Работа с универсальным средним слоем CommonMiddleware (CMW)

Использование:

Сокращения:
REST (***Re**presentational **S**tate **T**ransfer*)
RPC (***R**emote **P**rocedure **C**all*)
JSON (***J**ava**S**cript **O**bject **N**otation*)
URI (***U**niform **R**esource **I**dentifier*)
URL (***U**niform **R**esource **L**ocator*)
API (***A**pplication **P**rogramming **I**nterface*)

Для проксирования запросов в процессе разработки, потребуется добавить следующую секцию в файл vue.config.js

```
devServer: {
        proxy: {
                '/api': {
                        target: 'http://dskscmw-dev.web.local/dsks-cmw',
                        ws: true,
                        changeOrigin: true
                }
        }
}
```

N.B. target указывает, куда проксировать запрос.

Формирование строки запроса в клиентской части

- "api" – первый сегмент URL (без /) указывающий на API приложения CMW для инфо-обмена с Oracle.
- "/rest", "/rpc" или "/cursor" – сегмент, определяющий ожидаемый тип данных из Oracle. Для rest и rpc – Clob с JSON внутри; для cursor – refcursor.
- "/имя_схемы/имя_пакета/имя_функции" – так же обязательные сегменты **(имя основной схемы, без постфикса _web)**
- "/код" – необязательный сегмент. Нужен для доступа к ресурсу/сущности по коду (идентифицирующему свойству - code). Присутствует только в rest и rpc, в cursor его нет.

Строки запроса итого (в общем виде):
api/rest/schema/catalog/function

api/rest/schema/catalog/function/code
api/cursor/schema/catalog/function

N.B. – никаких параметров для передачи в функции Oracle, через ? и & не предусмотрено. Все параметры должны передаваться в теле POST запроса в JSON формате!

Маршрутизация (routing) в клиентской части приложения

Для обеспечения корректной работы клиентской части приложения, настоятельно рекомундуется использовать режим **hash** роутера Vue.
https://router.vuejs.org/ru/api/#routes

N.B. – для передачи пользователя в Oracle, CMW всегда, даже если не вызывался метод getSession, создает JSON следующего вида

```
{
        data: JSON с данными, если таковые передавались в теле запроса (иначе
- null),
        user: AD логин пользователя, если аутентифицирован (иначе - null)
}
```

T.o. в Oracle, в любом случае, будет передаваться, как минимум один параметр (вышеуказанный JSON). T.e. все методы API в Oracle должны создаваться с учетом этого факта.

Применение API

Разработчику предлагаются на выбор следующие подходы к инфо-обмену между СУБД Oracle и клиентской частью веб-приложения. Отсортированы в порядке приоритетности применения.

1. REST с формированием JSON в Oracle – REST одноуровневый (без вложенных ресурсов в URI) и только с точки зрения взаимодействия клиент-средний слой. Между средним слоем и Oracle – по-прежнему RPC. Сегмент в URL – "/rest".

Важно! ***При формировании JSON объектов в Oracle, именовать свойства объектов, соблюдая camelCaseNotation!***
Рассмотрим подход на примере сущности Unit (сегмент в URI ресурса, в этом случае

будет "/unit" или "/units"). Определим необходимую нам структуру сущности:
unit: {code: …, name: …, score: …, date: …}

- **Получение коллекции.**

В БД потребуется создать функцию с именем **unit**. Вход – varchar или clob с JSON.
Выход – Clob с JSON (коллекция исходных или редуцированных объектов **unit**).
В клиентской части – вызов "**/unit**" тип запроса – **GET**.
api/rest/имя_схемы/имя_пакета/unit - вернет всех юнитов, как правило – это коллекция редуцированных объектов, т.е. с минимальным необходимым набором свойств.
Например: {code: …, name: …}.

- **Получение конкретного ресурса.**

В БД потребуется создать функцию с именем **get_unit**. Вход – код сущности **code** (**number**) **!первый параметр**, Clob с JSON (**varchar2 или clob**) **!второй параметр**.
Выход – Clob с JSON unit.
В клиентской части – вызов "**/unit/{code}**" тип запроса – **GET**.
api/rest/имя_схемы/имя_пакета/unit/100 - вернет юнита с code: 100.

- **Создание ресурса.**

В БД потребуется создать функцию с именем **post_unit**. Вход – Clob с JSON (**varchar2 или clob**). В передаваемом объекте **unit,** свойство code должно быть = null.
Выход – varchar2/clob с JSON-сущностью **unit**. В возвращаемом объекте, свойство code заполнено присвоенным кодом в БД.
В клиентской части – вызов "**/unit**" тип запроса – **POST**.
api/rest/имя_схемы/имя_пакета/unit - вернёт юнита с присвоенным в БД свойством code.

- **Обновление ресурса.**

В БД потребуется создать функцию с именем **put_unit**. Вход – код сущности **code** (**number**) **!первый параметр**, Clob с JSON (**varchar2 или clob**) **!второй параметр**.
Выход – varchar2/clob с JSON-ответом.
В клиентской части – вызов "**/unit/{code}**" тип запроса – **PUT**.
api/rest/имя_схемы/имя_пакета/unit/100 - вернет JSON с ответом, например {resultCode: 0, message: null}.

- **Удаление ресурса.**

В БД потребуется создать функцию с именем **delete_unit**. Вход – код сущности **code** (**number**) **!первый параметр**, Clob с JSON (**varchar2 или clob**) **!второй параметр**.
Выход – varchar2/clob с JSON-ответом.
В клиентской части – вызов "**/unit/{code}**" тип запроса – **DELETE**.

api/rest/имя_схемы/имя_пакета/unit/100 - вернет JSON с ответом, например {resultCode: 1, message: "Ресурс с указанным кодом не найден"}.

2. RPC с формированием JSON в Oracle, но не с позиции ресурсов, а с точки зрения действий. Сегмент в URL – "/rpc".

GET запросы:
api/rpc/schema/catalog/function - В БД потребуется создать функцию с одним входным параметром типа **varchar2 или clob - это Json вида {user: "" , data: null}**. Выход – Clob с JSON.
api/rpc/schema/catalog/function/code - В БД потребуется создать функцию с двумя входными параметрами. **Код сущности первый параметр, Clob с JSON (varchar2 или clob) второй параметр.**. Выход – Clob с JSON.

POST запросы:
api/rpc/schema/catalog/function - В БД потребуется создать функцию с одним входным параметром типа **varchar2 или clob - это Json вида {user: "" , data: тело запроса}**. Выход – Clob с JSON.

Для использования процедуры в Oracle (ничего не возвращает):

api/rpc/proc/schema/catalog/procedure - В БД потребуется создать процедуру с одним входным параметром типа **varchar2 или clob - это Json вида {user: "" , data: тело запроса}**. Выход – без параметров.

3. RPC с возвращаемым Cursor'ом из Oracle. Сегмент в URL – "/cursor".

GET запросы:
api/cursor/schema/catalog/function - В БД потребуется создать функцию с одним входным параметром типа **varchar2 или clob - это Json вида {user: "" , data: null}**. Выход – refcursor.

POST запросы:
api/cursor/schema/catalog/function - В БД потребуется создать функцию с одним входным параметром типа **varchar2 или clob - это Json вида {user: "" , data: тело запроса}**. Выход – refcursor.