ООО «КВАНТУМ АРТ»

115184, Москва, Озерковский переулок, д. 12
тел. (495) 783-65-74

Поисковая система QP8.Search

Руководство по установке

Москва
2024

1 Оглавление

1. ЗАВИСИМЫЕ КОМПОНЕНТЫ 3

2. НАСТРОЙКА БАЗЫ ДАННЫХ 3

3. УСТАНОВКА С ИСПОЛЬЗОВАНИЕМ DOCKER 3

3.1. Заполнение compose-файла 4

4. УСТАНОВКА НА LINUX 4

4.1. Требования к окружению Linux 4
4.2. Скачивание продукта QP8.Search. 4
4.3. Создание пользователя и группы. 4
4.4. Распаковка архива с продуктом QP8.Search 5
4.5. Установка компонента QP8.Search.Integration 5
4.5.1 Создание директории для log-файлов 5
4.5.2 Настройка компонента QP8.Search.Integration 5
4.5.3 Создание service-файла для systemd 10
4.5.4 Добавление настройки индексации в "Application Settings" 11
4.5.5 Запуск компонента 12
4.5.6 Автоматический запуск компонента при старте ОС 12
4.6. Установка компонента QP8.Search.Admin 12
4.6.1. Создание директории для log-файлов 12
4.6.2. Настройка компонента QP8.Search.Admin 12
4.6.3. Создание service файла для systemd 15
4.6.4. Запуск компонента 16
4.6.5. Автоматический запуск компонента при старте ОС 17
4.7. Установка компонента QP8.Search.Api 17
4.7.1. Создание директории для log-файлов 17
4.7.2. Настройка компонента QP8.Search.Api 17
4.7.3. Создание service файла для system 18
4.7.4. Запуск компонента 19
4.7.5. Автоматический запуск компонента при старте ОС 20

5. УСТАНОВКА С ИСПОЛЬЗОВАНИЕМ KUBERNETES 20

5.1. Заполнение манифестов 20
5.2. Применение манифестов 20

1. Зависимые компоненты

Работа QP8.Search зависит от следующих компонентов:

- ElasticSearch версии 8 (документацию по установке можно найти на официальном сайте) или OpenSearch версии 2.8 (документацию по установке можно найти на официальном сайте);
- QP8.WidgetPlatform (документацию по установке можно найти на сайте QuantumArt);
- PostgreSQL 13+ (допускается использование того же экземпляра, что используется в QP8.WidgetPlatform).

2. Настройка базы данных

Для работы компонента Admin продукта QP8.Search требуется предварительно настроить базу данных для внутренней авторизации и управления.

Для этого скачиваем файл по ссылке. В файле создаётся две базы данных, два пользователя, а также прописываются права на базу созданным пользователям.

Редактируем в файле имена баз данных, имена пользователей и пароли пользователей на желаемые, после чего выполняем скрипт.

Пример команды на выполнение скрипта:

```
sudo psql -h 127.0.0.1 -p 5432 -U postgres -d postgres -a -f create-dbs.sql
```

Разберём запрос по параметрам:

- -h – DNS-имя или IP адрес сервера, на котором работает PostgreSQL;
- -p – номер порта, который слушает PostgreSQL;
- -U – имя пользователя, с которым будет производиться подключение (у пользователя должны быть права администратора БД);
- -d – название базы данных (должна существовать);
- -a – флаг, отвечающий за вывод в консоль всех выполняемых команд;
- -f – путь до SQL-файла, который требуется выполнить.

3. Установка с использованием Docker

Для установки QP8.Search с использованием Docker требуются:

- Docker;
- Docker-compose.

Всего будет развёрнуто 3 контейнера:

- Search-api – web-api компонента поиска;
- Search-admin – компонент администрирования (web-приложение);
- Search-integration – компонент индексации данных.

Шаги необходимые для разворачивания:

- Скачать docker-compose файл;
- Заполнить compose-файл (подробнее в пункте 3.1);
- Выполнить команду `docker-compose up -d`

3.1. Заполнение compose-файла

В compose-файле меняем следующие параметры:

- ElasticSettings__Address – адрес до кластера ElasticSearch/OpenSearch;
- ElasticSettings__ProjectName – имя проекта (в примере используется "demosite" для проекта демонстрационного сайта);
- ContextConfiguration__ConnectionString – строка подключения к БД, на которой работает сайт и виджетная плфторма (QP8.WidgetPlatform);
- ContextConfiguration__SqlServerType – тип используемой БД (PostgreSql или SqlServer);
- ContextConfiguration__DefaultSchemeName – схема, в которой находятся объекты виджетной платформы;
- ContextConfiguration__ContentAccess – режим доступа к данным (Live или Stage);
- ContextConfiguration__FormatTableName – формат имени таблицы (0 – схема, 1 – имя таблицы);
- Settings__AdminAppUrl – адрес, по которому доступен компонент Admin продукта QP8.Search;
- ConnectionStrings__AdminSearchDbContextConnection – строка подключения к первой БД, созданной в пункте 2;
- ConnectionStrings__CrawlerSearchDbContextConnection – строка подключения ко второй БД, созданной в пункте 2.

4. Установка на Linux

4.1. Требования к окружению Linux

- ОС Linux с версией ядра 4.9 или новее;
- Systemd;
- ASP.NET Core Runtime (в версии под .NET 6);
- Sudo;
- Nano;
- Tar.

4.2. Скачивание продукта QP8.Search.

Продукт доступен к загрузке на сайте QuantumArt по ссылке.

4.3. Создание пользователя и группы.

Для запуска компонентов продукта QP8.Search рекомендуется создать отдельную группу и пользователя в системе.

В качестве примера будем использовать имя пользователя и группы quantumart.

Создадим группу следующей командой:

```
sudo groupadd quantumart
```

После чего создадим пользователя и добавим его в созданную ранее группу:

```
sudo useradd quantumart -g quantumart -m
```

Зададим пароль созданному пользователю:

```
sudo passwd quantumart
```

4.4. Распаковка архива с продуктом QP8.Search

Разархивируем скачанный ранее архив в домашнюю директорию созданного ранее пользователя командой:

```
sudo tar -xf /path/to/archive -C /home/quantumart/
```

После распаковки выполнив команду `sudo ls /home/quantumart/` можно увидеть три новых директории:

- QP8.Search.Admin – компонент веб-интерфейса администратора;
- QP8.Search.Api – компонент api-поиска;
- QP8.Search.Integration – компонент сервиса индексации данных из QP в ElasticSearch/OpenSearch.

Выдадим пользователю quantumart права на владение всеми тремя директориями командами:

```
sudo chown quantumart:quantumart /home/quantumart/QP.Search.Admin
sudo chown quantumart:quantumart /home/quantumart/QP.Search.Api
sudo chown quantumart:quantumart /home/quantumart/QP.Search.Integration
```

4.5. Установка компонента QP8.Search.Integration

4.5.1 Создание директории для log-файлов

Создадим директорию куда будут сохраняться log-файлы компонента QP8.Search.Integration командой:

```
sudo mkdir /var/log/search-integration
```

После чего выдадим права владения этой директорией пользователю quantumart командой:

```
sudo chown quantumart:quantumart /var/log/search-integration
```

4.5.2 Настройка компонента QP8.Search.Integration

Откроем на редактирование файл конфигурации компонента командой:

```
sudo nano /home/quantumart/QP.Search.Integration/appsettings.json
```

Параметр IndexPermissions отвечает за необходимость индексировать доступы к контентам на основе ролевой модели QP8 CMS. Указываем true если в проекте планируется использование индексации на основе ролевой модели QP8 CMS, в противном случае оставляем false.

Параметр IndexerLibraries содержит массив имён библиотек индексации, описывающих логику индексации данных из QP8 CMS. Прописываем туда названия библиотек индексации, которые планируется использовать на проекте.

В качестве примера будем использовать стандартную библиотеку индексации Демо-сайта под названием QA.Search.Integration.Demosite.

Пример корректного заполнения параметра:

```
"IndexerLibraries":
[
 "QA.Search.Integration.Demosite"
```

```
]
```

Секция ElasticSettings описывает настройки ElasticSearch/OpenSearch и содержит следующие параметры:

- Address – адрес сервера ElasticSearch/OpenSearch;
- ProjectName – короткое название проекта латиницей;
- RequestTimeout – время ожидания ответа от ElasticSearch/OpenSearch.

Внимание: данная секция должна совпадать с одноимённой секцией, используемой при настройке компонентов QP8.Search.Admin и QP8.Search.Api.

Пример корректного заполнения секции:

```
"ElasticSettings":
{
 "Address": "http://127.0.0.1:9200",
 "ProjectName": "demosite",
 "RequestTimeout": "00:10:00.000"
}
```

Секция Settings.QP описывает правила полной переиндексации контентов и статей QP8 CMS. В единственном параметре CronSchedule указываем частоту, с которой будет запускаться полная переиндексация всех контентов (создание нового индекса в ElasticSearch/OpenSearch, заполнение его данными из QP8 CMS и удаление старого индекса) в формате CronTab.

Создать расписание в формате CronTab можно, например на этом сайте.

Для примера будем использовать запуск каждую десятую минуту.

Пример корректного заполнения секции:

```
"Settings.QP":
{
 "CronSchedule": "*/10 * * * *"
}
```

Внимание: полная переиндексация является достаточно долгой и тяжелой операцией, запускать её желательно как можно реже.

Секция Settings.QP.Update описывает правила обновления и добавления контентов и статей QP8 CMS с момента последнего обновления или полной индексации. В единственном параметре CronSchedule указываем частоту, с которой будет запускаться обновление данных в индексе ElasticSearch (добавление в существующий индекс ElasticSearch/OpenSearch новых данных и обновление изменившихся данных в QP8 CMS) в формате CronTab.

Создать расписание в формате CronTab можно, например на этом сайте.

Для примера будем использовать запуск каждую пятую минуту.

Пример корректного заполнения секции:

```
"Settings.QP.Update":
{
 "CronSchedule": "*/5 * * * *"
```

```
}
```

Внимание: Обновление должно запускаться не реже полной переиндексации.

Секция Settings.QP.Permissions описывает правила индексации прав доступа согласно ролевой модели QP8 CMS. В единственном параметре CronSchedule указываем частоту, с которой будет запускаться переиндексация индекса с правами на контенты в формате CronTab.

Создать расписание в формате CronTab можно, например на этом сайте.

Для примера будем использовать запуск каждую десятую минуту.

Пример корректного заполнения секции:

```
"Settings.QP.Permissions":
{
 "CronSchedule": "*/10 * * * *"
}
```

Внимание: при выключенной настройке IndexPermissions - эта настройка не используется.

Секция PermissionsConfiguration описывает настройки индексации ролевой модели QP8 CMS и содержит следующие параметры:

- DefaultRoleAlias – стандартное имя роли, используемой для публичных разделов;
- PermissionIndexName – название индекса ElasticSearch/OpenSearch куда индексируются права доступа к контентам QP8 CMS;
- QPAbstractItems – список алиасов разделов в QP8 CMS для которых необходимо индексировать права доступа. Массив строк.

Пример корректного заполнения секции:

```
"PermissionsConfiguration":
{
 "DefaultRoleAlias": "Reader",
 "PermissionIndexName": "Permissions",
 "QPAbstractItems":
 [
  "corporate_releases",
  "technology_news",
  "anticorruption_policies"
 ]
}
```

Внимание: значения параметров DefaultRoleAlias и PermissionIndexName должны совпадать с параметрами DefaultReaderRole и PermissionsIndexName соответственно, используемыми при настройке компонента QP8.Search.Api.

Секция ViewOptions описывает кол-во данных, которые нужно выбирать при чтении из QP8 CMS при индексации за один проход. Настройка влияет на потребляемый объём памяти и скорость индексации.

Чем больше указано значение, тем больше будет потребляться памяти компонентом во время индексации и тем дольше будет ElasticSearch/OpenSearch выполнять сохранение данных в индекс, но общее количество проходов будет меньше.

Для примера если в контенте QP8 CMS содержится 1 000 статей, и указан размер в 10 статей, то будет выполнено 100 проходов. Если же указать размер в 100, то будет выполнено 10 проходов.

Секция содержит как настройку размера пачки по умолчанию DefaultBatchSize, так и возможность указать индивидуальный размер для каждого индексируемого контента QP8 CMS.

При указании индивидуального размера пачки размер указывается в параметре BatchSize, а он, в свою очередь, помещается в параметр с именем View который отвечает за индексацию контента в подключаемой библиотеке индексации, имя которой указано в параметре IndexerLibraries.

Пример корректного заполнения этой секции для проекта индексатора демо-сайта:

```
"ViewOptions":
{
 "DefaultBatchSize": 10,
 "ViewParameters":
 {
  "NewsPostView":
  {
  "BatchSize": 10
  },
  "TextPageExtensionView":
  {
  "BatchSize": 10
  }
 }
}
```

В секции ContextConfiguration описываются параметры подключения к БД, в которой QP8 CMS хранит данные. В секции присутствуют следующие параметры:

- ConnectionString – строка подключения к БД, где QP8 CMS хранит данные. Требуется учётная
- запись с правами на чтение из таблиц и представлений;
- SqlServerType – тип используемой БД, может быть PostgreSQL или MSSQL;
- DefaultSchemeName – имя схемы, которая используется в БД QP8 CMS;
- ContentAccess – Live или Stage в зависимости от того, используется QP8 CMS в качестве Live
- окружения или Stage окружения;
- FormatTableName – формат именования таблиц в БД.

Пример корректного заполнения секции:

```
"ContextConfiguration":
{
 "ConnectionString": "Server=127.0.0.1:5432;Database=qa_demosite_rus;User
Id=user;Password=password;",
 "SqlServerType": "PostgreSQL",
 "DefaultSchemeName": "public",
 "ContentAccess": "Live",
 "FormatTableName": "{0}.{1}"
}
```

Не упомянутые выше параметры конфигурации изменять не рекомендуется.

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

Откроем файл конфигурации log-файлов командой:

```
sudo nano /home/quantumart/QP.Search.Integration/NLog.config
```

Найдём строку, содержащую текст logDirectory и в ней у параметра value заменим содержимое на путь к директории log-файлов, которую создали ранее.

Должно получиться так:

```
<variable name="logDirectory" value="/var/log/search-integration"/>
```

Далее в строке

```
<logger name="*" writeTo="console" minlevel="Info" final="true" />
```

следует заменить "console" на "bufferingWrapper_logger".

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

Словари морфологии

Если в рамках проекта требуется в поиске учитывать семантику, синонимы и прочие особенности слова как единицы языка, то требуется подключить словари hunspell в сам ElasticSearch/OpenSearch.

Подключение словаря

Словари можно найти в репозитории.

Скачиваем директории с необходимыми в проекте словарями из репозитория и помещаем их на сервере, где установлен ElasticSearch/OpenSearch (выбор директории зависит от варианта подключения).

Добавление фильтров словарей

Для начала использования словаря требуется создать фильтр с подключенным словарём, для чего добавим в «filters» следующий блок (пример для русского языка):

```
"hunspell_russian":
{
    "locale": "ru",
```

```
    "type": "hunspell",
    "dedup": "true"
}
```

Где «locale» должно соответствовать названию директории, содержащей файлы словаря, «type» следует указать «hunspell» (для Elasticsearch/OpenSearch – это подключаемый словарь, наличие словаря будет проверено в момент создания шаблона), «dedup» указывает на необходимость удаления дубликатов из результатов работы анализатора (после того как по всему тексту индексируемого поля проанализирована семантика и построены синонимы).

Использование фильтров словарей

Для использования словаря в анализаторах требуется указать название фильтра с нужным словарём в списке фильтров анализатора, например, следующим образом:

```
"analyzer_text":
    {
    "filter":
    [
        "lowercase",
        "stopwords_russian",
        "stopwords_english",
        "hunspell_russian",
        "hunspell_english"
    ],
    "type": "custom",
    "tokenizer": "unicode_words"
}
```

4.5.3 Создание service-файла для systemd

Для работы с компонентом как с сервисом с возможностью управления и автоматического запуска создаём файл с именем search-integration.service в директории /usr/lib/systemd/system/ командой:

```
sudo touch /usr/lib/systemd/system/search-integration.service
```

После чего открываем файл на редактирование командой:

```
sudo nano /usr/lib/systemd/system/search-integration.service
```

И заполняем его следующим содержимым:

```
[Unit]
Description=QP Search Integration Service
After=network.service
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=quantumart
WorkingDirectory=/home/quantumart/QP.Search.Integration/
ExecStart=dotnet
/home/quantumart/QP.Search.Integration/QA.Search.Generic.Integration.API.dll --
urls http://*:5500
[Install]
WantedBy=multi-user.target
```

В этом файле нас интересует:

- Description – понятное описание сервиса;
- After – указание на то, после запуска какого сервиса нужно запускать этот;
- Restart – политика автоматического перезапуска;
- RestartSec – пауза между перезапуском;
- User – имя пользователя, от которого будет запущен сервис;
- WorkingDirectory – директория, где лежат все файлы приложения;
- ExecStart – команда, выполняемая для запуска сервиса;
    - --urls – указание под каким DNS-именем или IP-адресом разрешено подключаться к компоненту и на каком порту компонент будет ожидать подключения.

Подробно узнать, как настраивать systemd сервисы можно тут.

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

4.5.4 Добавление настройки индексации в "Application Settings"

Для добавления настройки индексации необходимо в настройках QP (Рисунок 1) добавить новый параметр в блоке «Прочие настройки» (Рисунок 2):

- Ключ (Key): SEARCH_DEFAULT_HOST
- Значение (Value): http://адрес_индексируемого_сайта

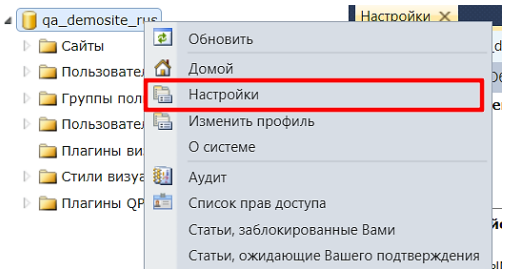

Рисунок 1. Контекстное меню - Настройки

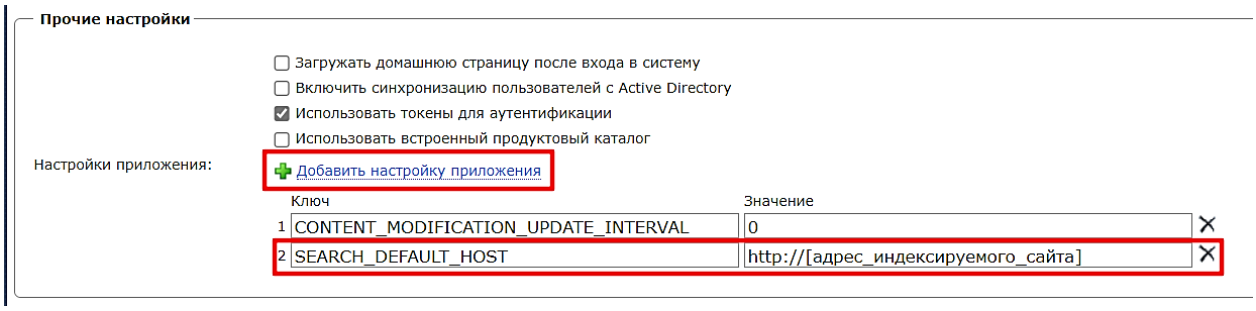

Рисунок 2. Добавление настройки индексации

Внимание: для применения шаблона индекса требуется выполнение скрипта (см. «Шаблон индекса и скрипты его применения»):

- demosite_template.json – шаблон индекса
- apply_template.sh – скрипт применения шаблона в Linux
- apply_template.bat – скрипт применения шаблона в Windows

4.5.5 Запуск компонента

Для запуска компонента выполняем команду:

```
sudo systemctl start search-integration.service
```

После чего проверяем что компонент запустился без ошибок командой:

```
sudo systemctl status search-integration.service
```

В случае успешного запуска в строке Active будет указано active (running), дата и время, когда сервис был запущен, а также сколько прошло времени с момента запуска.

Если при запуске компонента что-то пошло не так и у сервиса статус не active, то вероятнее всего при конфигурировании была допущена ошибка. Проверить в чём ошибка можно в log-файлах компонента.

4.5.6 Автоматический запуск компонента при старте ОС

Если сервис успешно запустился, то можно добавить его в автоматический запуск после загрузки ОС.

Для этого выполним команду:

```
sudo systemctl enable search-integration.service
```

4.6. Установка компонента QP8.Search.Admin

4.6.1. Создание директории для log-файлов

Создадим директорию куда будут сохраняться log-файлы компонента QP8.Search.Admin командой:

```
sudo mkdir /var/log/search-admin
```

После чего выдадим права владения этой директорией пользователю quantumart командой:

```
sudo chown quantumart:quantumart /var/log/search-admin
```

4.6.2. Настройка компонента QP8.Search.Admin

Откроем на редактирование файл конфигурации компонента командой:

```
sudo nano /home/quantumart/QP.Search.Admin/appsettings.json
```

В секции ConnectionStrings присутствуют две строки подключения к базам данных, созданным на предыдущем шаге. Заполняем их актуальными данными, использованными в скрипте создания. Пример корректного заполнения секции:

```
"ConnectionStrings": {
    "AdminSearchDbContextConnection":
   "Server=127.0.0.1:5432;Database=qa_search_admin;User
   Id=qa_search_admin_user;Password=StrongPass1234;",
    "CrawlerSearchDbContextConnection":
   "Server=127.0.0.1:5432;Database=qa_search_crawler;User
   Id=qa_search_crawler_user;Password=StrongPass1234;"
   }
```

Секция Settings описывает общие параметры компонента, тут следует актуализировать только параметр AdminAppUrl в котором указать DNS-имя или адрес сервера, на котором работает компонент QP8.Search.Admin, этот адрес используется для формирования ссылки на восстановление пароля.

Пример: `"AdminAppUrl": "http://127.0.0.1:5600"`

Секция SmtpServiceSettings отвечает за настройки подключения к SMTP-серверу для отправки писем восстановления пароля доступа к панели администратора и содержит следующие параметры:

- Host – адрес smtp-сервера;
- Port – порт smtp-сервера;
- From – адрес электронной почты, с которой будет происходить отправка писем для восстановления;
- DisplayName – имя пользователя, подставляемое к адресу отправителя;
- UseDefaultCredentials – если true, то не используется авторизация по логину и паролю (SMTPсервер должен быть настроен для принятия запросов без авторизации);
- EnableSsl – использовать ли ssl для установки подключения;
- User – имя пользователя для аутентификации на почтовом сервере (используется если параметр UseDefaultCredentials установлен в false);
- Password – пароль для аутентификации на почтовом сервере (используется если параметр UseDefaultCredentials установлен в false).

Пример заполнения секции:

```
"SmtpServiceSettings": {
    "Host": "smtp.yandex.ru",
    "Port": 465,
    "From": "qp-search-admin@yandex.ru",
    "DisplayName": "Search Admin App",
    "UseDefaultCredentials": false,
    "EnableSsl": true,
    "User": "qp-search-admin@yandex.ru",
    "Password": "super_strong_password"
   }
```

Секция IndexingApiServiceConfiguration содержит настройки подключения к компоненту QP8.Search.Integration.

Параметры, которые требуется поменять:

- Scheme – http-схема которая используется для подключения к API (http или https);
- Host – DNS-имя или IP-адрес сервера, на котором работает компонент индексации;
- Port – номер порта, который слушает компонент индексации;
- ProxyAddress – адрес proxy-сервера, если используется (если не используется - оставить пустым).

Пример корректно заполнения секции:

```
"IndexingApiServiceConfiguration": {
    "Scheme": "http",
    "Host": "127.0.0.1",
    "Port": "5500",
    "Timeout": "00:01:10",
    "RelativePath": "api",
    "ProxyAddress": ""
   }
```

Секция ReindexWorkerSettings описывает параметры внутреннего сервиса переиндексации в компоненте QP8.Search.Admin и содержит следующие параметры:

- Interval – периодичность запуска внутренних задач переиндексации;
- RunTasks – признак необходимости выполнять задачи переиндексации.

Пример корректных настроек:

```
"ReindexWorkerSettings": {
    "Interval": "00:00:10",
    "RunTasks": true
   }
```

Секция ElasticSettings описывает настройки ElasticSearch/OpenSearch и содержит следующие параметры:

- Address – адрес сервера ElasticSearch/OpenSearch;
- ProjectName – короткое название проекта латиницей;
- RequestTimeout – время ожидания ответа от ElasticSearch/OpenSearch.

Внимание: данная секция должна совпадать с одноимённой секцией, используемой при настройке компонентов QP8.Search.Integration и QP8.Search.Api.

Пример корректного заполнения секции:

```
"ElasticSettings":
{
 "Address": "http://127.0.0.1:9200",
 "ProjectName": "demosite",
 "RequestTimeout": "00:10:00.000"
}
```

Не упомянутые выше параметры конфигурации изменять не рекомендуется.

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

Откроем файл конфигурации log-файлов командой:

```
sudo nano /home/quantumart/QP.Search.Admin/NLog.config
```

Найдём строку, содержащую текст logDirectory и в ней у параметра value заменим содержимое на путь к директории log-файлов, которую создали ранее.

Должно получиться так:

```
<variable name="logDirectory" value="/var/log/search-admin"/>
```

Далее в строке

```
<logger name="*" writeTo="console" minlevel="Info" final="true" />
```

следует заменить "console" на "bufferingWrapper_logger".

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

4.6.3. Создание service файла для systemd

Для работы с компонентом как с сервисом с возможностью управления и автоматического запуска создаём файл с именем search-admin.service в директории /usr/lib/systemd/system/ командой:

```
sudo touch /usr/lib/systemd/system/search-admin.service
```

После чего открываем файл на редактирование командой:

```
sudo nano /usr/lib/systemd/system/search-admin.service
```

И заполняем его следующим содержимым:

```
[Unit]
Description=QP Search Administration Console
After=search-integration.service
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=quantumart
WorkingDirectory=/home/quantumart/QP.Search.Admin/
ExecStart=dotnet /home/quantumart/QP.Search.Admin/QA.Search.Admin.dll --urls
http://*:5600
[Install]
WantedBy=multi-user.target
```

В этом файле нас интересует:

- Description – понятное описание сервиса;
- After – указание на то, после запуска какого сервиса нужно запускать этот;
- Restart – политика автоматического перезапуска;
- RestartSec – пауза между перезапуском;
- User – имя пользователя, от которого будет запущен сервис;
- WorkingDirectory – директория, где лежат все файлы приложения;
- ExecStart – команда, выполняемая для запуска сервиса;
    - --urls – указание под каким DNS-именем или IP-адресом разрешено подключаться к компоненту и на каком порту компонент будет ожидать подключения.

Подробно узнать, как настраивать systemd сервисы можно тут.

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

4.6.4. Запуск компонента

Для запуска компонента выполняем команду:

```
sudo systemctl start search-admin.service
```

После чего проверяем что компонент запустился без ошибок командой:

```
sudo systemctl status search-admin.service
```

В случае успешного запуска в строке Active будет указано active (running), дата и время, когда сервис был запущен, а также сколько прошло времени с момента запуска.

Если при запуске компонента что-то пошло не так и у сервиса статус не active, то вероятнее всего при конфигурировании была допущена ошибка. Проверить в чём ошибка можно в log-файлах компонента.

4.6.5. Автоматический запуск компонента при старте ОС

Если сервис успешно запустился, то можно добавить его в автоматический запуск после загрузки ОС.

Для этого выполним команду:

```
sudo systemctl enable search-admin.service
```

4.7. Установка компонента QP8.Search.Api

4.7.1. Создание директории для log-файлов

Создадим директорию куда будут сохраняться log-файлы компонента QP8.Search.Api командой:

```
sudo mkdir /var/log/search-api
```

После чего выдадим права владения этой директорией пользователю quantum art командой:

```
sudo chown quantumart:quantumart /var/log/search-api
```

4.7.2. Настройка компонента QP8.Search.Api

Откроем на редактирование файл конфигурации компонента командой:

```
sudo nano /home/quantumart/QP.Search.Api/appsettings.json
```

Секция Settings описывает основные настройки компонента. Среди всех параметров изменим следующие:

- UsePermissions – указывает использовать ли систему ролевой модели QP8 CMS для ограничения доступа к индексам;
- PermissionsIndexName – название индекса, хранящего списки доступов. Должен совпадать с параметром PermissionIndexName в конфигурации компонента QP8.Search.Integration;
- DefaultReaderRole – название роли доступа к данным без ограничения ролевой модели. Должен совпадать с параметром DefaultRoleAlias в конфигурации компонента QP8.Search.Integration;
- SuggestionsDefaultLength – кол-во результатов выдаваемых в Api подсказок по умолчанию.

Пример корректного заполнения секции:

```
"Settings": {
    "ContextualFields": [ "SearchUrl" ],
    "UserQueryIndex": "userquery",
    "UsePermissions": false,
    "PermissionsIndexName": "",
    "DefaultReaderRole": "Reader",
    "SuggestionsDefaultLength": 10
   }
```

Секция ElasticSettings описывает настройки ElasticSearch/OpenSearch и содержит следующие параметры:

- Address – адрес сервера ElasticSearch/OpenSearch;
- ProjectName – короткое название проекта латиницей;
- RequestTimeout – время ожидания ответа от ElasticSearch/OpenSearch.

Внимание: данная секция должна совпадать с одноимённой секцией, используемой при настройке компонентов QP8.Search.Integration и QP8.Search.Admin.

Пример корректного заполнения секции:

```
"ElasticSettings":
{
 "Address": "http://127.0.0.1:9200",
 "ProjectName": "demosite",
 "RequestTimeout": "00:10:00.000"
}
```

Не упомянутые выше параметры конфигурации изменять не рекомендуется.

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

Откроем файл конфигурации log-файлов командой:

```
sudo nano /home/quantumart/QP.Search.Api/NLog.config
```

Найдём строку, содержащую текст logDirectory и в ней у параметра value заменим содержимое на путь к директории log-файлов, которую создали ранее.

Должно получиться так:

```
<variable name="logDirectory" value="/var/log/search-api"/>
```

Далее в строке

```
<logger name="*" writeTo="console" minlevel="Info" final="true" />
```

следует заменить "console" на "bufferingWrapper_logger".

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

4.7.3. Создание service файла для system

Для работы с компонентом как с сервисом с возможностью управления и автоматического запуска создаём файл с именем search-api.service в директории /usr/lib/systemd/system/ командой:

```
sudo touch /usr/lib/systemd/system/search-api.service
```

После чего открываем файл на редактирование командой:

```
sudo nano /usr/lib/systemd/system/search-api.service
```

И заполняем его следующим содержимым:

```
[Unit]
Description=QP Search Api Service
After=search-integration.service
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=quantumart
WorkingDirectory=/home/quantumart/QP.Search.Api/
ExecStart=dotnet /home/quantumart/QP.Search.Api/QA.Search.Api.dll --urls
http://*:5700
[Install]
WantedBy=multi-user.target
```

В этом файле нас интересует:

- Description – понятное описание сервиса;
- After – указание на то, после запуска какого сервиса нужно запускать этот;
- Restart – политика автоматического перезапуска;
- RestartSec – пауза между перезапуском;
- User – имя пользователя, от которого будет запущен сервис;
- WorkingDirectory – директория, где лежат все файлы приложения;
- ExecStart – команда, выполняемая для запуска сервиса;
    - --urls – указание под каким DNS-именем или IP-адресом разрешено подключаться к компоненту и на каком порту компонент будет ожидать подключения.

Подробно узнать, как настраивать systemd сервисы можно тут.

Сохраним внесённые изменения путём нажатия сочетания клавиш Ctrl+O, затем закроем файл сочетанием клавиш Ctrl+X.

4.7.4. Запуск компонента

Для запуска компонента выполняем команду:

```
sudo systemctl start search-api.service
```

После чего проверяем что компонент запустился без ошибок командой:

```
sudo systemctl status search-api.service
```

В случае успешного запуска в строке Active будет указано active (running), дата и время, когда сервис был запущен, а также сколько прошло времени с момента запуска.

Если при запуске компонента что-то пошло не так и у сервиса статус не active, то вероятнее всего при конфигурировании была допущена ошибка. Проверить в чём ошибка можно в log-файлах компонента.

4.7.5. Автоматический запуск компонента при старте ОС

Если сервис успешно запустился, то можно добавить его в автоматический запуск после загрузки ОС.

Для этого выполним команду:

```
sudo systemctl enable search-api.service
```

5. Установка с использованием Kubernetes

Для установки QP8.Search с использованием Kubernetes требуются:

- Установленный кластер Kubernetes (1.20 и выше);
- Установленный kubectl (1.20 и выше).

5.1. Заполнение манифестов

Вместе с инструкцией поставляется набор из файлов манифестов для развертывания в k8s. Для правильной работоспособности требуется внести корректировки в файл k8s/configmap.yaml:

- ELASTIC_HOST – адрес до кластера ElasticSearch/OpenSearch;
- PROJECT_NAME – имя проекта (в примере используется "demosite" для проекта демонстрационного сайта);
- DB_CONNECTION_STRING – строка подключения к БД, на которой работает сайт и виджетная платформа (QP8.WidgetPlatform);
- SQL_TYPE – тип используемой БД (PostgreSql или SqlServer);
- DEF_SCHEME_NAME – схема, в которой находятся объекты виджетной платформы;
- CONTENT_ACCESS – режим доступа к данным (Live или Stage);
- FORMAT_TABLE_NAME – формат имени таблицы (0 – схема, 1 – имя таблицы, например "{0}.{1}");
- ADMIN_APP_URL – адрес, по которому доступен компонент Admin продукта QP8.Search;
- CONNECTION_ADMIN – строка подключения к первой БД, созданной в пункте 2;
- CONNECTION_CRAWLER – строка подключения ко второй БД, созданной в пункте 2.

В файле k8s/ingress.yaml требуется для каждого приложения в параметре host указать dns-имя по которому приложение должно быть доступно.

5.2. Применение манифестов

После корректировки манифестов согласно вышеописанному пункту следует применить манифесты.

Для этого необходимо открыть командную строку, перейти в директорию с манифестами и выполнить приведённые ниже команды строго в том порядке, в котором они указаны:

```
kubectl apply -f namespace.yaml
kubectl apply -f configmap.yaml
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
kubectl apply -f ingress.yaml
```