Для дальнейшего расширения системы необходимо перейти на WebSocket

Задача состоит в том, чтобы поддерживать постоянный коннект с пользователями для быстрого обновления информации и возможно в реалтайм отслеживать подключенные системы, а также убивать сессии пользователей, заставляя их перелогиниваться

С использованием технологии WebSocket необходимо

1. При подключении пользователя к любому url мы должны запоминать текущее подключение.
   1. Это можно реализовать на момент проверки аутентификации. Даже если у него есть accessToken или RefreshToken, все равно он подтверждается
      1. (проверь насчет accessToken)
2. При наступлении события необходимо передать новые данные всем пользователям, у которых должна измениться информация(открыта соответствующая вкладка)
   1. Если был создан новый документ
   2. Если произошло событие смены статуса документа
   3. Если было создано новое поручение
   4. Если произошло событие смены статуса поручения
3. При устаревании информации необходимо поменять флаг актуальности и обновить информацию у пользователя. Сам флаг должен быть на стороне пользователя(local storage). Необходимо следить за актуальностью информации:
   1. Список документов(тут больше всего статус и этап подписания)
   2. Список поручений(статус, результат)
   3. Нотификации(количество непрочитанных). Отслеживание прочитывания нотификации пока не актуально, подразумевается что один пользователь – один компьютер
4. Должна быть возможность выбивать пользователя из системы путем удаления токенов и насильной перезагрузки страницы
   * 1. Так как сейчас не реализована такая возможность, выкладывать придется вечером, чтобы с утра все зашли с нуля и дали свои подключения
5. Динамически должны обновляться как таблицы документов и поручений, так и текущие открытые документы и поручения
6. Не надо применять эту технологию в архиве
7. Не надо применять эту технологию в админке(пока что)
8. При выходе пользователя из системы любым путем(разлогирование или закрытие вкладки) соответствующее соединение должно быть закрыто

По сути для достижения всего обновления по документам и поручениям нам всегда нужно только вызвать refetch. То есть можно сделать один модуль сервиса WebSocket, который будет принимать для себя идентификатор события(open/close/refetch) и передавать ему функцию для refetch как необязательный параметр. Тогда по идее все должно быть классно. Особенно с учетом что сокеты работают как useState и будут динамически перерисовывать инфу.

Получается, остается только open и close. Open можно навесить на проверку авторизации в хедере(говнокод? Нет, так как вызов login у нас происходит только по истечению accessToken, а до этого надо как то знать о пользователе), close – на разлогин. При закрытии вкладки/браузера close по идее должен автоматически срабатывать

Вообще можно передавать что изменилось и изменять самостоятельно вместо refetch, но это много кода и много отслеживания. Система не настолько высоконагруженная, будут перегружать инфу

Вообще стоит сделать хранение сессий пользователей с помощью вебсокета отдельной вкладкой в админке, чтобы можно было удобно убивать сами сессии, но оставлять заполненные refreshToken

Пусть каждый раз как пользователь переходит на какую то страницу, посылалось сообщение через WebSocket, чтобы отслеживать на какой странице нужно убить кэш

Какие подходы были попробованы и от них отказались:

1. Каждую секунду пулить значения с БД, сравнивать с текущими и применять разницу при их наличии
   1. Слишком много данных по сети, грязный подход
2. Каждую секунду пулить хэш значений из БД и сравнивать с текущим хэшем. При разнице – пулить нормальные данные
   1. Нагрузка на сеть легче, но процесс сбора данных для хеша занимает много времени, сложно поддерживать. Плюс страдает консистентность, при наличии изменений в моей реализации много надо было бы переделывать
3. При получении сигнала по WebSocket запускать процесс перегрузки данных
   1. Потребуется сложная логика, чтобы для каждого пользователя понять нужно ли ему перегружать данные. Получается рабочее решение, но уже через неделю разобраться в нем было очень сложно, даже с учетом комментариев. Плюс из-за циклических зависимостей возникали проблемы и приходилось дублировать код. В общем, сложно, неудобно, но довести до ума недели за 2 можно

Какой подход не успел реализовать:

Идея была в том, чтобы по WebSocket рассылается общее сообщение что договор с таким то ID(допустим) был изменен или создан. Дальше если договор был изменен, то логика на самом клиенте ищет такой договор у себя в кэше и, если находит, то перегружает данные. Если же договор был создан, то необходимо определить, относится ли договор к этому пользователю. Думал сделать еще один endpoint для выгрузки только id договоров для этого пользователя и поиска нового договора в этом. Или просто посылать на сервер запрос по типу «Я пользователь с id=1, получил сообщение что договор 100 был обновлен/добавлен. Мне надо его выгружать?» и ответ Да/Нет, по получению которого происходит соответствующее действие