной из систем CI (например, Microsoft Team Foundation Server). Тестирование по п. 3.1 происходит после каждого изменения сборки.

Если сборка прошла успешно, то сервису присваивается версия и далее тестирование по п. 3.2 производится на серверах компании (CD) QA-инженерами.

QA-инженеры запускают подготовленный скрипт, который тестирует отдельные функции сервиса. Для проверки работоспособности и отказоустойчивости, скрипт запускает сервис необходимое количество раз, чтобы количество запросов API превысило предельный уровень. *Ожидаемое поведение:* При превышении числа запросов в час, сервис приостанавливает свою работу и выдаёт сообщение «Превышение числа запросов. Подождите *n* секунд», *n*>0. По истечению указанного времени, сервис возобновляет работу, производя повторно запрос, на котором произошло событие превышения предельного числа запросов в час.

1. Рефакторинг.

В ходе разработки код нуждается в рефакторинге. Рефакторинг проводится с помощью одной из библиотек, предназначенных для анализа кода на несоответствие принципам и устоявшимся паттернам проектирования, принципам SOLID.

1. Тестирование.

3.1. Тестирование unit-тестами (CI).

При разработке каждого нового метода разработчик пишет автоматизированный unit-тест. Для написания теста используется библиотека unittest.

В теле метода определяются ветки выполнения кода. Для каждой из веток определяется комбинация подаваемых параметров на вход метода. Значения параметров должны приводить к выполнению кода в заданной ветке. Сравнивается ожидаемый и получаемый в результате теста результат. Если они не равны, то метод тест не проходит.

* 1. Интеграционное тестирование (CD).

После разработки интерфейса и пройденных unit-тестов его функционала проводится интеграционное тестирование на корректность работы выполняемых им функций. Под корректностью работы выполняемых функций следует понимать, что результаты выполнения кода будут ожидаемыми для конечного пользователя сервиса. Например, при готовом функционале, отвечающим за вывод в stdout числа закрытых issues, проверяется, что число закрытых issues действительно соответствует указанному на странице issues репозитория в github.com. Проверяется, что текст сообщения «Количество closed issues:» выводится в stdout, проверятся корректность указанного периода в сообщении, которое должно соответствовать изначально введённому при запуске скрипта и т.д.

1. Подготовка к публикации кода.

Если код успешно прошёл все этапы сборки и тестирования, то сервис готов к выпуску.