Руководство разработчика для модуля Крит-ФМ «RAIDO»

Назначение модуля

Функциональный модуль предназначен для обеспечения возможности интерактивной визуализации, редактирования и анализа семантических графов в веб-браузере для рабочих станций и планшетных компьютеров. Модуль позволяет осуществлять автоматическое размещение узлов графа с помощью силового алгоритма, вычисление спектральных и структурных графовых метрик, кластеризацию и выделение устойчивых сообществ; поиск и слияние отдельных вершин; поиск путей, односвязных и двусвязных окружений, остовных деревьев; фильтрацию на основе семантических типов и меток; визуализировать изменение структуры и метрик графа во времени.

Интерактивная визуализация осуществляется непосредственно на компьютере конечного пользователя. Это обеспечивает отсутствие сетевых задержек между изменением графа и отображением его на устройстве пользователя. При этом производительность размещения узлов графа и визуализации их положения в первую очередь зависит от производительности рабочего места конечного пользователя.

Стандартные рабочие места позволяют комфортно работать с графами размером до нескольких сотен тысяч узлов.

Модуль может использоваться как в составе ИАС "Крит-ФМ", так и как отдельное приложение или как компонент в существующей системе.

Используемый веб-браузер должен поддерживать технологию WebGL, используемое системное окружение должно обеспечить возможность её применения.

Установка и развертывание

Требования к серверу

Для запуска модуля используется современная версия Docker Engine.
Модуль также может быть запущен с использованием иных подобных средств контейнеризации (Podman, Kubernetes).

Установка docker-образа

Дистрибутив поставляется в архиве стандартного формата ZIP. После распаковки в директории `raido-{VERSION}` будут доступны файлы:

- `README.txt` -- краткое описание содержимого архива;
- `docs/raido-user-guide.pdf` -- руководство пользователя;
- `docs/raido-dev-guide.pdf` -- руководство разработчика;
- `raido-{VERSION}.tar` -- образ контейнера в формате docker archive (используется командами `docker image save/docker image load`)

Импорт образа контейнера осуществляется командой

```
docker image load -i raido-{VERSION}.tar
```

После импорта образ контейнера будет доступен в локальном хранилище образов с идентификатором `raido:{VERSION}`.

Запуск модуля осуществляется следующей командой:

```
docker run --name raido -d -p 8080:80 --restart always -m 512m
raido:{VERSION}
```

В этом случае модуль доступен по адресу http://localhost:8080 .

Доступ по сети

Доступ к модулю осуществляется по протоколу HTTP по порту, который указали при запуске контейнера.

Многопоточность

Модуль поддерживает раскладку графов в многопоточном режиме и автоматически переключается в этот режим при возможности его использования.

Для использования многопоточного режима должны выполняться **требования безопасности для использования разделяемой памяти и таймеров высокого разрешения**:

1. Страница модуля должна быть доступна в **безопасном контексте**: через https.
2. Должен быть передан заголовок **Cross-Origin-Opener-Policy: same-origin** .
3. Должен быть передан заголовок **Cross-Origin-Embedder-Policy: require-corp** .

Эти требования должен обеспечивать промежуточный обратный прокси-сервер. В некоторых случаях эти требования могут быть ослаблены. Например, часть браузеров считает **http://localhost** и **http://*.localhost** безопасными контекстами или допускает использование **Cross-Origin-Embedder-Policy: credentialless** вместо **require-corp**.
Для корпоративных браузеров требования могут отличаться, при необходимости уточните конкретные требования у службы поддержки вашего браузера.

HTTP-сервер модуля не требует и не обеспечивает доступ по https, а также не выставляет данные заголовки.

Подготовка файлов для визуализации

Крит-ФМ «RAIDO» предназначен в первую очередь для визуализации графов, созданных внешними системами. Функции редактирования графа не являются основными, создание нового графа напрямую из интерфейса не предусмотрено.

Формат файлов

Входным форматом файлов является собственный формат на основе JSON.
Все ограничения, указанные в руководстве пользователя, являются необходимыми требованиями для входных файлов, а именно:

1. идентификаторы узлов/классов/сетей должны быть уникальными;
2. связи должны соединять узлы соответствующих классов.

На верхнем уровне файл является JSON-объектом типа **Graph**

Поле	Тип	Обязательно	Описание
Graph			
raidoFormatVersion	number	да	Константа версии формата, должна быть равна **-4**
id	string	нет	Название графа
nodes	**Node**[]	да	Массив узлов
nodesets	**Nodeset**[]	да	Массив классов узлов
edges	**Edge**[]	да	Массив связей
networks	**Network**[]	да	Массив сетей
origin	**Origin**	нет	Источник документов из внешней системы
Node			
id	string	да	Идентификатор узла
title	string	нет	Название узла
documents	number[]	нет	Подкрепляющие документы узла
properties	**Property**[]	нет	Свойства узла
Nodeset			
id	string	да	Идентификатор класса узлов
title	string	да	Название класса узлов

type	**NodeType**	нет	Тип класса узлов
start	number	да	Индекс первого узла класса в массиве **nodes**
end	number	да	Индекс последнего узла класса в массиве **nodes** + 1
Edge			
from	number	да	Индекс исходящего узла
to	number	да	Индекс входящего узла
value	number	да	Вес связи
documents	number[]	нет	Подкрепляющие документы связи
properties	**Property**[]	нет	Свойства связи
Network			
id	string	да	Идентификатор сети
title	string	да	Название сети
from	string	да	Идентификатор класса исходящих узлов
to	string	да	Идентификатор класса входящих узлов
start	number	да	Индекс первой связи сети в массиве **edges**.
end	number	да	Индекс последней связи сети в массиве **edges** + 1
Property			
id	string	да	Имя свойства
value	string	да	Значение свойства
* у одного объекта может быть несколько свойств с одинаковым именем, тогда свойство рассматривается как многозначное			
Origin * для стандартного коннектора, см. ниже раздел "Коннекторы для документов"			
id	string	да	Идентификатор источника
title	string	да	Название источника
type	string	да	Тип источника, всегда **documents** для

			стандартного коннектора
documents	**Document**[]	да	Массив документов
properties	**Property**[]	да	Массив свойств источника, всегда должен содержать свойство **uri_prefix**
Document			
id	string	да	Идентификатор документа
date	string	да	Дата документа в формате YYYY-MM-DD
title	string	нет	Название документа
properties	**Property**[]	нет	Свойства документа

Важно: массив **origin.documents** должен быть упорядочен по паре (**date**, **id**) ! Невыполнение требования приведет к ошибке загрузки графа.

Массивы **node.documents** и **edge.documents** должны быть упорядочены.

Тип **NodeType** представляет собой одну из строк из набора: **Agent**, **Organization**, **Knowledge**, **Location**, **Resource**, **Belief**, **Event**, **Task**, **Role**, **Action**. На его основе выбирается внешний вид узла по умолчанию.

Количество узлов в графе не может превышать 2^26 - 1 = 67 108 863, однако это ограничение практически недостижимо исходя из производительности модуля.

Свойство **URI** обрабатывается специальным образом: его значение будет показано в виде гиперссылки.

Связь с нулевым весом и отсутствием свойств считается **тривиальной**. Тривиальная связь может содержать информацию о подкрепляющих документах. В графе не должны присутствовать дублированные тривиальные связи (в одной сети между одними и теми же узлами в одном направлении).

Пример кода на языке Python, который может использоваться для создания файлов в требуемом формате приведен в дополнительных материалах (**bus_demo.py**).

Слияние графов

Объединение графов происходит через интерфейс пользователя или при потоковой загрузке. При объединении применяются следующие правила:

1. для всех объектов: обязательные поля у объектов с одинаковым идентификатором должны совпадать.
2. для узлов:
    1. классы узлов с одинаковым идентификатором должны совпадать;
    2. для узлов с одинаковым идентификатором будет использовано первое присутствующее название узла;
    3. свойства узлов будут объединены;

    4. подкрепляющие документы узлов будут объединены.
3. для связей:
    1. тривиальные связи между одними и теми же узлами в той же сети будут объединены в одну (их подкрепляющие документы будут объединены);
    2. нетривиальные связи никогда не объединяются между собой, в графе могут существовать параллельные связи.
4. для источников:
    1. метаданные источников должны совпадать;
    2. документы с совпадающей датой и идентификатором будут объединены.

Интеграция со внешними системами

Модуль может быть использован как для визуализации графов, загружаемых из файлов на локальной машине, так и для работы с графами, получаемыми напрямую из веб-интерфейсов других систем.

Под **внешними** системами и сервисами ниже понимаются системы и сервисы, не являющиеся частью модуля, доступные в той же сети, что и развернутый модуль.

Прокси

Модуль может быть доступен через общий обратный прокси-сервер совместно с другими внешними системами.

Модуль не использует для своей работы абсолютные пути и может быть размещен за обратным прокси-сервером по любому пути.

Страница **путь_до_модуля/graph/** использует относительный путь **../static/** .

Для корректного отображения прогресса выполнения длительных задач (расчет метрик, поточная загрузка графа) и предотвращения разрывов соединения по таймауту, прокси-сервер не должен препятствовать корректной работе **long polling** запросов, а именно не должен буферизовать ответы от соответствующих компонент.
Для NGINX, например, отключение буферизации реализуется с помощью директивы **proxy_buffering off** .

Необходимо отключить буферизацию для путей **путь_к_модулю/graph/api**/, использующегося для расчета серверных метрик и для внешних ссылок при использовании режима поточной загрузки (см. ниже).

При расчете метрик или использовании внешних конвертеров может быть нужно увеличить максимально допустимый размер тела запроса клиента. Например, для NGINX используется директива **client_max_body_size 512m** .

При возникновении ошибки 413 при расчете метрик необходимо увеличить размер тела запроса для путей **путь_к_модулю/graph/api**/, при использовании внешних конвертеров для путей до них.

Максимальный размер тела запроса для расчета метрик может быть оценен как **количество_связей_в_графе * 12** байт.

Передача графа в соседнее окно браузера

Основным способом интеграции является использование **window.open** и установка специальной переменной, содержащей собранный объект.

Для этого необходимо открыть новое окно браузера через **window.open("путь_к_модулю/graph/#!")** и поместить собранный объект в переменную **passedGraph** дочернего окна.

Загрузка графа по прямой ссылке

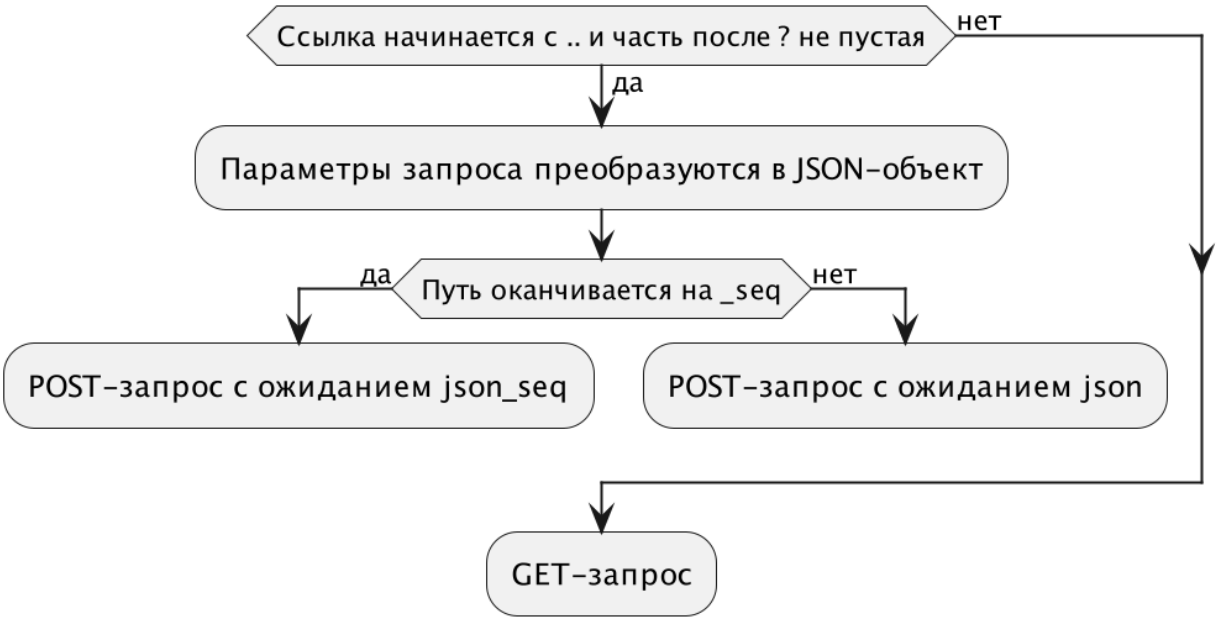
Другим способом интеграции является передача в модуль прямой ссылки на скачивание графа, обращение к которой будет выполнено непосредственно из пользовательского клиента (браузера).

Для этого внешняя система должна осуществить переход по ссылке вида **путь_к_модулю/graph/#?ссылка_на_граф**

Важно: поддерживаются относительные пути. Если ссылка начинается с символов **..** , то модуль использует коннектор к внешней системе, доступной по соседнему пути.

При переходе по ссылке внутри пользовательского клиента (браузера) во фрагменте (части после **#**) обычно можно передать существенно больше данных чем в запросе (части после **?**). Поэтому стандартный коннектор преобразует переданную во фрагменте ссылку в JSON-объект.

Стандартный коннектор работает следующим образом:



1. переданная ссылка разбирается как URL;
2. если «запрос» URL полученного в п.1 (часть после **?**) не пуст, то параметры запроса преобразуются в JSON, на указанный URL выполняется POST-запрос с полученным JSON и модуль ожидает на вход JSON-файл;
3. если используется п.2 и путь оканчивается на **_seq**, то модуль ожидает на вход **application/json-seq,** состоящий из потока объектов специального вида (см. ниже);

4. иначе выполняется GET-запрос и модуль ожидает на вход JSON-файл.

Если коннектор не используется, то всегда выполняется GET-запрос и модуль ожидает на вход JSON-файл.

При необходимости в рамках работ по внедрению модуля могут быть добавлены дополнительные коннекторы, в том числе заменен стандартный и/или изменены правила активации коннекторов.

Потоковая загрузка

Для графов большого размера предусмотрен режим потоковой загрузки. Модуль ожидает поток объектов в формате Record separator-delimited JSON с медиатипом **application/json-seq** (см. RFC 7464: JSON Text Sequences).
Для использования данного режима необходим коннектор к внешней системе.

Допустимые объекты:

1. `{"type": "header", "data": 0}`
2. `{"type": "chunk", "data": ГРАФ}`
3. `{"type": "progress", "data": {"amount": ЧИСЛО, "total": ЧИСЛО}}`
4. `{"type": "error", "data": "ТЕКСТ_ОШИБКИ"}`
5. `{"type": "footer", "data": 0}`

Все графы из объектов **chunk** будут объединены в результирующий граф. На основе объектов **progress** будет показан индикатор выполнения. Объект **error** будет использован для показа ошибок внешней системы.

Поток должен начинаться с объекта **header** и заканчиваться объектом **footer**.

Порядок слияния графов не определен (является деталью реализации), предполагается, что графы полностью корректны и совместимы друг с другом.

Конвертеры для входных данных

В случае если графы загружаются пользователями из файлов на локальной машине, возможно использование внешних сервисов для конвертации графов в требуемый формат.

В таком случае при загрузке файла с локальной машины он будет передан внешнему сервису по протоколу HTTP, и модуль будет ожидать ответ в требуемом формате.

Создание и подключение таких конвертеров может быть выполнено в рамках работ по внедрению модуля.

Коннекторы для документов

Модуль позволяет работать с графами, узлы и связи которых подкрепляются ссылками на внешние документы.

Стандартный коннектор предполагает использование поля **origin** с типом **documents** и будет создавать ссылки на документы вида **<url_prefix>**ids:{<список **id** через пробел>} .

При необходимости в рамках работ по внедрению модуля могут быть добавлены дополнительные коннекторы.

Коннекторы для аутентификации

Модуль не требует аутентификации и авторизации пользователей для своей работы. Предполагается, что передача данных из внешней системы осуществляется только авторизованным пользователем.

При необходимости модуль может проверять статус авторизации пользователя при старте путем запроса на **путь_до_модуля/graph/api/auth** и проверки кода возврата.

При передаче графа по ссылке возможна обработка ошибок 401/403 с показом диалогового окна, либо переходом во внешнюю SSO-систему и повторным выполнением запроса после аутентификации.

Добавление таких коннекторов может быть выполнено в рамках работ по внедрению модуля.