Собирать логи требуется из всех систем (кроме Managed Databases).

В логах любого уровня по умолчанию логируем следующие поля:

* Время события — точное время записи лога (в формате UTC).
* Идентификатор системы — название микросервиса или компонента, из которого был записан лог.
* Уникальный идентификатор запроса (Trace ID) — для связывания событий в рамках одного запроса.
* Уникальный идентификатор операции (Span ID) — для детального анализа действий внутри запроса.
* Идентификатор пользователя — если событие связано с конкретным пользователем (например, ID клиента или сотрудника).
* Версия сервиса, имя docker-образа
* Окружение (dev, prod)

Необходимые логи с уровнем INFO:

* Бизнес-события
  + Создание нового заказа: список товаров, сумма заказа, источник (например, мобильное приложение или веб-сайт).
  + Изменение статуса заказа:номер заказа, новый статус.
  + Поступление оплаты
  + Регистрация пользователя
  + Изменение профиля пользователя: измененные поля
* Технические события
  + Успешная/неуспешная авторизация
  + Взаимодействие с внешними сервисами: время вызова, название сервиса, результат (успешно/ошибка).
  + Запуск или завершение приложения/сервиса
  + Запуск или завершение задачи на обработку 3D-модели
  + Запуск или завершение бизнес-процесса (саги)

Дополнительные уровни логирования:

* DEBUG: для диагностики проблем в процессе разработки или при сложных инцидентах в продакшене (например, детализированные запросы к БД).
* ERROR: фиксируются только ошибки, влияющие на работоспособность системы, например, сбои внешних интеграций или фатальные исключения.
* WARN: логирование потенциальных проблем, не влияющих на текущую работу, например, превышение времени ответа на запрос.

# Мотивация

Логирование в системе позволяет фиксировать ключевые события, что способствует:

* Быстрой диагностике ошибок. При возникновении сбоев или аномалий логирование помогает определить источник проблемы, сократить время на её устранение и минимизировать последствия для бизнеса.
* Мониторингу состояния системы. Логи дают возможность отслеживать производительность, нагрузку и точки отказа системы.
* Анализу пользовательского поведения. С помощью логов можно понять, как клиенты взаимодействуют с системой, выявить узкие места или определить потенциальные точки улучшения.

Метрики, на которые влияет логирование

* **Время реакции на инциденты.** Логирование позволяет быстрее выявлять и устранять причины проблем, что снижает простой системы.
* **Объём ошибок в производственной среде.** Логи позволяют вовремя обнаруживать и устранять ошибки до того, как они затронут пользователей.
* **Удовлетворённость клиентов.** Прозрачный анализ событий повышает качество сервиса и снижает недовольство клиентов.

### **Приоритеты в настройке логирования и трейсинга**

Если ресурсы разработчиков ограничены, то предлагаю настроить только логирование с использование corellationId - идентификатора, который будет “путешествовать” между системами в заголовках запросов. Таким образом, можно будет по этому идентификатору отфильтровать все логи по интересующему запросу в Kibana, что в общем-то является отдаленным аналогом трейсинга.

* MES: Настроить логирование изменения статусов заказов, взаимодействие с базой данных, обработку API-запросов.
* Интернет-магазин: логирование входящих запросов, время обработки заказов, изменения статуса заказов.
* CRM: настроить логирование операций с клиентами, входящих и исходящих запросов, операции с заказами

# Предлагаемое решение

### **Реализация логирования и необходимые технологии**

**Технологии для логирования:**

* **Logstash/Fluent Bit/Fluentd** для сбора логов с разных систем
* **Elasticsearch/OpenSearch** для хранения и анализа логов.
* **Kibana/OpenSearch Dashboards** для визуализации и работы с логами.
* Встроенные библиотеки логирования в приложениях (например, Serilog для .NET).

**Компоненты и доработки:**

* Настройка агентов Fluent Bit/Fluentd на всех серверах и контейнерах.
* Интеграция логеров в приложения для отправки логов в Fluent Bit.
* Настройка индексации и ротации логов в Elasticsearch/OpenSearch.

**Схема (**рисовать не буду, просто словами поясню**):**

* Приложения (интернет-магазин, CRM, MES) → Fluent Bit → Elasticsearch/OpenSearch → Kibana/OpenSearch Dashboards.
* RabbitMQ и базы данных → Fluent Bit → Elasticsearch/OpenSearch.

### **Политика безопасности**

**Работа с чувствительными данными:**

* Логи, содержащие персональные данные или конфиденциальную информацию (например, идентификаторы клиентов, платежные данные), будут обезличены или замаскированы перед отправкой в хранилище.
* Логи передаются и хранятся в зашифрованном виде (TLS для передачи, шифрование данных в Elasticsearch).

### **Политика хранения**

**Индексы:**

* Разделение логов по системам: отдельный индекс для логов интернет-магазина, CRM, MES, RabbitMQ.
* Дополнительный индекс для логов ошибок (ERROR).

**Срок хранения:**

* **Информационные логи (INFO):** хранятся 30-90 дней.
* **Ошибки (ERROR):** хранятся 6-12 месяцев для анализа инцидентов.
* **Отладочные логи (DEBUG):** хранятся 7-14 дней.

**Размер логов:**

* Рассчитать на основании ожидаемого объёма логов. Например, для интернет-магазина при средней нагрузке можно ожидать около 1–5 ГБ логов в сутки.

**Ротация:**

* Настроить автоматическую ротацию на уровне Elasticsearch/OpenSearch.
* Архивирование устаревших логов (gzip и перенос на дешёвое хранилище, например, S3).

### **Мероприятия для превращения системы сбора логов в систему анализа логов**

#### **Настройка алертинга**

* **Метрики производительности и доступности:**
  + Увеличение количества HTTP 5xx (ошибки серверов) выше установленного порога.
  + Увеличение времени ответа API (латентности) выше 95-го перцентиля.
* **События безопасности:**
  + Попытки несанкционированного доступа (многочисленные неудачные попытки аутентификации).
  + Аномальный рост числа запросов с одного IP-адреса (потенциальный DDoS).
* **События приложения:**
  + Резкое увеличение количества записей об ошибках (ERROR).
  + Снижение числа заказов в CRM или завершённых операций в MES.
* **Инфраструктура:**
* Увеличение загрузки CPU/памяти выше 80% на инстансах приложений или баз данных.
* Переполнение дискового пространства на серверах логов.

#### **Поиск аномалий**

* **Автоматический поиск аномалий:**
  + Использование таких инструментов, как **Elasticsearch Machine Learning** или **Prometheus Alertmanager**, для анализа трендов в логах и поиска отклонений.
* **Примеры сценариев аномалий:**
  + Резкий рост записей о создании заказов (например, до 10 000 в секунду вместо обычных 5–10).
  + Увеличение частоты ошибок авторизации в системе (возможные атаки на пароли).
  + Аномально большое число соединений к базе данных.