Ты из API берешь данные и заносишь их в базу (1).

Далее Ты по тикеру достаёшь последние 50 записей со статусом PROCESSED или NEW (2)

Далее Ты скармливаешь их воркерам (3)

Если воркер обработал OK, то он пишет обратно в базу(4)

Если плохо - просто пропускает (5)

Если один из воркеров ловит ошибку за исключением 429 ТО ? (6)

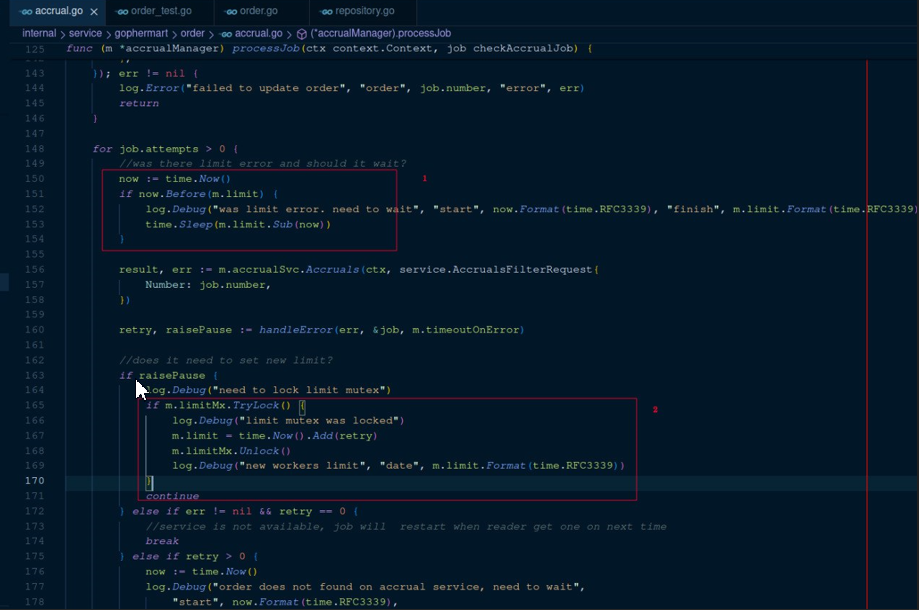
Если ловит 429, то засыпает и все засыпают воокеры (7) - вот тут поподробнее что происходит и где ?

Если происходит завершение приложения, то канал doneCh ловит это и завершаются все воруеры , после этого только происходит отключение приложухи

по идее обратаывается 2 типа ошибок. LimitError и ErrEntityDoesNotExists, остальные случаи и считают, что сервис не доступен и это п.5

5 - заказ пропускается и через время опять появится в п.2

6 - обработка ErrEntityDoesNotExists. handleError уменьшает у job количество возможных попыток, засыпает и потом опять пытается получить данные с accrual. по сути это закрывает кейс, когда нам загрузили липовый заказ, а интернет-магазин и следовательно accrual сервис не знает о нем



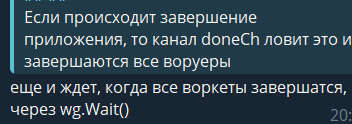
7.

на этапе 1, проверяем, можно ли уже оправлять запросы, если дата лимит меньше чем текущая дата, то продолжаем нормально работу.

иначе засыпаем и ждем на limitDate - nowDate

2. Если поймали ошибку лимита, нам возврается time.Duration, то есть через сколько можно продолжить работу. Мы пробуем заблокировать мютекс через TryLock, нам вернет false если его уже кто-то заблокировал, значит другой воркер уже поймал такую же ошибку и уже устанавливает новый лимит.

Если заблокировали, устанавливаем новый лимит и уходим на следующую итерацию и попадает в п.1 и значит засыпаем



accrual возвращает ошибку 429, там есть заголовок Retry-After: {qty seс when you can send request again}

эта количество секунд прибавляется к limit и всё

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

На rules service прилетает новое правило. Через менеджер уведомяются подписчики. Тоже самое с новым заказом.

Рассчитывать может ведь другой сервис

Гофермарт тут вообще не причём

income это признак приход/расход,

sum сколько денег начислено/списано.

В income в принципе смысла особо нету сейчас т.к. sum может быть отрицательной и положительной и это является признаком прихода/расхода

manager нужен для регистрации определенных функций, чтобы их выполнять параллельно верно ?

Чтобы получать уведомления о событиях

Распространение событий.

В accrual происходит два события, NewRule, NewOrder.

В данный момент единственный подписчик на события это calculation service.

В целом подписчик это просто функция определённой сигнатуры.

Ты уже работал с подобным. Это хэндлеры у веб сервиса. Ты подписываешь на определённый pattern uri и когда приходит запрос сервер передаёт твоей функции этот запрос. Логика такая же происходит событие и менеджер распространяет его по подписчикам

Не. Это не workedPool. Это просто распространён событий. RegisterHandler подписываешь на событие. RaiseEvent добавляет событие в обработку