1. *Подумайте, какие сервисы системы имеет смысл перевести на событийную модель работы. Всю систему или только отдельные сервисы? Приведите бизнес-обоснование для сервисов, которые будут переведены на событийную модель. Каковы польза и риски такого решения?*

Считаю, что чем проще – тем лучше. Меньше логики – меньше точек отказа и потенциальных проблем. События – дополнительная точка отказа (в виде обработчика событий, будь то очереди или что иное). Потому перевод – по минимуму.

Во-первых, логирование как таковое (оно же сбор метрик и статистик). Отдельно для маркетинговых показателей (Пользователь зарегистрировался, перейдя с Яндекса), отдельно для технических (Пользователь зарегистрирован в БД Пользователей за эн секунд). На схеме – красные взаимодействия.

Во-вторых, работу с заказом ПОСЛЕ его оформления. Тоесть тот этап, когда сиюминутный ответ не нужен, можно спокойно разбираться и подождать. Мало ли, понабежит зерг-раш и застопит очередь сверх-нагрузкой. Показаны синими стрелочками на схеме.

Польза – логи не требуют сиюминутной реакции, в то же время уменьшается нагрузка на сервисы (они не думают куда, не выбирают, просто кричат в очередь «я сделал»). Всегда будет удобная информация для менеджеров (маркетинг и бизнесс-аналитик) и технических товарищей (разграничение доступа никто не отменял. Требования к базам и способам взаимодействия с сервисами логирования так же можно настроить индивидуально.). Повышение скорости реагирования на отказы (отсутствие событий – тоже событие).

Обработка заказа после оплаты (на этапе склада и доставки) – весьма длительный процесс, зависящий от человека. События позволят одновременно и не ждать ответа, и не потерять его (склад может собрать заказ за неделю, а может и за 5 минут. Логистика может месяц не иметь подходящих машин. НО! Менеджер будет видеть статус каждого заказа на каждом этапе и периодически будет прозванивать задерживающую службу с фразой «ну что там».) Повышенный контроль при пониженных затратах на синхронизацию взаимодействий. Так же повышается безопасность пользователя: даже если упадет любой из сервисов насмерть с потерей БД – из логов можно восстановить все заказы и не потерять их.

Сайты, в основном, завязаны на прямых запросах к иным сервисам (Сайту неважно знать все Заказы, только те, что принадлежат конкретному Пользователю в данный момент.) Формировать команду «Пользователь зашел на страницу личного кабинета, дай мне данные для отображения» и ждать реакции – а вдруг очередь зависнет? Или именно в этот момент иной сервис отчаянно отбивается от спам-атаки и очередь забита иными сообщениями? В таких случаях прямой запрос – надежнее и быстрее (без посредников). А скорость в работе сайта – критически важна. Аналогично любые взаимодействия с Клиентом – любые изменения должны получаться мгновенно, а не когда-нибудь, когда придет очередь. Изменение телефона должно обрабатываться максимально быстро, чтобы менеджер всегда имел максимально актуальную информацию.

Логистика и склады так же, в основном, работает в рамках самих себя и генерировать события должны только в ответ на пришедшие команды и события извне.

1. *Для выбранных в первом вопросе сервисов опишите основные агрегаты, их назначение и события/команды, которые они будут обрабатывать/генерировать. Результат представьте в виде перечисления и описания агрегатов. Составьте диаграмму событий/команд, покажите их связь, то есть на какие события или команды реагируют агрегаты.*

Логи: реагируют на события иных разделов. Агрегатами являются Логи, в общем случае служебные и маркетинговые. Более точное деление зависит от собираемых метрик и целей. Например, «Лог Заказа».

Заказ: показывает статус и движение заказа клиента в системе. События: «Заказ сформирован Клиентом», «Заказ отправлен на оплату», «В Заказ добавлен Продукт», «Из Заказа удален Продукт», «Заказ оплачен». Команды: «Собери Заказ», «Доставь заказ». После получения «Заказ доставлен» - «Заказ успешно выполнен».

Склад: реагирует на Заказы, отвечает на них. Получает «Заказ отравлен на оплату» и «бронирует» Продукты на время оплаты. Если не получает «Заказ оплачен» - снимает бронь. Получает «Собери заказ» и, после фактической сборки, генерирует «Заказ собирается», «Заказ собран».

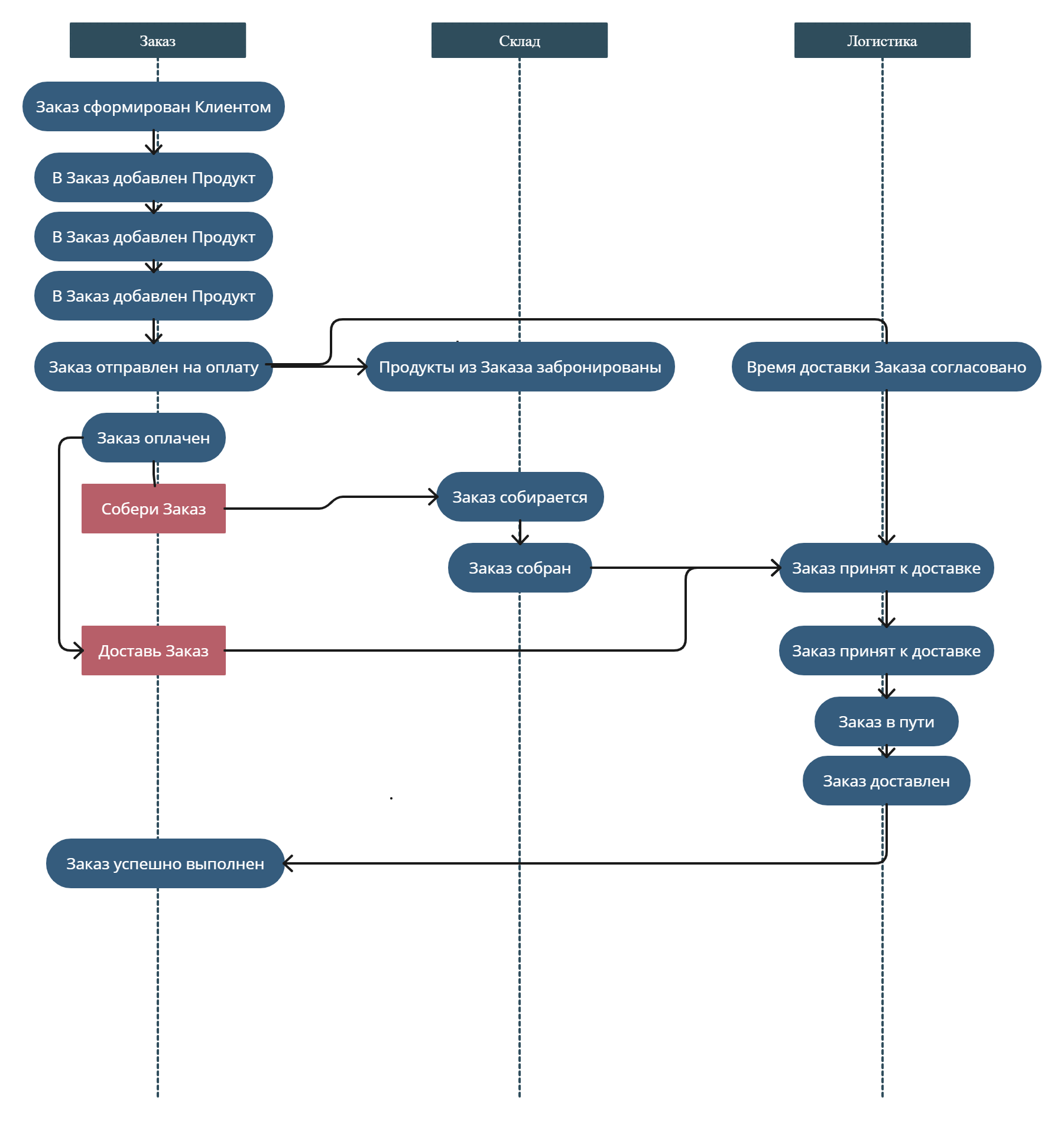
Логистика: реагирует на Заказы, отвечает на них. Получает «Доставь Заказ», ждет «Заказ собран» и, после доставки, генерирует «Заказ принят к доставке», «Заказ в пути», «Заказ доставлен».

Это идеальное движение. Теперь корректировки:

«Заказ отменен Клиентом» - все реагируют: Склад – разбирает, Логистика – не привозит, ставит у себя ссоответствующие статусы.

«В оплаченном Заказе изменилась комплектация» - как про добавление Продукта – только в уже оплаченные Заказы в процессе сбора и доставки. Все реагируют, пересобирают, передоставляют.

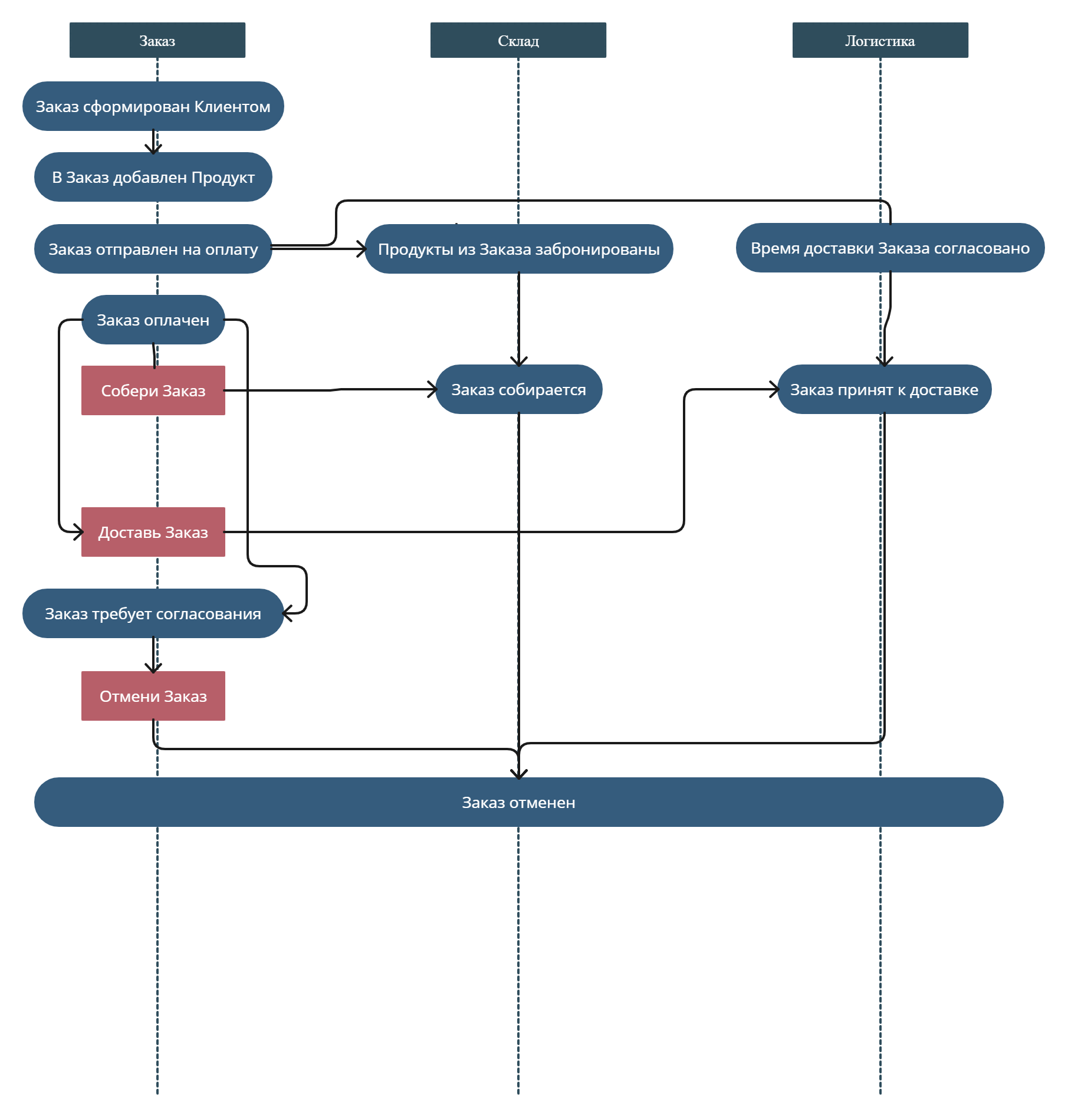
«Заказ требует согласования» и «Заказ согласован» - при особых случаях, когда возникают какие-либо триггеры (большой заказ, дорогой заказ, «важный» Клиент), все обработки приостанавливаются до момента согласования. Или в иных случаях по решению бизнеса.



Здесь сознательно упрощено взаимодействие с логистикой: изменить время и дату, изменить тип доставки на самовывоз – много однообразного текста.

1. *Выберите одну из цепочек событий (не менее 4-5 шагов) и распишите содержание команд/событий для этой цепочки. Обоснуйте выбор полей для описанных команд/событий.*

Пусть ситуация не так радужна и Заказ сорвался. Последовательность такова:



Начало стандартное:

* Заказ сформирован клиентом – по этому событию идут записи в Логи, ~~менеджеры берут низкий старт~~. Статус Заказа стандартный «На оформлении».
* В Заказ добавлен Продукт – маркетологам важно знать какие товары «конкурентны», обычно люди накидывают в корзину несколько вариантов и выбирают лучший. К заказу прикрепляются Id Продукта и их колличество.
* Заказ отправлен на оплату – с этого момент Менеджерам нужно быть на низком старте. Раз клиент сформировал счет для оплаты – его можно додавить, связаться, скинуть скидочку. Склад, получив данный сигнал бронирует нужные Продукты, чтобы следующим Клиентам не демонстрировать потенциально купленный товар и не сесть в лужу, продав десяток экземпляров эксклюзивной мебели, существующей в единственном экземпляре. Отдельного события не генерирует, пишет у себя в базе, сам же и снимает бронь через некоторое время, в этот период отдает Сайту актуальные данные о наличии Продуктов. В логистике ставят бронь на время доставки, начинают думать, какой именно машиной её осуществить. Аналогично, снимают без внешних условий через некоторое время. В Заказе статус – «в оплате». Сам скидывается обратно через некоторое время.
* Заказ оплачен – менеджеры вздыхают спокойнее, Клиент не сорвался. По этому сигналу генерируются две команды складу и логистике соответственно на сбор и доставку товара. Статус заказ «Оплачен», склад и логистика работают по внутреннему распорядку.
* Но случилась беда. Было куплено 5 экземпляров Продукта в ином Заказе, Клиенту придется ждать (сработал триггер в базе на перебор по проданным Продуктам) – генерируется событие «Заказ требует согласования». Получив его, Склад и логистика доводят текущий этап до конца (склад собирает, но Логистике не отдает), а дальше ждут отмашки. Менеджер вручную ртом должен звонить Клиенту или иным образом решать проблему до момента согласования…
* Менеджер позвонил, пообщался, расстроился и нажал кнопку «Клиент ушел». Сгенерирована команда «Отменить Заказ». По ней Заказ ставит статус «Отмен», Склад возвращает Продукты на полки, снимает брони, Логистика – освобождает ячейку в календаре на доставку. Менеджер получает втык.

На каждом этапе каждое событие пишется в логи, причем одно событие содержит всю инфу, а пишется разная:

* Техническая, связанная со временем получения и обработки событий, скоросью обработки и т.д. забирается Логом Техническим, а маркетинговые фишки (товар добавлен-удален, заказ сформирован 10 раз с 2-3 товарами и т.д.) – Маркетинговым.

1. *На архитектурной диаграмме (общий план архитектуры с прошлых домашних заданий) отобразите необходимые технические компоненты для работы с командами и событиями.*



События взаимодействуют по схеме. Логи читаются внутренним интерфейсом. Сайт напрямую запрашивает данные у сервисов (стрелками не указано во избежание награмождления). Системы доставки и логистики тоже сложные, но в рамках решаемой задачи – несущественно.