Sprint 3. Сложные сущности в REST. Тесты производительности

Спецификация:

К предыдущему заданию добавлены REST-сервисы: Summary, GrouppedView и Report. Summary и GrouppedView имеют только метод GET, Report – GET и POST.

Все GET - методы в качестве параметра получают сущность EventFilter , которая отвечает за фильтрацию и паджинацию данных.

Сервис Summary возвращает сущность Summary, отражающую статистику по событиям, попавшим в выборку. Если заполнено поле GroupBy, то метод должен возвращать ошибку InternalServerError. Метод должен игнорировать поле FileFilter.

В сущности Summary:

- поле Count – означает общее количество событий и интеракций в выборке (обратите внимание, что работает паджинация)
- поле InteractionCount – общее количество интеракций в выборке
- поле EventCount – общее количество событий в выборке
- MostPopularBroker – выдает имя самого встречающегося брокера среди записей в выборке
- AverageDuration – среднее значение поля Duration из записей в выборке.

Сервис GrouppedView группирует события, попавшие в выборку, по полю GroupBy и возвращает их. Поле GroupBy является обязательным. Метод должен игнорировать поля FileFormat, Page и PageSize. Для событий с пустым полем, по которому идет группировка, должна создаваться отдельная группа.

Сервис Report работает с файлами. GET возвращает все события, попавшие в выборку и генерирует файл формата, указанного в FileFormat-параметре. Если заполнено поле GroupBy, то метод должен возвращать ошибку InternalServerError.

Report/POST-метод загружает файл определенного формата (создает новые события для заданного sessionId). Примеры файлов находятся в папке с заданием. Обратите внимание, что параметры EventId, Created и Updated игнорируются, а в CSV-файле их нет (обратите внимание, что для CSV-формата файл полученный через GET не подходит для загрузки через POST). Метод выполняет ту же самую валидацию, что и POST-метод в Event/Interaction сервисах. Дополнительно, CSV-файл обязан иметь все поля.

Все методы REST-сервисов принимают в качестве одного из параметров SessionId, который можно получить с помощью сервиса Accounts. Если SessionId не предоставлен или предоставлен неверный, то должно выбрасываться исключение “Access is denied”.

Все новые и старые методы REST-сервисов должны выполняться не более чем 2 секунды, методы сервиса Reports – 7 секунд.

Описание полей фильтра:

- DateFrom – фильтрация по Date (дата не ранее)
- DateTo – фильтрация по Date (дата не позднее)
- Broker – фильтрация по полю Broker (поиск по подстроке)
- InteractionType – фильтрация по полю InteractionType (полное совпадение)
- MeetingTypes – фильтрация по полю MeetingType (совпадение хотя бы 1 значения в списке фльтра с 1 значением в списке сущности)
- Title – фильтрация по полю Broker (поиск по подстроке)

- Location – фильтрация по полям Location.City и Location.Country (поиск по подстроке), подстрока должна присутствовать хотя бы в 1 из полей.
- Company – фильтрация по полю Company (поиск по подстроке)
- AddressType – фильтрация по полю AddressType (полное совпадение)
- Page – номер страницы (начиная с 0)
- PageSize – количество записей на 1 странице
- IsInteraction – если задано true – присылает только события с Date меньше текущей даты, false – с Date больше текущей даты. Если не задано – возвращает все записи.

Адреса сервисов:

Accounts – `http://soap-att.dio.red/AccountsV3.svc`
Interactions – `http://rest-att.dio.red/v3/Interaction`
Events - `http://rest-att.dio.red/v3/Event`
Summary - `http://rest-att.dio.red/v3/Summary`
Report - `http://rest-att.dio.red/v3/Report`
GrouppedView - `http://rest-att.dio.red/v3/GrouppedView`

Задания:

Написать не менее 150 тестов на новую функциональность (всего должно быть не менее 280 тестов).
Найти не менее 15 багов (вместе со старым функционалом) и написать на них «красные» тесты.

PS не стоит забывать про регрессионное тестирование, потому что возможно что-то сломалось/починилось в новой версии API ☺