

# Projekt interfejsu sieciowego wyszukiwarki Poliqarp2

Bartosz Zaborowski

Marzec 2015

## 1 Wstęp

Niniejszy dokument opisuje projektowany interfejs sieciowy wyszukiwarki. Interfejs ten stanowi podstawowy sposób komunikacji z silnikiem wyszukiwarki także i w wersji lokalnej. W związku z tym został już zaimplementowany w nieco uproszczonej wersji w ramach kamienia milowego “M24 — Silnik przeszukiwania korpusu (wersja lokalna)”. Poniżej opisuję pełną przewidywaną postać interfejsu, przeznaczoną do udostępnienia publicznego.

## 2 Założenia ogólne

Opisywany interfejs został zbudowany w technice REST. Jest to powszechnie stosowane rozwiązanie, dzięki czemu integracja różnych narzędzi z wyszukiwarką powinna być łatwa i mało pracochłonna. W związku z tym warstwą transportową dla interfejsu jest protokół HTTP 1.1. Poszczególne elementy wyszukiwarki są udostępnione zgodnie ze standardem pod adresami URL, z parametrami zakodowanymi w ścieżce dostępu. Szczegółowo te adresy URL są omówione w sekcji 3.

### 2.1 Format danych

Technika REST w ogólności przewiduje różne formaty treści przesyłanych komunikatów. Ze względu na ograniczone możliwości oraz wymagania wydajnościowe w Poliqarpie 2 ograniczamy się do formatu JSON w kodowaniu UTF-8. Format JSON dotyczy komunikatów przesyłanych w obu kierunkach. Przy tym, w kierunku klient->silnik treść JSON jest przesyłana tylko w niektórych sytuacjach (udokumentowanych explicite), w pozostałych zostanie zignorowana. Natomiast w kierunku silnik->klient w standardowych okolicznościach odpowiedź jest zwracana w postaci nie pustego dokumentu JSON. Wymagane jest, żeby klient wysyłając treść JSON do silnika ustawił nagłówek *Content-Type* zgodnie ze standardem na *application/json*.

Silnik wyszukiwarki oznacza zwracane odpowiedzi stosownymi kodami odpowiedzi HTTP. W przypadku błędnego żądania standardowym kodem 4XY towarzyszy dokument JSON opisujący błąd w sposób czytelny dla człowieka. Odpowiedzi z kodami 5XY są rozumiane w standardowy sposób, ale nieprzeznaczone dla zasygnalizowania błędnego żądania, więc nie zawierają dodatkowych opisów. Lista możliwych odpowiedzi sygnalizujących błąd znajduje się w sekcji 4. Treści odpowiedzi oznaczających sukces i kody w zależności od kontekstu są omówione szczegółowo w sekcji 3.

### 2.2 Bezpieczeństwo

Projekt nie przewiduje żadnych zaawansowanych zabezpieczeń dostępu ponad to, co oferuje specyfikacja HTTP *Basic authentication*. Dla zachowania poufności wskazane jest przysyłanie treści komunikacji z interfejsem za pomocą połączenia HTTPS. Dostępność nieszyfrowanego połączenia HTTP i szyfrowanego HTTPS zależy od ustawień konkretnej instancji uruchomionego silnika wyszukiwarki.

## 3 Udostępniane elementy interfejsu

### 3.1 Przyjęte konwencje

W poniższych sekcjach są opisane poszczególne “zasoby”<sup>1</sup> API Poliqarpa2. Każdy “zasób” ma podany wzorec adresu dostępowego i opisane udostępniane metody HTTP. Niektóre wzorce zawierają atrybuty — z nazwami zawartymi między znakami < oraz >. Jeśli przy opisie danej metody nie zaznaczono inaczej, spodziewany kod odpowiedzi HTTP to 200. Jeśli nie zaznaczono inaczej, silnik wyszukiwarki nie oczekuje danych w treści żądania HTTP. W przykładowych listingach obiektów JSON napis (...) oznacza fragment wycięty dla zwiększenia czytelności.

Konwencje nazw atrybutów w obiektach JSON:

- **"links"**: tablica zawierająca lokalizacje “zasobów” związanych z obecnym żądaniem. Każda lokalizacja jest w postaci obiektu z atrybutami:
  - **"href"**: bezwzględny URL
  - **"rel"**: relacja między żądaniem a “zasobem”. Dla żądań POST wartość **"self"** oznacza zasób wygenerowany żądaniem.
- **"error"**: opis błędu zrozumiały dla człowieka

UWAGA: przy implementacji klienta zaleca się korzystać ze zwróconych przez silnik gotowych adresów do poszczególnych “zasobów”. O ile wzorce są ustalone dla danej wersji API, to niektóre szczegóły dotyczące wyglądu identyfikatorów są prywatną cechą implementacji i mogą się zmieniać.

### 3.2 Punkt wejściowy interfejsu

Wzorec URL: /

#### 3.2.1 metoda GET

Strona powitalna:

Listing 1: przykład zwracanego obiektu JSON

```
1 {
2   "Poliqarp2 API" : "0.1",
3   "help" : "use OPTIONS method for discovering the API"
4 }
```

Pod kluczem "Poliqarp2 API" dostępny jest numer wersji API. W ramach jednej wersji struktura adresów i format/układ wyników jest ustalony.

Możliwe błędy: 500.

#### 3.2.2 metoda POST

Utworzenie nowej sesji.

Listing 2: przykład zwracanego obiektu JSON

```
1 {
2   "links" : [
3     {
4       "href" : "/sessions/cOPA4JC5Y8xvPhug",
5       "rel" : "self"
6     },
7     {
8       "href" : "/sessions/cOPA4JC5Y8xvPhug/queries",
9       "rel" : "queries"
10    }
11  ]
12 }
```

Możliwe błędy: 500.

---

<sup>1</sup>w terminologii REST

### 3.2.3 metoda OPTIONS

Opis wszystkich elementów interfejsu — skrócona postać tego dokumentu.

Listing 3: przykład zwracanego obiektu JSON

```
1 {
2   "entries" : [
3     {
4       "urlpattern" : "/corpora/<corpus_id>",
5       "method" : "GET",
6       "description" : "Detailed description of attributes of the corpus specified by <corpus_id>
7         identifier"
8       "request_data" : "None",
9     },
10    (...)
11    {
12      "urlpattern" : "/session/<session_id>",
13      "method" : "PUT",
14      "description" : "Update per-session settings. Session identified by <session_id>"
15      "request_data" : "JSON-encoded object with fields \"settings\" and/or \"current_corpus\". The
16        first one contains an object defining new settings values (key : value pairs). The second
17        gives an identifier of a new corpus to be used",
18      "example_request_data" : {
19        "settings" : { "use_index" : false },
20        "current_corpus" : "NKJP_1m",
21      }
22    }
23  ]
24 }
```

Możliwe błędy: 500.

## 3.3 Lista dostępnych korpusów

Wzorzec URL: `/corpora`

### 3.3.1 metoda GET

Lista korpusów wraz z ich opisami czytelnymi dla użytkownika (definiowanymi podczas indeksowania korpusu).

Listing 4: przykład zwracanego obiektu JSON

```
1 [
2   {
3     "desc" : "1-million subcorpus of National Corpus of Polish",
4     "group" : "NKJP",
5     "links" : [
6       "href" : "/corpora/nkjp_1m",
7       "rel" : "corpus info"
8     ],
9     "access" : "restricted",
10    "id" : "nkjp_1m"
11  },
12  (...)
13 ]
```

Uwagi:

- "group": grupa korpusów z identycznymi formatami anotacji (definiowana podczas indeksowania korpusu)
- "access": "public" albo "restricted" — informacja, czy praca z korpusem wymaga uwierzytelnienia

Możliwe błędy: 500.

## 3.4 Informacje o korpusie

Wzorzec URL: `/corpora/<corpus_id>`

### 3.4.1 metoda GET

Lista typów struktur atrybutowych<sup>2</sup>, z których jest zbudowany korpus, wraz ze szczegółowymi informacjami odnośnie atrybutów i dodatkowych cech. Przydatna np. przy implementacji podpowiedzi albo kolorowania składni zapytania w kliencie.

Listing 5: przykład zwracanego obiektu JSON

```
1 [
2   // edge entry
3   {
4     "attrs" : [
5       {
6         "name" : "synh",
7         "opt" : true,
8         "type" : "boolean",
9       }
10    ],
11    "type" : ""
12  },
13  // node entry
14  {
15    "attrs" : [
16      {
17        "fs_types" : [ "morph" ],
18        "name" : "msd",
19        "opt" : false,
20        "type" : "amblis[fs]",
21      },
22      {
23        "name" : "orth",
24        "opt" : false,
25        "type" : "string",
26      }
27    ],
28    "edges" : [ "" ],
29    "is_graph" : "always",
30    "largescale" : "never",
31    "targets" : [ "node" ],
32    "type" : "synw"
33  }
34 ],
35 // attribute-structure entry
36 {
37   "attrs" : [
38     {
39       "name" : "accommodability",
40       "opt" : true,
41       "type" : "enum",
42       "values" : [ "congr", "rec" ]
43     },
44     (...)
45   ],
46   "type" : "morph",
47   "standalone" : false
48 },
49 (...)
50 ]
```

Uwagi:

- Każdy wpis zawiera minimum 2 pola: "type" i "attrs". Pierwsze jest obowiązkowym identyfikatorem *rodzaju* struktury. Drugie zawiera opis atrybutów występujących w strukturach tego *rodzaju* w korpusie. Elementy "attrs":
  - "name": nazwa atrybutu
  - "opt": informacja czy atrybut jest opcjonalny (czy występuje tylko w niektórych strukturach danego rodzaju)
  - "type": rodzaj przechowywanej wartości, jedno z: "boolean", "string", "number", "enum", "list[<podtyp>]", "amblis[<podtyp>]", "fs", "multi" (gdzie <podtyp> to dowolny z powyższych napisów).
  - (atrybut opcjonalny) "fs\_types": *rodzaj* struktur atrybutowych określonych jako "fs" w poprzednim punkcie.

<sup>2</sup>podstawowy element modelu danych w Poliqarpie 2

- (atrybut opcjonalny) "values": lista możliwych wartości typu enumerowanego wskazanego w wartości "type".
- Krawędzie grafu dokumentu mają tylko dwa powyższe atrybuty.
- Węzły grafu dokumentu mają ponadto atrybuty:
  - "is\_graph": czy węzeł jest elementem grafu składni czy grafu innej anotacji (np. LFG), jedno z "always", "opt", "never" (odpowiednio każdy węzeł danego *rodzaju*, niektóre, żaden)
  - "largescale": czy węzeł jest elementem wielkoskalowej segmentacji, jedno z "always", "opt", "never"
  - (atrybut opcjonalny) "edges": lista *rodzajów* krawędzi wychodzących z tego węzła
  - (atrybut opcjonalny) "sec\_edges": lista *rodzajów* krawędzi drugorzędnych wychodzących z tego węzła
  - (atrybut opcjonalny) "targets": lista typów obiektów docelowych krawędzi wychodzących z tego węzła, możliwe wartości: "node", "fs", "literal\_node".
  - (atrybut opcjonalny) "sec\_edges": lista typów obiektów docelowych krawędzi drugorzędnych wychodzących z tego węzła (wartości j.w.)
- Struktury niewęzłowe i niekrawędziowe mają atrybut "standalone", informujący, czy jest to struktura pomocnicza — będąca jedynie techniczną reprezentacją złożonej anotacji (np. interpretacja morfoskładniowa), czy też jest to samodzielna struktura — element dokumentu poza grafem albo jako liść grafu.

Możliwe błędy: [404](#), [500](#).

## 3.5 Zarządzanie sesją

Wzorzec URL: `/sessions/<session_id>`

### 3.5.1 metoda GET

Aktualne ustawienia sesji wraz z opisami i legalnymi wartościami.

Listing 6: przykład zwracanego obiektu JSON

```

1 {
2   "current_corpus" : "nkjp_1m",
3   "settings" : [
4     {
5       "default" : true,
6       "desc" : "Prun large result graphs deleting fragments unrelated to the result",
7       "group" : "General settings",
8       "name" : "prun_result_graphs",
9       "type" : "boolean",
10      "value" : true,
11      "values" : [
12        {
13          "value" : true,
14          "value_desc" : "On"
15        },
16        {
17          "value" : false,
18          "value_desc" : "Off"
19        }
20      ]
21    },
22    (...)
23  ]
24 }
```

Uwagi:

- Zwrócony obiekt zawiera zawsze dwa pola: "current\_corpus" i "settings". Pierwsze zawiera identyfikator aktualnie używanego korpusu, drugie opis ustawień rozmaitych przełączników.

- pozycje "settings" mają określone:
  - domyślną wartość "default",
  - opis czytelny dla człowieka "desc",
  - grupę ustawień (informacyjnie dla użytkownika) "group",
  - nazwę przełącznika "name",
  - typ wartości przełącznika "type" ("boolean" albo "enum"),
  - aktualną wartość "value",
  - listę legalnych wartości wraz z opisami czytelnymi dla człowieka "values".

Możliwe błędy: 404, 500.

### 3.5.2 metoda PUT

Zmień aktualne ustawienia sesji.

**Listing 7:** przykład wysyłanego obiektu JSON

```
1 {
2   "settings" :
3   {
4     "prun_result_graphs" : false
5   }
6 }
```

**Listing 8:** przykład zwracanego obiektu JSON

```
1 {
2   "corpus chosen" : false,
3   "settings updated" : true
4 }
```

Uwagi:

- klient wysyła obiekt JSON z polami "settings" i/lub "current\_corpus", ich znaczenie jest analogiczne do omówionego przy metodzie GET.
- w "settings" klient wysyła same pary *nazwa:wartość* i tylko tych atrybutów, które chce zmienić
- **Zmiana korpusu unieważnia wyniki ostatniego zapytania**

Możliwe błędy: 400, 401, 404, 415, 500, 503.

### 3.5.3 metoda DELETE

Zakończ sesję.

**Listing 9:** przykład zwracanego obiektu JSON

```
1 {
2   "session terminated" : true
3 }
```

Możliwe błędy: 404, 500, 503.

## 3.6 Zapytania w sesji

Wzorzec URL: /sessions/<session\_id>/queries

### 3.6.1 metoda GET

Lista dotychczasowych zapytań w ramach sesji w porządku chronologicznym od najnowszego.

**Listing 10:** przykład zwracanego obiektu JSON

```
1 [
2   {
3     "corpus" : "nkjp_1m",
4     "exec_status" : "not_executed",
5     "id" : "hbaKqVzoiI4e2bc",
6     "query" : "[type=\"seg\" && pos=\"subst\"]",
7     "query_status" : "compiled"
8   },
9   (...)
10 ]
```

Możliwe błędy: 404, 500.

### 3.6.2 metoda POST

Utworzenie nowego zapytania.

**Listing 11:** przykład wysyłanego obiektu JSON

```
1 "[type=\"seg\" && pos=\"subst\"]"
```

**Listing 12:** przykład zwracanego obiektu JSON

```
1 {
2   "id" : "hbaKqVzoiI4e2bc",
3   "links" : [
4     {
5       "href" : "/sessions/jA0Xrip8y0CFT0GJ/queries/hbaKqVzoiI4e2bc",
6       "rel" : "self"
7     }
8   ]
9 }
```

Uwagi:

- Klient przesyła zapytanie w postaci pojedynczego napisu zakodowanego w JSON.
- Spodziewany kod odpowiedzi: 202 (*Accepted*), zapytanie jest parsowane i kompilowane asynchronicznie. Status kompilacji można sprawdzić pod zwróconym adresem.
- **Utworzenie nowego zapytania unieważnia wyniki ostatniego zapytania w aktualnej sesji**

Możliwe błędy: 400, 404, 415, 500, 503.

## 3.7 Zarządzanie zapytaniem

Wzorzec URL: `/sessions/<session_id>/query/<query_id>`

### 3.7.1 metoda GET

Status kompilacji zapytania.

**Listing 13:** przykład zwracanego obiektu JSON

```
1 {
2   "query is ok" : true
3 }
```

Uwagi:

- Odpowiedź 503 oznacza, że silnik jeszcze nie zakończył kompilacji zapytania i należy ponowić żądanie za jakiś czas.
- W przypadku wykrycia błędu składni zapytania kod odpowiedzi to 406, zwrócony zostaje też szczegółowy, czytelny dla człowieka opis błędu w polu "error" (patrz też 4.1.4).

Możliwe błędy: 404, 406, 500, 503.

### 3.7.2 metoda POST

Uruchomienie wykonywania zapytania.

**Listing 14:** przykład zwracanego obiektu JSON

```
1 {  
2   "links" : [  
3     {  
4       "href" : "/sessions/jA0Xrip8y0CFT0GJ/queries/hbacKqVzojI4e2bc/job_status",  
5       "rel" : "job_status"  
6     }  
7   ],  
8   "executing" : true  
9 }
```

Uwagi:

- Postęp w wykonywaniu zapytania można sprawdzić pod zwróconym adresem.

Możliwe błędy: [404](#), [406](#), [500](#), [503](#).

## 3.8 Proces wykonywania zapytania

Wzorzec URL: `/sessions/<session_id>/query/<query_id>/job_status`

### 3.8.1 metoda GET

Postęp wykonania zapytania.

**Listing 15:** przykład zwracanego obiektu JSON

```
1 {  
2   "done" : false,  
3   "found_so_far" : 12,  
4   "links" : [  
5     {  
6       "href" : "/sessions/jA0Xrip8y0CFT0GJ/queries/hbacKqVzojI4e2bc/results/0",  
7       "rel" : "first_result"  
8     }  
9   ],  
10  "percent_done" : 87.31  
11 }
```

Uwagi:

- Jeśli został znaleziony chociaż jeden wynik, tzn. `"found_so_far" > 1`, pole `"links"` zawiera adres dostępu do pierwszego wyniku zapytania. W przeciwnym wypadku to pole nie występuje.
- W przypadku wykrycia błędu wykonania zapytania kod odpowiedzi to 406, zwrócony zostaje też szczegółowy, czytelny dla człowieka opis błędu w polu `"error"` (patrz też [4.1.4](#)).

Możliwe błędy: [404](#), [406](#), [500](#), [503](#).

### 3.8.2 metoda DELETE

Anulowanie wykonania zapytania.

**Listing 16:** przykład zwracanego obiektu JSON

```
1 {  
2   "aborted" : true  
3 }
```

Możliwe błędy: [404](#), [500](#).

## 3.9 Wyniki zapytania

Wzorzec URL: `/sessions/<session_id>/query/<query_id>/results/<result_id>`



### 3.9.1 metoda GET

Pojedynczy wynik zapytania.

Listing 17: przykład zwracanego obiektu JSON

```
1 {
2   "colors" : [
3     { "color" : 1, "var" : "_Q" },
4     (...)
5   ],
6   "id" : 0,
7   "links" : [
8     {
9       "href" : "/sessions/jA0Xrip8y0CFT0GJ/queries/hbacKqVz0jI4e2bc/results/1",
10      "rel" : "next_result"
11    }
12  ],
13  "table_vars" : [ "_Q", "X", "Y" ]
14  "literals" : { "X" : "nie", "Y" : "jest" },
15  "meta" : "",
16  "elements" : [
17    {
18      "attrs" : {},
19      "color" : 0,
20      "id" : 4498001752,
21      "target" : { "ptr" : 4517000007 },
22      "type" : ""
23    },
24    {
25      "attrs" : {
26        "msd" : {
27          "all" : [
28            { "ptr" : 4498010878 },
29            (...)
30          ],
31          "sel" : [ { "ptr" : 4498010878 } ]
32        },
33        "orth" : "Sukces"
34      },
35      "color" : 1,
36      "edges" : [ { "ptr" : 4498001752 } ],
37      "graph" : true,
38      "id" : 4517000002,
39      "sec_edges" : [],
40      "span" : [ 1, 2 ],
41      "type" : "seg"
42    },
43    {
44      "attrs" : {
45        "base" : "sukces",
46        "case" : "acc",
47        "gender" : "m3",
48        "number" : "pl",
49        "pos" : "subst"
50      },
51      "color" : 0,
52      "id" : 4498010878,
53      "standalone" : false,
54      "type" : "morph"
55    },
56    {
57      "color" : 0,
58      "id" : 4517000726,
59      "literal_value" : "common",
60    },
61    (...)
62  ]
63 }
```

Uwagi:

- "colors": lista kolorów/znaczników dla korpusowych elementów wyniku wraz z przyporządkowaniem do nazw zmiennych podanych w zapytaniu,
- "id": numer kolejny wyniku,
- "links": link do następnego wyniku (jeśli taki istnieje),
- "literals": lista wyników wyników spoza grafu dokumentu (np. użytych w postprocessingu) z przypisaniem do zmiennych,

- "table\_vars": lista wszystkich zmiennych występujących w zapytaniu,
- (atrybut opcjonalny) "meta": metainformacje dotyczące wyniku,
- "elements": lista elementów korpusowych składających się na wynik i jego kontekst. W zależności od posiadanych pól reprezentują one:
  - "literal\_value": *liść-literał* — np. w strukturach LFG,
  - "target": krawędź grafu dokumentu,
  - "edges": węzeł grafu dokumentu,
  - "standalone": niewęzłową i niekrawędziową strukturę atrybutową.

Zagnieżdżone obiekty z atrybutami "sel" i "all" reprezentują listy niejednoznaczności. W ramach "elements" wzajemne zagnieżdżenia/połączenia elementów korpusu są zrealizowane obiektami { "\ptr\" : <identyfikator>}. W ich miejsca należy wstawić referencję do elementu o podanym identyfikatorze.

- znaczenie pozostałych pól w zwracanych obiektach zostało ogólnie wyjaśnione w sekcji 3.4.1 albo wyniku wprost z modelu danych korpusu bądź języka zapytań.

Możliwe błędy: 404, 500, 503.

## 4 Odpowiedzi sygnalizujące błąd

### 4.1 Błędy klienta 4XY

#### 4.1.1 400: nieprawidłowe żądanie

Błędna treść żądania, np. błąd parsowania dokumentu JSON.

**Listing 18:** przykład zwracanego obiektu JSON

```
1 {
2   "error" : "* Line 1, Column 2\n Missing '}' or object member name\n"
3 }
```

#### 4.1.2 401: nieautoryzowany dostęp

Zasoby, do których odwołuje się żądanie, wymagają autoryzacji. Taki błąd pojawi się przy próbie dostępu do korpusu oznaczonego jako "access" : "restricted"

**Listing 19:** przykład zwracanego obiektu JSON

```
1 {
2   "error" : "Access to restricted corpus requires authorization"
3 }
```

#### 4.1.3 404: nie znaleziono zasobu

Zasób nie został znaleziony — albo jego identyfikator jest błędny albo zasób został usunięty.

**Listing 20:** przykład zwracanego obiektu JSON

```
1 {
2   "error" : "session not found"
3 }
```

#### 4.1.4 406: niedozwolone żądanie

Żądanie asynchroniczne zakończyło się błędem. Np. wystąpił błąd parsowania.

**Listing 21:** przykład zwracanego obiektu JSON

```
1 {
2   "error":
3   {
4     "details": [
5       {
6         "msg_text": "unable to proceed with parsing",
7         "query_text": "[pos=adj]",
8         "type": "error",
9         "groups": [
10          {
11            "q_start": 9,
12            "q_end": 10
13          }
14        ]
15      }
16    ],
17    "msg": "Parsing error:"
18  },
19  "query is ok": false
20 }
```

Uwagi:

- "error" zawiera informacje o błędzie, w szczególności pole "msg" zawiera ogólny opis błędu. Pola "details" i "additional\_details" są opcjonalne, ich zawartość zależy od wersji i ustawień silnika wyszukiwarki.

#### 4.1.5 415: nieznany format żądania

Błędne żądanie: albo brakuje nagłówka *Content-Type* albo treść żądania nie jest w formacie JSON.

**Listing 22:** przykład zwracanego obiektu JSON

```
1 {
2   "error" : "Unsupported request contents"
3 }
```

## 4.2 Błędy silnika: 5XY

### 4.2.1 500: wewnętrzny błąd silnika wyszukiwarki

Nastąpiła wewnętrzna awaria silnika wyszukiwarki.

### 4.2.2 503: zasób niedostępny

Silnik wyszukiwarki jest zajęty obsługą poprzedniego żądania asynchronicznego albo jest przeciążony z powodu nadmiaru użytkowników. W przypadku, gdy w danym momencie jest wykonywane zapytanie, aby zmienić ustawienia albo zapytanie należy obecny proces wyszukiwania anulować (patrz 3.8.2). Można też ponowić żądanie po jakimś czasie.