ООО «КВАНТУМ АРТ»

115184, Москва, Озерковский переулок, д. 12

тел. (495) 783-65-74

Компонент QP8.Search.API продукта QP8.Search

Документация разработчика

Москва

2023

Назначение документа

Настоящий документ – руководство компонента QP8.Search.API. Документ предназначен для разработчиков, реализующих интеграцию программного обеспечения с компонентом QP8.Search.API.

История изменений

Версия	Дата	Автор	Описание
1.0	22.01.2023	Григорьева М.А.	Создание документа

Оглавление

1. Общие сведения

Компонент QP8.Search.API – это proxy-сервис поиска, обеспечивающий взаимодействие между пользователем (backend или frontend сайта, либо мобильное приложение) и Elasticsearch, валидацию запросов, ограничение поиска на основе ролевой модели QP8.CMS, а также дополнение запроса заранее сконфигурированными настройками (например, весами полей для поиска).

QP8.Search.API состоит из следующих функциональных частей:

- Completion;
- Search;
- Suggest.

Запросы отправляются в виде запросов POST в JSON-формате.

Ответ возвращается в JSON-формате.

2. Метод Completion

Метод Completion предназначен для дополнения поиска по индексам Elasticsearch. Дополнение может осуществляться следующими способами:

- дополнение по одному или нескольким индексам Elasticsearch;
- по средствам объединения нескольких независимых запросов (входные данные для запроса - массив).

При выполнении дополнения допустимы полные имена индексов или использование шаблонов.

Запросы отправляются на адрес:

```
/api/v1/completion
```

Для выполнения запроса требуется указать обязательные параметры:

- $from – индекс, список индексов или шаблон, на основе которого будут выбраны индексы, по которым будет осуществляться поиск;
- $query – текст, по которому необходимо осуществлять поиск.

Пример запроса:

```
{
  "$from": "*",
  "$query": "нов"
}
```

Будет выполнен запрос во все индексы с текстом "нов".

В результате будет возвращён JSON-документ, который содержит в себе "status" (статус выполнения) и "phrases" - массив найденных слов.

Пример ответа:

```
{
  "status": 200,
  "phrases": [
    "новый",
    "новости"
  ]
}
```

3. Метод Search

Метод Search предназначен для выполнения полнотекстового поиска по слову или словам в ранее проиндексированных данных.

При полнотекстовом поиске может быть использован поиск с учётом синонимов, морфологии, стоп-слов и последовательности слов (шинглов). Однако для работы этого функционала необходимо корректно сконфигурировать настройки индексации и подключить соответствующие языковые словари. Подробнее об этом можно узнать в документации разработчика к компоненту QP8.Search.Integration.

Минимально необходимое требование для работы полнотекстового поиска: текстовые поля должны быть проиндексированы с типом "text".

Запрос отправляется на адрес:

```
/api/v1/search
```

Обязательные для заполнения поля:

- $from – индекс, список индексов или шаблон, на основе которого будут выбраны индексы, по которым будет осуществляться поиск;
- $query – текст, по которому необходимо осуществлять поиск.

Пример запроса:

```
{
  "$from": "*",
  "$query": "абонент"
}
```

В результате будет возвращён набор документов:

```
{
  "status": 200,
  "totalCount": 143505,
  "documents": [
    {
      "_id": "12345",
      "_index": "qp.textpages",
      "_score": 16.47304,
      "Title": "Помощь абоненту",
      // ...
    },
    // ...
  ]
}
```

В ответе обязательно присутствуют:

- status – статус выполнения запроса;
- totalCount – общее кол-во найденных документов (может отличаться от фактически вернувшегося кол-ва, если выдача была ограничена параметров $limit);
- documents – список найденных документов
  - _id – системное поле с Id записи;
  - _index – название индекса, в котором был найден документ;
  - _score – мера релевантности документа в рамках запроса, чем больше число, тем релевантнее документ, пот этому полю происходит сортировка поисковой выдачи по умолчанию;
  - * – стальные поля, которые были заданы для возврата в самом запросе или шаблоном поиска.

4. Метод Suggest (поиск по префиксам слов)

Метод Suggest – это метод поиска по префиксам слов, который объединяет логику работы двух предыдущих методов. Метод выполняет поиск по части слова по аналогии с тем, как это делает completion, но вместо выдачи просто списка найденных слов, выдаёт сразу конкретные документы, которые содержат в себе найденные слова, как это делает search.

Запрос отправляется на адрес:

```
/api/v1/suggest
```

Обязательные для заполнения поля:

- $from – индекс, список индексов или шаблон, на основе которого будут выбраны индексы, по которым будет осуществляться поиск;
- $query – текст, по которому необходимо осуществлять поиск.

Пример запроса:

```
{
  "$from": "*",
  "$query": "абон"
}
```

В результате будет возвращён набор документов:

```
{
  "status": 200,
  "totalCount": 143505,
  "documents": [
    {
      "_id": "12345",
      "_index": "qp.textpages",
      "_score": 16.47304,
      "Title": "Помощь абоненту",
      // ...
    },
    // ...
  ]
}
```

В ответе обязательно присутствуют:

- status – статус выполнения запроса;
- totalCount – общее кол-во найденных документов (может отличаться от фактически вернувшегося кол-ва, если выдача была ограничена параметров $limit);
- documents – список найденных документов:
  - _id – системное поле с Id записи;
  - _index – название индекса, в котором был найден документ;
  - _score – мера релевантности документа в рамках запроса, чем больше число, тем релевантнее документ, поэтому полю происходит сортировка поисковой выдачи по умолчанию;
  - * – остальные поля, которые были заданы для возврата в самом запросе или шаблоном поиска.

5. Мультиплексирование поисковых запросов

В случае, если есть необходимость выполнить несколько запросов одного типа, допускается их объединение в массив документов и отправка в соответствующий мультиплексированный метод Api.

Соответствие методов:

Completion – Multi_completion;

Search – Multi_search;

Suggest – Multi_suggest.

Пример мультиплексированного запроса на поиск:

```
[
  {
    "$from": "*",
    "$query": "абонент"
  },
  {
    "$from": "*",
    "$query": "новость"
  }
]
```

6. Дополнительные возможности

6.1 Роли пользователя

В случае, если в рамках проекта присутствуют заранее определенные роли пользователей, и в конфигурации проекта API была заранее включена поддержка ролей (а также включена индексация ролей), то в данном случае имеется возможность добавить роль или список ролей через параметр "$roles" в запросе.

При не пустом параметре ролей Api поиска производит выборку разрешенных индексов, куда можно делать запросы с указанной ролью/ролями, после чего содержимое "$from" будет переопределено исходя из найденных индексов.

Применимо для запросов: completion, search, suggest.

Пример запроса:

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$roles": ["Manager", "Reader"]
}
```

6.2 Фильтрация по полям

В запросе поиска можно указывать дополнительные условия поиска в поле "$where".

Применимо для запросов: completion, search, suggest.

Можно задать условия на одно или несколько различных полей:

```
{
  "$from": "qp.news",
  "$where": {
    "Regions": { "Alias": ["moskva", "spb"] },
    "Groups.Title": "Новости Аборентам"
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

Условия на разные поля документа объединяются через "AND". Условия внутри массива значений одного поля объединяются через "OR". Таким образом, в примере выше мы выбираем все новости, которые имеют регион "moskva" или "spb" и принадлежат к группе с заголовком "Новости Абонентам".

Если поле, для которого задано условие, в исходных документах представлено не скалярным значением, а списком (или является полем одного из объектов в списке), то документ попадет в выдачу, когда условие выполняется хотя бы для одного из элементов этого списка.

Например, данный документ будет удовлетворять условию выше:

```
{
  "Title": "Какая-то новость",
  "Regions": [{ "Alias": "moskva" }, { "Alias": "tula" }],
  "Groups": [
    { "Title": "Новости Абонентам" },
    { "Title": "Новости корпоративным клиентам" }
  ]
}
```

Фильтровать можно по всем полям документа, включая служебные "_id" и "_index".

6.3 Расширенные условия

Если необходимо фильтровать по каким-то более сложным условиям, чем явные значения полей, можно воспользоваться расширенным синтаксисом условий:

```
{
  "$from": "qp.news",
  "$where": {
    "Regions": { "Alias": { "$all": ["moskva", "spb"] } },
    "Groups.Title": { "$ne": "Новости Абонентам" },
    "PublishDate": {
      "$gte": "2018-01-01T00:00:00",
      "$lt": "2019-01-01T00:00:00"
    }
  }
}
```

В примере выше: выбираем все новости, которые имеют регион "moskva" и "spb", не принадлежат к группе "Новости Абонентам" и имеют дату публикации с 2018 по 2019 год.

Каждое расширенное условие на поле представляет собой объект со следующими полями:

- $eq: scalar — равно,
- $ne: scalar — не равно,
- $in: scalar[] — содержит одно из @alias $any,
- $any: scalar[] — содержит одно из @alias $in,
- $all: scalar[] — содержит все из,
- $none: scalar[] — не содержит ни одного из,
- $lt: scalar — меньше,
- $lte: scalar — меньше или равно,
- $gt: scalar — больше,
- $gte: scalar — больше или равно.

Предикаты внутри одного объекта объединяются через "AND". Так же, как и с массивом явных значений, мы можем объединить несколько расширенных условий на одно поле через "OR":

```
{
  "$from": "qp.*",
  "$where": {
    "Regions.Alias": [{ "$eq": null }, { "$in": ["moskva", "spb"] }]
  }
}
```

6.4 Комбинирование условий

Если необходимо задать фильтр, когда условия на одно поле зависят от условий на другое, мы можем воспользоваться булевскими комбинаторами:

- $every: Condition[] — должны быть выполнены все условия из списка,
- $some: Condition[] — должно быть выполнено хотя бы одно условие из списка,
- $not: Condition — условие не должно быть выполнено.

Например, ищем тарифы с ценой до 500 в Калуге или до 1000 в Москве:

```
{
  "$from": "dpc.tariff",
  "$where": {
    "$some": [
      {
        "Regions.Alias": "kaluga",
        "ParametersByAlias.SubscriptionFee.NumValue": { "$lte": 500 }
      },
      {
        "Regions.Alias": "moskva",
        "ParametersByAlias.SubscriptionFee.NumValue": { "$lte": 1000 }
      }
    ]
  }
}
```

Все комбинаторы ($every, $some, $not) могут быть вложены друг в друга произвольным образом:

```
{
  "$from": "dpc.tariff",
  "$where": {
    "$every": [
      {
        "$some": [{ "Regions.Alias": "kaluga" }, { "Regions.Alias": "moskva"
}]
      },
      {
        "$not": {
          "ParametersByAlias.SubscriptionFee.NumValue": { "$gt": 1000 }
        }
      }
    ]
  }
}
```

6.5 Фасетный поиск

Фасетный поиск представляет собой поиск информации по нескольким характеристикам одновременно. Чаще всего, внешне он реализован как набор фильтров. Каждый фильтр связан только с одним свойством информации. Значения фильтра отображает информацию по всем возможным значениям свойства.

Применимо для запросов: search.

Задать фасеты по различным полям можно с помощью поля "$facets".

Например, интервал для даты публикации и 5 наиболее популярных рубрик:

```
{
  "$from": "qp.news",
  "$facets": {
    "PublishDate": "$interval",
    "Rubrics": { "Title": { "$samples": 5 } }
  }
}
```

В результате будет получен объект со значениями фасетов:

```
{
  "status": 200,
  // ...
  "facets": {
    "PublishDate": {
      "interval": { "from": "1999-05-05T00:00:00", "to": "2019-07-
09T17:18:00" }
    },
    "Rubrics.Title": {
      "samples": [
        { "value": "Домашний Интернет и ТВ", "count": 8649 },
        { "value": "Спецпредложения", "count": 4495 },
        { "value": "Услуги мобильной связи", "count": 4077 },
        { "value": "Тарифы и скидки на звонки", "count": 4031 },
        { "value": "Обсуживание абонентов", "count": 3597 }
      ]
    }
  }
}
```

Поля вложенных объектов можно объявлять, как через точку, так и во вложенной форме.

Но в результатах названия фасетов будут всегда представлять собой полные пути к вложенному полю через точку.

Фасеты строятся вместе с выполнением поиска и фильтрации по всему диапазону найденных документов.

Таким образом, диапазон построения фасетов ограничивается строкой запроса "$query" и фильтром "$where".

Если при выполнении запроса нам нужны только фасеты (а не найденные документы), то мы можем установить поле "$limit: 0".

```
{
  "$from": "qp.news",
  "$where": {
    "Regions": { "Alias": ["moskva", "spb"] }
  },
  "$limit": 0,
  "$facets": {
    // ...
  }
}
```

Доступны четыре типа фасетов: Interval, Samples, Ranges и Percentiles. Результатом построения каждого фасета является объект с одним полем, где имя поля — это тип фасета, а значение — это результаты.

6.5.1 Interval Facet

Interval Facet служит для определения минимального и максимального значения поля в выборке и задается с помощью ключевого слова: "$interval":

```
{
  "$from": "qp.news",
  "$facets": {
    "PublishDate": "$interval"
  }
}
```

Результатом является объект с полями "from" и "to":

```
{
  "facets": {
    "PublishDate": {
      "interval": { "from": "1999-05-05T00:00:00", "to": "2019-07-
09T17:18:00" }
    }
  }
}
```

6.5.2 Samples Facet

Samples Facet служит для нахождения наиболее популярных значений поля в выборке.

Задается как объект:

```
{"$samples": <count>}
```

где "<count>" — максимальное количество популярных значений.

Или в сокращенной форме, как ключевое слово "$samples" (тогда <count> = 100).

```
{
"$from": "qp.news",
  "$facets": {
    "Rubrics.Title": { "$samples": 2 }
  }
}
```

Результатом является массив объектов с полями "value" (значение поля) и "count" (количество документов, соответствующее этому значению):

```
{
  "facets": {
    "Rubrics.Title": {
      "samples": [
        { "value": "Домашний Интернет и ТВ", "count": 8649 },
        { "value": "Спецпредложения", "count": 4495 }
      ]
    }
  }
}
```

Если мы ищем сразу по нескольким индексам, можно построить фасет по спец. полю "_index":

```
{
  "$from": "media.*",
  "$facets": {
    "_index": "$samples"
  }
}
```

В результате мы получим распределение документов по индексам:

```
{
  "facets": {
    "_index": {
      "samples": [
        { "value": "media.materials", "count": 2115 },
        { "value": "media.menuandpages", "count": 35 }
      ]
    }
  }
}
```

6.5.3 Ranges Facet

Ranges Facet позволяет разбить значения поля на именованные диапазоны и задается как объект, содержащий список диапазонов:

```
{
  "$ranges": [
    { "$name": "<range_name>", "$from": "<min_value>", "$to": "<max_value>" }
    // ...
  ]
}
```

Где "$name" - имя диапазона, "$from"- нижняя граница (включая указанное значение), "to" — верхняя граница (не включая указанное значение).

Например:

```
{
  "$from": "dpc.internettariff",
  "$facets": {
    "ParametersByAlias.MaxSpeed.NumValue": {
      "$ranges": [
        { "$name": "high", "$from": 101 },
        { "$name": "mid", "$from": 30, "$to": 101 },
        { "$name": "low", "$to": 30 }
      ]
    }
  }
}
```

Результатом является массив объектов с полями "name" (имя диапазона), "from", "to" (границы диапазона) и "count"- кол-во документов, попавшее в диапазон:

```
{
  "facets": {
    "ParametersByAlias.MaxSpeed.NumValue": {
      "ranges": [
        { "name": "low", "to": 30, "count": 99 },
        { "name": "mid", "from": 30, "to": 101, "count": 165 },
        { "name": "high", "from": 101, "count": 11 }
      ]
    }
  }
}
```

6.5.4 Percentiles Facet

Percentiles Facet служит для построения доверительных интервалов и медианных значений и задается как объект:

```
{"$percentiles": number[]}
```

содержит список процентных значений.

Например, доверительный интервал 5-95 %:

```
{
  "$from": "dpc.device",
  "$limit": 0,
  "$facets": {
    "ParametersByAlias.SalePrice.NumValue": {
      "$percentiles": [5, 95]
    }
  }
}
```

Результатом является массив объектов с полями "percent" (процент распределения вероятностей) и "value" (соответствующее значение поля):

```
{
  "facets": {
    "ParametersByAlias.SalePrice.NumValue": {
      "percentiles": [
        { "percent": 5, "value": 1300 },
        { "percent": 95, "value": 6500 }
      ]
    }
  }
}
```

Другие примеры Percentiles Facet:

Медиана:

```
{ "$percentiles": [50] }
```

Минимум и максимум:

```
{ "$percentiles": [0, 100] }
```

Уровни доверия в процентах:

```
{ "$percentiles": [90, 99, 99.9, 99.99] }
```

6.6 Контекстные поля

Поле документа является контекстным, если при разных условиях фильтрации для одного и того же документа Elasticsearch необходимо выдавать разные значения этого поля.

Применимо для запросов: search, suggest.

Запрос для поиска документов также имеет поле "$context", в котором указан фильтр для контекстных полей документа. Если этот фильтр явно отсутствует, его значение берется из поля "$where".

Пример: поле "SearchUrl", содержащее Url страницы, который начинается с поддомена. Поддомен в свою очередь зависит от одного из регионов документа, хранящихся в массиве "Regions".

Для этого при индексации в контекстное поле добавляется массив объектов, каждый из которых содержит единственное значение контекстного поля, а также выбранные значения тех полей документа, по которым должна проходить контекстная фильтрация.

Пример:

```
{
  "SearchUrl": [
    {
      "SearchUrl": "http://moskva.domain.ru",
      "Regions": { "Id": 123, "Alias": "moskva" }
    },
    {
      "SearchUrl": "http://spb.domain.ru",
      "Regions": { "Id": 456, "Alias": "spb" }
    }
  ],
  "Regions": [{ "Id": 123, "Alias": "moskva" }, { "Id": 456, "Alias": "spb"
}]
  // ... other fields
}
```

Таким образом в каждом объекте контекстного массива ОБЯЗАТЕЛЬНО содержится поле с тем же названием, что и у исходного контекстного поля, плюс поля документа, от которых она зависит.

Далее для всех имен полей из фильтра "$context" или "$where", если такое поле содержится в объектах контекстного массива, то по этому полю применяется фильтрация на application-сервере, после загрузки документа из Elasticsearch.

Если в фильтре не указано ни одного подходящего поля, то будет выбран первый попавшийся объект контекстного массива.

Пример: если в фильтре указано:

```
{
  "$where": {
    "Regions.Alias": "moskva",
    "Tags": ["foo", "bar"]
  }
}
```

То фильтрация будет производиться только по "Regions.Alias", т.к. "Tags" не содержится в контекстных объектах.

После нахождения единственного контекстного объекта из него выбирается значение контекстного поля.

Результат:

```
{
  "SearchUrl": "http://moskva.domain.ru",
  "Regions": [{ "Id": 123, "Alias": "moskva" }, { "Id": 456, "Alias": "spb"
}]
  // ... other fields
}
```

Контекстная фильтрация полей поддерживает все выражения кроме булевских комбинаторов "$every", "$some", "$not". Эти комбинаторы будут проигнорированы при поиске значения контекстного поля.

6.7 Веса полей

Поиск по ElasticSearch поддерживает поиск по полям с учётом веса поля. Задавая веса полей, можно повысить приоритет поиска для определенных полей.

Применимо для запросов: completion, search, suggest.

Для этого нужно указать веса различных полей документа в поле "$weights":

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$weights": {
    "HeaderTitle": 5,
    "MainTag.Title": 10,
    "Tags": { "Title": 2 }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

6.8 Минимальное кол-во найденных слов

С помощью поля "$requiredWordsCount" можно указать минимальное количество найденных слов из "$query", при котором документ попадает в выдачу. По умолчанию необходимы ВСЕ слова.

Например:

- 3 — должны быть найдены не менее трех слова;
- 1 — должны быть найдены все слова кроме одного;
- "80%" — должны быть найдены не менее 80% слов.

Применимо для запросов: search, suggest.

Каждый возвращаемый документ содержит служебные поля, описанные в формате ответа для каждого запроса, а также остальные поля, которые можно выбрать с помощью поля "$select" (по умолчанию возвращаются все поля):

```
{
"$from": "media.materials",
  "$select": ["Id", "Content", "Tags.Title", "Category.*"]
}
```

Допустимы имена полей вложенных объектов через точку или wildcard-паттерны.

6.9 Подсвечивание HTML-фрагментов

Применимо для запросов: search, suggest.

С помощью поля "$snippets" можно указать, по каким полям документа нужно сгенерировать подсвеченные сниппеты, где "$count"- количество сниппетов (default 5), а "$length" – максимальная длина (default 100):

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": {
    "HeaderTitle": { "$count": 1, "$length": 100 },
    "Tags": { "Title": { "$count": 2, "$length": 50 } }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

В результате полученные сниппеты будут добавлены в каждый документ в спец. поле "_snippets":

```
{
  // ...
  "documents": [
    {
      "_id": "101391",
      "_snippets": { "HeaderTitle": ["<b>мобильный</b> телефон"] }
    }
    // ...
  ]
}
```

Также можно указать "$count" в сокращенной форме, в виде числа:

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": { "HeaderTitle": 1, "Tags": 2 }
}
```

Можно не указывать конкретного поля для сниппетов,

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": { "$count": 1, "$length": 100 }
}
```

Тогда они будут построены по объединению полей "_all":

```
{
  // ...
  "documents": [
    {
      "_id": "101391",
      "_snippets": { "_all": ["<b>мобильный</b> телефон"] }
    }
    // ...
  ]
}
```

Для того чтобы поле не разбивалось на несколько сниппетов, нужно задать "$count: 0":

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": { "HeaderTitle": { "$count": 0 } }
}
```

Тогда сниппет будет построен по всему полю целиком.

6.10 Сортировка

Применимо для запросов: search, suggest.

Сортировка результатов задается в поле "$orderBy":

- в виде имени поля "PublishDate";
- объекта с указанием направления сортировки:

```
{ "PublishDate": "desc" }
```

- или набора из нескольких полей:

```
["Id", { "PublishDate": "desc" }]
```

Также можно сортировать по спец. полю "_score", которое соответствует релевантности документа

```
{
  "$from": "media.materials",
  "$query": "мобильыне приложения",
  "$orderBy": [{ "PublishDate": "desc" }, "_score"]
}
```

По умолчанию выдача сортируется по "_score"

6.11 Постраничный вывод

Применимо для запросов: search.

Ограничить размер выдачи можно с помощью полей "$limit" – размер страницы (по умолчанию 50) и "$offset" – отступ от начала выдачи:

```
{
  "$from": "media.materials",
  "$limit": 10,
  "$offset": 20
}
```

6.12 Размер выдачи

Применимо для запросов: completion, suggest.

Ограничить размер выдачи можно с помощью полей "$limit" – размер страницы (по умолчанию 50):

```
{
  "$from": "media.materials",
  "$limit": 10
}
```

6.13 Исправление поискового ввода

Применимо для запросов: search.

Если при поиске не найдено ни одного результата (или найдено мало), то поисковая система может предложить исправление запроса. Чтобы включить эту функциональность нужно задать условия в поле "$correct":

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$correct": {
    "$query": { "$ifFoundLte": 5 },
    "$results": { "$ifFoundLte": 2 }
  }
}
```

В данном примере — предложить пользователю исправление запроса, если найдено не более 5 документов. Применять это исправление при поиске результатов, если изначально было найдено не более 2 документов.

В этом случае в ответе API будет присутствовать поле "queryCorrection."

```
{
  // ...
  "queryCorrection": {
    "text": "мобильные приложения",
    "snippet": "<b>мобильные</b> приложения",
    "resultsAreCorrected": true
  }
  // ...
}
```

Флаг "resultsAreCorrected" указывает на то, что исправленный запрос уже был применен при поиске результатов.