# **Задание 3. Трейсинг**

## **Анализ системы в контексте планирования трейсинга**

### **Потенциальные места, где заказ может «сломаться» или зависнуть**

1. **Создание заказа через интернет-магазин (Shop API)**:

* Ошибка при создании записи в базе данных (Shop DB).
* Ошибка при загрузке 3D-модели в S3.
* Заказ не отправляется в очередь (RabbitMQ) для расчёта стоимости.

1. **Обработка заказа в CRM**:

* Ошибка при чтении заказов с рассчитанной стоимостью из очереди RabbitMQ.
* Ошибка при обновлении статуса заказа в базе данных (Shop DB) после расчёта стоимости.
* Заказ не отправляется в очередь (RabbitMQ) после подтверждения.

1. **Обработка заказа в MES**:

* Ошибка при чтении сообщения из очереди (RabbitMQ) с информацией о новом заказе для расчёта стоимости.
* Ошибка при чтении сообщения из очереди (RabbitMQ) с информацией о подтверждённом заказе для производства.
* Ошибка при загрузке 3D-модели из S3 для расчёта стоимости.
* Ошибка при отправке результата расчёта в RabbitMQ.
* Ошибка при чтении заказов для обработки в производственной системе (MES DB).
* Ошибка при обновлении статуса заказа в MES DB во время работы оператора.

1. **Интеграции с S3**:

* Ошибка при загрузке 3D-модели в S3 (на этапе создания заказа).
* Ошибка при скачивании 3D-модели из S3 (на этапе расчёта стоимости).

1. **Очереди RabbitMQ**:

* Заказ может зависнуть, если сообщение не было прочитано из очереди (задержка в обработке).
* Сообщение попадает в dead-letter exchange, если его не удалось обработать корректно.

### **Элементы системы, которые стоит покрыть трейсингом**

1. **Shop API** — трейсинг всех операций по созданию заказа, загрузке модели в S3 и отправке заказа в RabbitMQ.
2. **CRM API** — трейсинг обработки сообщений из RabbitMQ, обновления статуса заказа в базе данных и отправки на производство.
3. **MES API** — трейсинг создания заказа B2B-клиентом, получения заказа из RabbitMQ, расчета стоимости, обновления статуса в MES DB и взаимодействия с S3.
4. **RabbitMQ** (опционально) — трейсинг всех сообщений, которые передаются между сервисами, с указанием статуса доставки и состояния очередей.

Элементы, которые стоит покрыть трейсингом, выделены на схеме 3.1

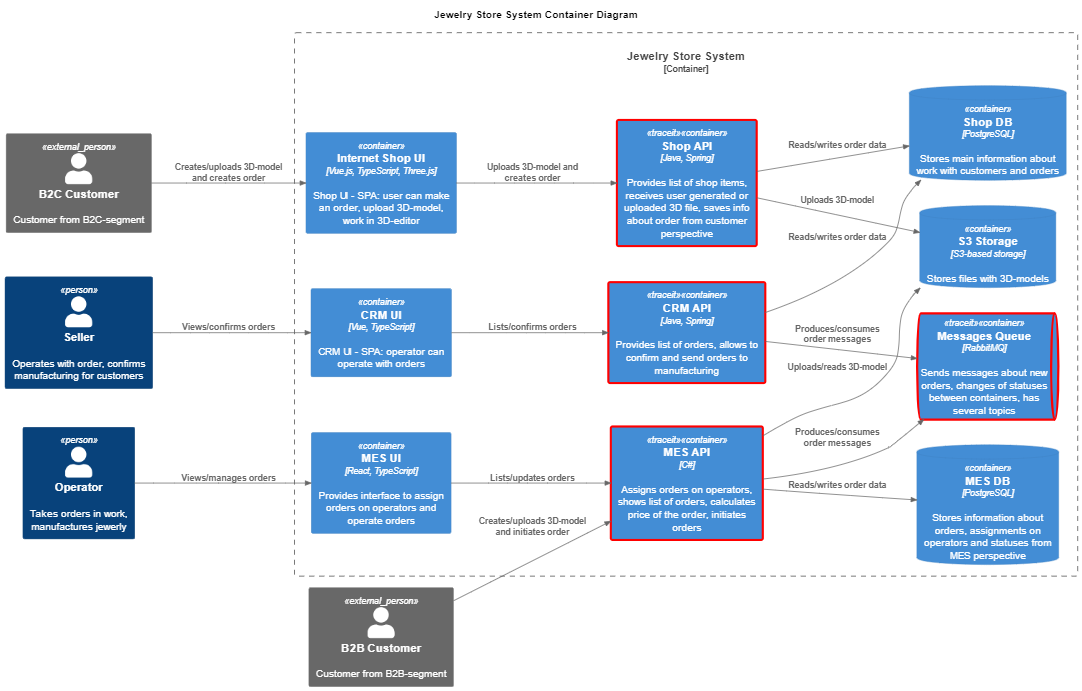


Схема 3.1 Контейнерная диаграмма ASIS для системы Jewelry Store System с выделением элементов для покрытия трейсингом

### **Список данных, которые должны попадать в трейсинг**

1. **ID заказа** — уникальный идентификатор, который должен передаваться на всех этапах, чтобы можно было проследить весь жизненный цикл заказа.
2. **Метаданные об операции**:

* Время начала и завершения операции (timestamps).
* Длительность выполнения операции.
* Имя метода или операции, которая выполняется.
* Имя сервиса, который выполняет операцию.

1. **Статус операции**:

* Успешность или неуспешность выполнения (например, success, failure).
* Тип ошибки, если операция завершилась сбоем (например, ошибка подключения к базе данных, ошибка при работе с S3).
* Код ответа, если операция взаимодействует с внешним API (например, HTTP-статусы).

1. **Сообщения RabbitMQ**:

* Время отправки и получения сообщения.
* Размер очереди в момент отправки и получения.
* Тип сообщения (например, SUBMITTED, PRICE\_CALCULATED).
* Статус обработки сообщения (успешно, в dead-letter, повторная попытка).

1. **S3-интеграция**:

* Путь к файлу в S3.
* Размер файла.
* Время загрузки или скачивания файла.
* Статус операции (успех, ошибка).

1. **База данных (Shop DB, MES DB)**:

* Время выполнения операций записи и чтения.
* Количество строк, которые были изменены или прочитаны.
* Статус операции (успех, ошибка).
* Код ошибки, если операция завершилась сбоем (например, ошибка таймаута, блокировка).

1. **Метрики производительности**:

* CPU и RAM в момент выполнения операции (в случае подозрений на перегрузку).
* Время ожидания ответа от очередей или внешних сервисов (например, S3, RabbitMQ).

## **Мотивация**

Трейсинг обеспечивает полный обзор того, как заказ движется через все этапы и взаимодействия. Он помогает отслеживать путь каждого заказа и выявлять узкие места или проблемы, такие как задержки, сбои в передаче сообщений или некорректная обработка данных.

Внедрение трейсинга решает несколько ключевых задач:

1. **Диагностика и устранение проблем**: Трейсинг позволяет видеть, на каком этапе заказ зависает или «ломается», что значительно ускоряет решение инцидентов.
2. **Проактивное обнаружение узких мест**: Трейсинг выявляет медленные или перегруженные части системы, позволяя принимать меры до того, как они начнут критически влиять на производительность и бизнес-процессы.
3. **Оптимизация производительности**: Зная, где система работает медленно или неэффективно, можно оптимизировать эти участки, что ведет к снижению задержек в обработке заказов.
4. **Улучшение SLI и пользовательского опыта**: Быстрая диагностика проблем с заказами напрямую влияет на качество обслуживания клиентов, сокращая время простоя и недовольство клиентов.

### **Технические и бизнес-метрики, на которые повлияет внедрение трейсинга**

**Технические метрики:**

1. **Время ожидания в очередях RabbitMQ**

* **Как влияет трейсинг**: Трейсинг позволит отслеживать, сколько времени сообщения ожидают обработки в очереди RabbitMQ. Это позволит выявить перегруженные очереди или задержки, связанные с медленной обработкой сервисов (например, MES или CRM). В случае выявления проблем можно распределить нагрузку или добавить инстансы.
* **Результат для компании**: Снижение задержек в очередях ускоряет обработку заказов и предотвращает их зависание, что приводит к повышению производительности всей системы и уменьшению количества недовольных клиентов.

1. **Нагрузка на ресурсы и их использование**

* **Как влияет трейсинг**: Трейсинг помогает выявить моменты пиковых нагрузок на различные сервисы и ресурсы (например, Shop API, CRM API, MES API), что позволяет предотвратить перегрузки и падение производительности, оптимизируя использование ресурсов. Например, можно адаптировать использование вычислительных мощностей и улучшить планирование масштабирования.
* **Результат для компании**: Оптимизация использования ресурсов снижает затраты на инфраструктуру и предотвращает сбои, вызванные недостатком ресурсов, особенно в пиковые периоды заказов.

1. **Уровень ошибок и количество инцидентов**

* **Как влияет трейсинг**: Отслеживание ошибок в процессе обработки заказов (например, ошибки взаимодействия с RabbitMQ, S3, базами данных) позволяет идентифицировать и устранять проблемные места до того, как они начнут сильно влиять на систему. Трейсинг поможет быстро найти место, где произошёл сбой, и устранить его.
* **Результат для компании**: Снижение количества ошибок в системе уменьшает количество обращений в службу поддержки, что снижает операционные расходы компании и улучшает клиентский опыт.

**Бизнес-метрики:**

1. **Среднее время обработки заказа**

* **Как влияет трейсинг**: Трейсинг поможет увидеть, сколько времени занимает каждый этап обработки заказа, начиная от создания заказа в интернет-магазине до его доставки. Сбор данных о продолжительности выполнения операций позволит оптимизировать процессы и сократить общее время выполнения заказа, что приведет к уменьшению времени выполнения заказа (Time-to-Delivery).
* **Результат для компании**: Сокращение времени обработки заказа улучшает удовлетворенность клиентов, что может напрямую влиять на повторные заказы и лояльность к компании.

1. **Процент завершённых заказов без ошибок**

* **Как влияет трейсинг**: Система трейсинга позволяет точно отслеживать процент заказов, которые доходят до статуса «SHIPPED» или «CLOSED» без проблем на разных этапах обработки. Если трейсинг фиксирует проблемы (например, недоступность базы данных или сбои при взаимодействии с S3), можно оперативно устранить их и предотвратить дальнейшие сбои.
* **Результат для компании**: Повышение процента успешных заказов снижает количество возвратов или отмен и увеличивает общую прибыльность бизнеса, поскольку клиенты получают свои заказы вовремя.

## **Предлагаемое решение**

1. **Выбор системы трейсинга**Основные технологии для распределённого трейсинга:
   * **OpenTelemetry** — платформа наблюдения с открытым исходным кодом, которая предоставляет стандартизированный способ сбора, обработки и экспорта телеметрических данных-сигналов (метрик, трассировок и журналов) из распределённых систем.
   * **Jaeger** — инструмент для агрегации и визуализации трейсинговых данных с поддержкой OpenTelemetry.
2. **Необходимые доработки**Для интеграции OpenTelemetry с сервисами необходимо добавить/настроить:
   * OpenTelemetry SDK для Java в Shop API.
   * OpenTelemetry SDK для Java в CRM API.
   * OpenTelemetry SDK для .NET в MES API.
   * Telemetry Exporter для RabbitMQ.
3. **Алертинг на основе трейсинга**Для эффективного использования трейсинга необходимо реализовать автоматический мониторинг процесса прохождения заказа и алертинг.  
   Для этого потребуется:
   * Установить Prometheus monitoring mixin в Jaeger.
   * Настроить отправку трейсов/метрик из Jaeger в Prometheus.
   * Настроить соответствующие панели и дашборды в Grafana.
   * Настроить Grafana Alerting для алертинга по следующим событиям:
     + Задержка в обработке заказа: если время выполнения определённых операций превышает заданные пороговые значения (например, расчёт стоимости в MES API занимает более 30 минут).
     + Проблема в цепочке обработки: если заказ не движется между сервисами в течение заданного времени (например, задержка между CRM и MES).

Новые компоненты и связи отражены на схеме 3.2

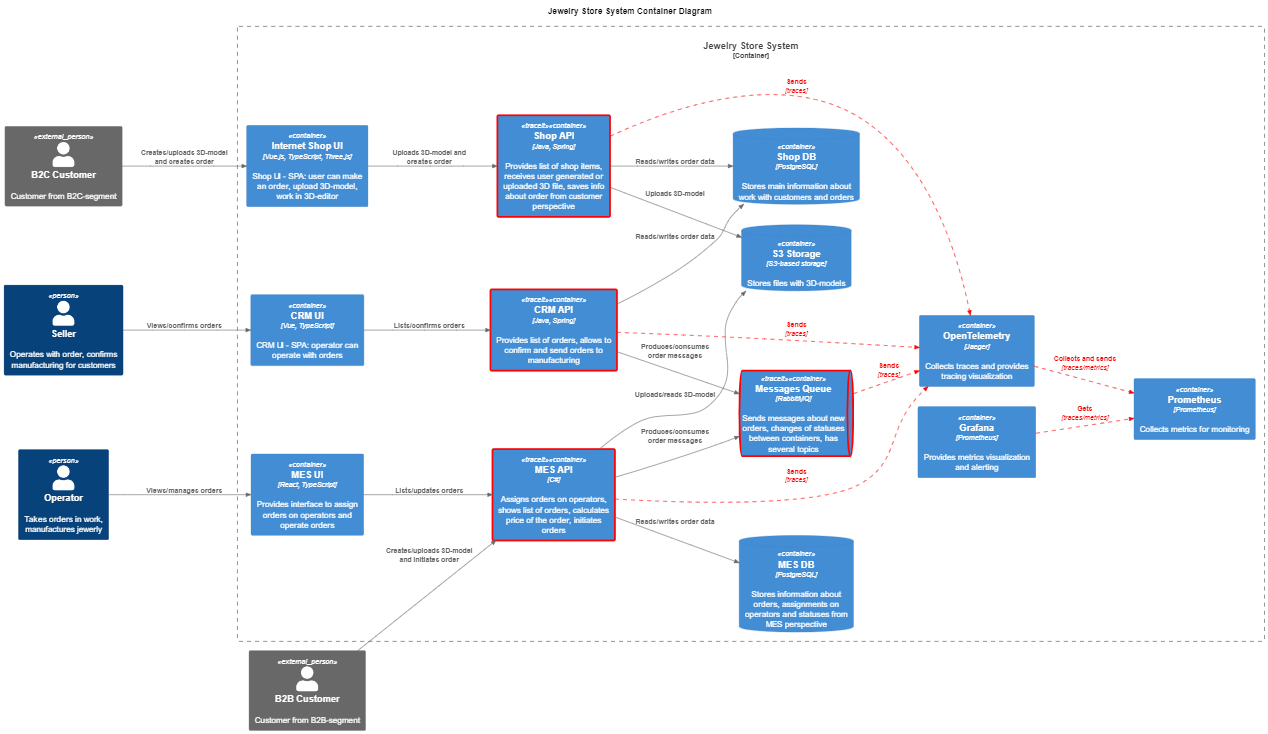


Схема 3.2 Контейнерная диаграмма для системы Jewelry Store System с новыми компонентами для трейсинга

## **Компромиссы**

Для экономии ресурсов можно отказаться от покрытия трейсингом RabbitMQ, т.к. действительно важная информация об отправке заказа в очередь, чтении из очереди и сбоях при этих операциях может быть получена из трассировочных сообщений сервисов Shop API, CRM API и MES API.

## **Проработка аспектов безопасности**

Для предотвращения несанкционированного доступа к системе трейсинга как внутри компании, так и снаружи, необходимо принять следующие меры (по уровням взаимодействия):

1. **Client to Collector** (передача трейсов от клиента напрямую к коллекторам Jaeger): Т.к. бэкенд Jaeger не поддерживает TLS для HTTP, необходимо развернуть реверс-прокси перед коллекторами (например, с использованием Nginx), настроенный для использования TLS и аутентификации.
2. **Browser to UI** (доступ к интерфейсу Jaeger UI через браузер): Т.к. бэкенд Jaeger не поддерживает TLS для HTTP, необходимо развернуть Jaeger UI за реверс-прокси, который будет обеспечивать HTTPS доступ и шифрование передаваемых данных.
3. **Client to Agent** (внутреннее взаимодействие, передача данных о трейсах от клиентов к агенту Jaeger): Исключить такой вариант взаимодействия, обеспечив интеграцию OpenTelemetry с сервисами через OpenTelemetry SDK и экспортеры данных.

Для прочих уровней взаимодействия (Agent to Collector, Collector/Query to Storage, Consumers to Query Service) предусмотрено использование TLS.

Также можно предусмотреть дополнительные меры по обеспечению безопасности:

1. **Внедрение единой системы аутентификации (SSO)**. Для доступа к трейсинговой системе только сотрудники компании смогут использовать свои корпоративные учетные записи. Это позволит централизовать управление доступом и сократить риски, связанные с потерянными или украденными учетными данными. Также можно предусмотреть двухфакторную аутентификацию.
2. **Использование системы ролей**. Доступ к трейсинговым данным будет предоставляться только сотрудникам с соответствующими ролями, например:
   * **Поддержка** (Support): может видеть основные трейсинговые данные для диагностики ошибок.
   * **Разработчики**: могут получать доступ к более детализированным данным для анализа производительности и поиска багов.
   * **Администраторы**: имеют полный доступ к настройкам системы трейсинга и его конфигурации.
3. **Сегментация доступа по зонам безопасности**. Доступ к системе трейсинга будет ограничен только внутренними сетями компании, с минимизацией количества внешних точек входа. Сотрудники, работающие удаленно, смогут получить доступ только через VPN с шифрованием.
4. **Ограничение по IP-адресам**. Настройка белых списков IP-адресов для доступа к системе трейсинга. Это означает, что доступ к системе возможен только из сетей, находящихся под управлением компании.
5. **Журналирование действий**. Вся активность в системе трейсинга будет журналироваться, включая входы в систему, доступ к трейсинговым данным, изменения конфигурации и настройки системы. Это позволит отслеживать любые подозрительные действия и оперативно реагировать на потенциальные угрозы.
6. **Мониторинг и защита от аномалий**. Система должна иметь возможность обнаруживать аномальные паттерны запросов (например, подозрительно большое количество запросов на чтение данных), что может быть признаком попытки взлома или DDOS-атаки.
7. **Обновления безопасности**. Все компоненты системы трейсинга (Jaeger) должны регулярно обновляться для устранения уязвимостей и улучшения их безопасности.