Zadanie Rekrutacyjne GWP

Należy napisać prosty serwer HTTP przechowujący w pamięci lub bazie danych zapisane urle oraz pobrane dane, udostępniający operacje CRUD (Create/Read/Update/Delete) na tych danych jako API RESTowe. Zadanie powinno zostać zrealizowane w dowolnym z następujących języków: C/C++/python/GO. Zachęcamy do próby rozwiązania tego zadania w GO.

Założenia:

- Serwer powinien nasłuchiwać na lokalnym interfejsie sieciowym na porcie 8080.
- Klucz "id" powinien być integerem.
- Maksymalny rozmiar payloadu POST to 1MB.
- Worker powinien pobierać cyklicznie dane z "url" z zadanym czasem co ile ma pobierać "interval" (w sekundach)
- Worker powinien mieć ustawiony timeout 5s na pobranie "url"

Poniżej krótki opis endpointów HTTP, które powinny być udostępniane przez serwer oraz ich przykładowe wywołanie z użyciem programu curl.

Tworzenie/aktualizacja nowego obiektu: POST /api/fetcher

Zadaniem tego endpointu jest wpisywanie nowej wartości obiektu pod oczekiwany klucz.

Poprawne wywołanie i odpowiedź serwera

```
$ curl -si 127.0.0.1:8080/api/fetcher -X POST -d '{"url":
"https://httpbin.org/range/15","interval":60}'
HTTP/1.1 200 OK
{"id": 1}
```

Wynik w formacie JSON jako obiekt.

Niepoprawne wywołanie (nieprawidłowy klucz) i odpowiedź serwera.

$ curl -si 127.0.0.1:8080/api/fetcher -X POST -d 'niepoprawny json' HTTP/1.1 400 Bad Request

Niepoprawne wywołanie (za duży obiekt) i odpowiedź serwera.

$ curl -si 127.0.0.1:8080/api/fetcher -X POST -d 'ponad 1 MB danych...' HTTP/1.1 413 Request Entity Too Large

Usuwanie urla: DELETE /api/fetcher/<id>

Usuwanie wskazanego urla.

```
$ curl -si 127.0.0.1:8080/api/fetcher -X POST -d '{"url":
"https://httpbin.org/range/15","interval":60}'
HTTP/1.1 200 OK
{"id": 12}
```

$ curl -s 127.0.0.1:8080/api/fetcher/12 -X DELETE $ curl -s 127.0.0.1:8080/api/fetcher/12/history -i HTTP/1.1 404 Not Found

Listowanie zapisanych urli GET /api/fetcher

Listowanie wszystkich zapisanych urli:

```
$ curl -s 127.0.0.1:8080/api/fetcher
[{"id":1,"url":"https://httpbin.org/range/15","interval":60},
{"id":2,"url": "https://httpbin.org/delay/10","interval":120}]
```

Wynik w formacie JSON jako lista obiektów.

Pobieranie historii pobrań: GET /api/fetcher/<id>/history

Pobieranie historii pobranych danych spod klucza <id>.

$ curl -si 127.0.0.1:8080/api/fetcher/abc HTTP/1.1 400 Bad Request $ curl -si 127.0.0.1:8080/api/fetcher/99 HTTP/1.1 404 Not Found

```
$ curl -si 127.0.0.1:8080/api/fetcher/1/history
HTTP/1.1 200 OK
[
    {
        "response": "abcdefghijklmno",
        "duration": 0.571,
        "created_at": 1559034638.31525,
    },
    {
        "response": null,
        "duration": 5,
        "created_at": 1559034938.623,
    },
]
```

Wynik w formacie JSON jako lista obiektów.

Worker pobierający dane z urli

Próbuje pobierać w background dane z "url" z zadanym "interval". W przypadku błędu timeout lub innego do pola "response" powinien zostać zapisany `null`

Do pola "response" zapisujemy cały response jaki zwróci "url"

Na przykład:

```
curl -si httpbin.org/range/50
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwx
```

Czyli pod response zapisujemy "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwx"

Część rozszerzona

- Napisanie testów

Wskazówki

- Do wykonania zadania przydatny będzie router HTTP (dla golang polecamy https://github.com/pressly/chi, dla pythona polecamy https://github.com/huge-success/sanic), ale w wyborze bibliotek pozostawiamy pełną dowolność.
- Kod źródłowy powinien zostać umieszczony w serwisie github.com z nazwą projektu base64(<Imie> + "-" + <Nazwisko>).
- W razie jakichkolwiek wątpliwości/pytań zapraszamy do kontaktu - chętnie pomożemy.