1. **Проанализируйте схему и описание системы.** Идентифицируйте существующие и потенциальные проблемные места. Напишите их список.

**Существующие проблемы:**

* + 1. Клиенты жалуются менеджерам по продажам, что они не получили заказ и над их изделием работают уже несколько месяцев, хотя обещали закончить за три недели.
    2. После того как компания открыла свой API и стала принимать заказы от других продавцов жалоб от клиентов на просроченные заказы стало ещё больше
    3. Добавились жалобы со стороны пользователей API (это другие продавцы ювелирных изделий). Представители компаний ежедневно сообщают, что не получили свои заказы.
    4. Когда оператор заходит на первую страницу MES, система долго прогружается. (На первой странице отображается список заказов в работе по статусам — это дашборд с фильтром. Раньше страница показывала все заказы, но это тормозило загрузку).
    5. QA-инженер прогоняет E2E-сценарии вручную в тестовом окружении. Если обнаруживает баги уровня high или highest, то это часто приводит к значительным задержкам релиза.
    6. Отсутствие сбора бизнесовых и инфраструктурных метрик.
    7. Отсутствие сбора логов
    8. Отсутствие сбора трейсов внутренних вызовов

**Потенциальные проблемные места:**

* + 1. Просроченные и неполученные заказы. Раз проблемы не на стороне производства, то возможны такие сценарии:

производство не получило информацию о новом заказе и не делала его, но статус заказа у клиента стоит как «оператор взял заказ в работу»

производство выполнило заказ отправила его, но статус заказа у клиента НЕ отображается как «заказ завершен»

транспортная компания задерживает доставку заказов

Тут скорее всего проблема в сервере MES, он не выдерживает высокую нагрузку, связанную с повышением количества заказов в следствии чего происходит падение системы. Либо очередь не справляется с нагрузкой.

* + 1. Слишком долгое исполнение запроса, т.е. высокая латентность, которая отрицательно влияет на отображение списка заказов для оператора. Тут проблема в том, что нет кеширования.
    2. Нехватка компетенций у QA инженера.
    3. Могут возникнуть проблемы при изменении контракта открытого АПИ. То есть если API уже используется другими приложениями (наших клиентов), изменения могут привести к несовместимости, что потребует от разработчиков обновления их кода.
    4. Для микросервисов CRM и Shop реализована общая БД, т.е. не соблюдается паттерн Database per Service. При каком подходе масштабировать микросервисы независимо друг от друга не получится, возможна конкуренция микросервисов за ресурсы в перспективе. Общая БД будет также препятствием для автономной работы команды разработки над конкретным сервисом, т.е. не соблюдение паттерна Service per Team.

1. **Разработайте инициативы, которые необходимы для устранения нежелательных ситуаций.**
   * 1. Устранить проблему просроченных заказов. Тут необходимо применить один и паттернов снижения нагрузки (Backpressure). Тогда получится избежать падения системы и ограничить поток данных, возможно также предотвратить обслуживание пользователей во время пиковых нагрузок.
     2. Для быстрого отображение списка заказов для оператора и клиентов, потребуется реализация кеширования.
     3. Нанять в штат опытного QA инженера, для реализации автоматического тестирования системы.
     4. Надо понять, где чаще всего возникают баги уровня high или highest , это влияет на выпуск релиза. Возможно, понадобится повысить квалификацию разработчиков или дополнительно нанять.
     5. Добавить мониторинг. Это позволит собрать бизнесовые и инфраструктурные метрики
     6. Добавить логирование. Позволит фиксировать изменение состояния системы с течением времени.
     7. Добавить трейсинг, для отслеживания запросов в системе и анализа логов.
     8. Нанять еще одного DevOps инженера , который будет отвечать за мониторинг, логирование и трейсинг + настроит автоматический деплой приложения в продакшн и release-окружение
     9. Можно попробовать просчитать уровень нагрузки микросервиса: определить основные задачи микросервиса и ожидаемый объем трафика (примерное количество клиентов, частота заказов). На основе исторических данных попробовать смоделировать нагрузку. Потом провести нагрузочные тесты с использованием сторонних инструментов. Это нам поможет определить, как микросервис реагирует на различные уровни нагрузки. Полученная картина поможет в выборе архитектурных решений.
     10. Посмотреть существующий код, это позволит найти узкие места и потенциальные проблемы производительности. Возможно аккуратно отрефакторить его.

1. **Расставьте инициативы в порядке приоритета.** Опишите ход своих рассуждений и ответьте на вопросы:
   * Какой вы видите целевую архитектуру через полгода?
     1. Настроить мониторинг. Метрики помогут оценить, насколько будут эффективны предпринятые нами шаги.
     2. Самое важное это устранить проблему просроченных заказов. Т.е. реализовать паттерн Backpressure. Он обеспечит обратную связь для клиентов и представителей компаний, с помощью которой регулируется скорость формирования новых заказов в соответствии со скоростью их обработки на производстве.
     3. Реализовать серверное кеширование c использованием инструмента Redis для сервисов MES/Shop/CRM.
     4. Настроено логирование/трейсинг.
   * Если бы у вас была возможность выполнить только три пункта из списка инициатив в ближайшие полгода, что бы вы выбрали и почему? Не обязательно добавлять в список только эпики. Вы можете включить в план как крупные изменения, так и локальные задачи.
     1. Поставить задачу DevOps инженеру настроить мониторинг, логирование и трейсинг + автоматический деплой. Нам необходимо знать проблемные места в системе и оценить эффективность наших дальнейших действий.
     2. Устранить проблему просроченных заказов – т.к критично, иначе потеряем всех клиентов (реализовать паттерн Backpressure).
     3. Реализовать кеширование в MES для быстрого отображения актуальных данных о заказах оператору. Это важно , оператор должен оперативно получать информацию о заказах в системе.