**Примеры сценариев для описания:**

**1. Процесс «От поставщика до склада»**

* *Как новый товар появляется в системе?*
* Вы создаёте Product и ProductUnit одновременно?
* Как он попадает в определённый Spine?
* Какие статусы (CLEAR, CANDIDATE, SPROUTED) проставляются и когда? Кто и почему меняет quantityInCandidate?

**2. Процесс «Создание и жизнь заявки»**

* *Что такое «заявка» в вашей системе?*
* Как пользователь формирует заявку? Он просто фильтрует товары по Spine и меняет им статус на IN\_REQUEST?
* Как связаны quantityInRequest, createdAtRequest и requestPricePerUnit? Кто и когда их заполняет?
* Что происходит с заявкой дальше? Она превращается в продажу?

**3. Процесс «Продажа товара»**

* *Как происходит продажа?*
* Пользователь работает с конкретным ProductUnit или сразу с несколькими?
* Как создаётся CashEvent типа SALE? Кто заполняет salePrice, soldAt?
* Как закрывается CashDay? Вручную или автоматически?

**4. Процесс «Возврат товара»**

* *Как оформляется возврат?*
* Как меняются статусы? Как это отражается в CashEvent и CashDay?

**Почему это важно?**

Эти сценарии покажут **реальную нагрузку** на вашу схему данных и помогут спроектировать **сервисный слой** в NestJS правильно. Например:

* Если статусы меняются сложно — нужна отдельная StatusService.
* Если продажа involves несколько сущностей — нужна TransactionService с транзакцией.
* Если нужно агрегировать данные (например, остатки по Spine) — нужен ReportService.

**Опишите 1-2 самых главных процесса, с которых начнёте разработку.** Этого будет достаточно для глубокого анализа и конкретных рекомендаций по архитектуре NestJS модулей, сервисов и DTO.

хорошо я буду тебе отвечать . а ты сейчас переключить в режим слушателя. ты принимаешь обьяснения. а не пишешь мне в ответ много текста. ты будешь задавать четкие краткие вопросы а я отвечать. а когда все будет готово - ты переключишься обратно в режим когда ты сможешь полностью коментировать . ключевая фраза для этого -"Я ДАЛ ОТВЕТЫ НА ВСЁ, СЛУШАЮ РЕКОМЕНДАЦИИ" от меня. сценарии: 1) создание Product. Для этого надо создать категории, бренд, spine (название как у товара). далее я создаю Product. (привязка к Spine) он является шаблоном. и не играет дальше этого шага роли в нашем проекте. 2) на странице отображения Produts в виде карточек есть кнопка - создать карточку товара. так создаётся Product-unit . переносятся данные от Product в unit. 3) на странице Spines у нас обобщаются все units привязанные к Spine. на одной карточке разграниченные по Brands . визуально на карточке я могу переключать и видеть статусы , количество. еще какую то информацию. 4) Формирование заявки. а) На странице Spines я нахожу карточку , выбираю бренд. б) у нее внутри отображен Unit со статусом Clear , перевожу в Candidate. (она становится кандидатом на заявку). в) после я указываю количество , цену за штуку, нажимаю кнопку В ЗАЯВКУ. Меняется статус на IN REQUEST. Возникает потребность создание новой Unit от того же Product как вышеупомянутая, статус у нее будет CLEAR. 5) так как на этом Spine карточке мы видим состояние всех unit то статистика в цифрах должна отображать количество units clear | units candidate | units in REQUEST | DELIVERY | IN STORE | SALE (за эту дату). 6) логирование и отчет состояний , мелким шрифтом где то я должен прочитать отчет что происходило. и открывающийся ЛОГ по unit должен иметь запись о том что произошло с этой unit. (это два разных кейса). 6.5) логика создания заявки когда у нас unit уникален. имеет 2 варианта исполнения а) единичный. ( 1 ед. заказана - 1 получена). б) множественный. ( несколько единиц заказано). логика: - фиксируем сколько единиц заказано.quantityInRequest и цена за единицу. - статус unit становится SPROUTED (связан) - столько сколько quantityInRequest указано - создаются units , (это дети от SPROUTED ) - parentProductUnitId у них, а у родителя массив childProductUnits - у детей становится статус IN\_REQUEST, цена за единицу копируется от родителя, - прочие данные которые должны скопироваться тоже копируются от родителя. (например поставщик, покупатель). 7) Должна быть страница Получения товара. на ней spine должны отображаться unit длинными малыми карточками, группированными по Spine и по дате заявки. отображают дату заявки, артикул unit , имя, цену за единицу , и в правой части - часть для поставки - дата поставки , (сегодня), количество пришедшего, сумма. примечание. подтвердить (кнопкой). при реализации поставки статус меняется на IN\_DELIVERY. (добавляется дата в unit и время). после этого статус меняется на IN\_STORE (Добавляется дата и время). 8) при продаже товара статус меняется на SOLD, цена продажи сохраняется в поле Unit ( если продан в кредит, то статус меняется на CREDIT, цена продажная 0, привязка к покупателю). может я где то упустил . но все движения с изменением unit - записываются в логах. чтобы жизненный цикл можно было прочитать.

Отлично, я слушаю. Задаю уточняющие вопросы по вашим сценариям.

**Вопрос 1:** Когда вы создаете Product-unit (п.2), вы сразу указываете поставщика (supplierId) и прочие данные? Или это происходит позже, на этапе заявки?

**Вопрос 2:** В п.4в, после перевода единицы в заявку, вы говорите о создании *новой* Unit со статусом CLEAR. Зачем? Это резервирование под будущие поставки?

**Вопрос 3:** В п.6.5 (множественная заявка), кто является "родителем" - оригинальная Unit со статусом CLEAR, или специально созданная для этого Unit со статусом SPROUTED?

**Вопрос 4:** В п.7, при подтверждении поставки, статус меняется на IN\_DELIVERY, а затем на IN\_STORE. Это два отдельных действия пользователя или один шаг (подтверждение поставки автоматически перемещает в IN\_STORE)?

**Вопрос 5:** В п.8, при продаже в кредит, цена продажи 0. Это окончательно? Не будет ли позже отдельного этапа оплаты этого кредита, который зафиксирует цену и создаст CashEvent?

К Вопрос 1: supplier должен формировать до создания product и передаваться в unit к Вопрос 2: Spine отображают только units, они с products не взаимодействуют, и CLEAR это начальное состояние units, от него я могу перейти в кандидаты и в заявку, а без CLEAR не могу. К Вопрос 3: при множественной заявке - CLEAR трансформируется в SPROUTED - и она является родителем, - после формирования множественной заявки (как операция), создается unit со статусом Clear, за место истраченной unit, которая стала SPROUTED. ( чтоб мы могли видеть в spine карточку с начальной стадией цикла). понятно ли это? К Вопрос 4: пользователь подтверждает поставку единицы товара кнопкой. (товар поставлен), - IN\_REQUEST переходит в - IN\_DELIVERY добавляется дата и время поставки для дальнейших отчетов. и далее само переходит в IN\_STORE и добавляется дата и время, (для отчетов), к Вопрос 5: я недоконца тебе написал бизнес логику, но суть в том что человеку дается товар, без оплаты (поэтому цена продажи 0)( за день эта продажа не суммируется в сумме продаж), однако записывается координаты человека, а потом когда он приходит и дает деньги, поднимается событие с оплатой в кредит, (Unit еще не был в статусе SOLD), и мы получаем деньги, переводим в статус продано, и можешь посчитать сумму продаж..

Понял. Задаю последние уточняющие вопросы.

**Вопрос 6:** В п.6.5 (множественная заявка), когда SPROUTED-родитель создаёт IN\_REQUEST-детей, должны ли эти дети сразу привязываться к конкретному поставщику и заказчику (копируя данные от родителя)?

**Вопрос 7:** В п.8 (продажа в кредит), когда товар отдают без оплаты, какой статус у Unit в этот момент? CREDIT? И меняется ли он на SOLD только после финальной оплаты?

**Вопрос 8:** В п.7, при подтверждении поставки, статус меняется с IN\_REQUEST на IN\_DELIVERY, а затем автоматически на IN\_STORE. Нужно ли логировать оба этих перехода (IN\_REQUEST->IN\_DELIVERY и IN\_DELIVERY->IN\_STORE) в ProductUnitLog?

К Вопрос 6: да должны. все данные которые были у родители для работы должны копироваться. К Вопрос 7: - при продаже в кредит, у товара становится статус CREDIT - меняется на SOLD после получения денег. К Вопрос 8 : да. все переходы статусов нужно логировать. абсолютно все.

Отлично. Картина полностью ясна.