# **Задание 2. Мониторинг**

## **Мотивация**

Добавление мониторинга в систему обеспечит компании улучшенную видимость системы, контроль и управление производственными процессами, что приведёт к значительным бизнес-выгодам.

* Мониторинг позволит выявить узкие места системы для точечного применения масштабирования.
* Мониторинг позволит наиболее оперативно выявлять проблемы в системе, чтоб устранить их до того, как появятся новые жалобы от клиентов. Это особенно важно в виду планируемых архитектурных изменений.
* Систематическое выявление и устранение проблем на ранних этапах поможет сократить расходы на техническую поддержку, внеплановые ремонтные работы и снизить риски потерь от простоев.

## **Выбор подхода к мониторингу**

Для мониторинга различных сервисов системы стоит применить различные подходы:

* **Shop DB** и **MES DB**: применяем **USE**, т.к. для баз данных важен мониторинг инфраструктурных аспектов, которые **USE** как раз и отражает в наибольшей степени.
* **Shop API**, **CRM API**, **MES API** (API операторов и B2B-клиентов): применяем **RED**, т.к. он лучше подходит для сервисов, управляемых запросами, но отслеживать инфраструктурные метрики (CPU, память) для этих сервисов также необходимо.
* **MQ**: применяем **4 золотых сигнала** и инфраструктурные метрики (CPU, память). Поскольку всё взаимодействие между компонентами завязано на MQ, контроль трафика, лага и ошибок в очередях будет в значительной степени отражать состояние системы в целом.

### Список метрик для отслеживания

1. **Онлайн-магазин (Shop API и Shop DB)**

* **Number of requests (RPS) for internet shop API**  
  Зачем нужна: Важно отслеживать RPS, чтобы выявлять пиковые нагрузки и обеспечивать стабильную работу системы.  
  Ярлыки:
  + url: для анализа нагрузки по конкретным эндпоинтам.
  + method: типы запросов.
* **Response time (latency) for shop API**  
  Зачем нужна: Задержки в ответах могут негативно сказаться на пользовательском опыте. Важно отслеживать эту метрику, чтобы понимать, когда возникают проблемы с производительностью.  
  Ярлыки:
  + url: для отслеживания задержек по конкретным эндпоинтам.
  + method: типы запросов.
  + region: регион пользователей, что поможет идентифицировать проблемы с сетью или географической доступностью.
* **Number of HTTP 200/500 for shop API**  
  Зачем нужна: Метрика успешных (200) и ошибочных (500) запросов помогает быстро идентифицировать проблемы в системе. Всплески ошибок укажут на сбои.  
  Ярлыки:
  + url: для анализа успешности и сбоев для конкретных эндпоинтов.
  + method: типы запросов.
* **CPU % for shop API / Memory Utilisation for shop API**  
  Зачем нужна: Эти метрики показывают уровень использования ресурсов серверов. Важно отслеживать их для предотвращения перегрузок и оптимизации ресурсов.  
  Ярлыки:
  + instance: для мониторинга нагрузки на каждом сервере.
* **Number of connections for shop db instance**  
  Зачем нужна: Количество соединений с базой данных позволяет оценить, насколько интенсивно используются её ресурсы, и вовремя выявлять проблемы, связанные с перегрузкой соединений.  
  Ярлыки:
  + status: активные или простаивающие соединения.
* **Size of shop db instance**  
  Зачем нужна: Контроль за объемом базы данных позволяет планировать масштабирование и предотвратить переполнение дисков.  
  Ярлыки: не требуются.
* **Number of active sessions**

Зачем нужна: Для предотвращения злоупотреблений учётной записью с передачей её другим пользователям.  
Ярлыки:

* + ip: ip-адрес, с которого осуществлён вход..

1. **CRM (CRM API и Shop DB)**

* **Number of requests (RPS) for CRM API**  
  Зачем нужна: Эта метрика поможет отслеживать количество запросов по управлению заказами для сравнительного анализа.  
  Ярлыки:
  + url: для анализа использования конкретных эндпоинтов.
  + method: типы запросов.
* **Response time (latency) for CRM API**  
  Зачем нужна: Время отклика CRM влияет на эффективность работы сотрудников. Задержки могут негативно сказываться на оперативности работы.  
  Ярлыки:
  + url: для отслеживания задержек по конкретным эндпоинтам.
  + method: типы запросов.
* **Number of HTTP 200/500 for CRM API**  
  Зачем нужна: Отслеживание успешных и ошибочных запросов помогает в выявлении проблем в работе CRM и своевременном их исправлении.  
  Ярлыки:
  + url: для анализа сбоев и успешных запросов по конкретным эндпоинтам.
  + method: типы запросов.
* **CPU % for CRM API / Memory Utilization for CRM API**  
  Зачем нужна: Как и в случае с Shop API, отслеживание использования ресурсов помогает оценивать производительность системы и избежать перегрузок.  
  Ярлыки:
  + instance: для мониторинга нагрузки на каждом сервере.
* **Number of connections for shop db instance**  
  Зачем нужна: Количество соединений с базой данных позволяет оценить, насколько интенсивно используются её ресурсы, и вовремя выявлять проблемы, связанные с перегрузкой соединений.  
  Ярлыки:
  + status: активные или простаивающие соединения.

1. **MES (MES API и MES DB)**

* **Number of requests (RPS) for MES API**Зачем нужна: Количество запросов к MES API от операторов и B2B-клиентов позволяет отслеживать нагрузку на систему.  
  Ярлыки:
  + url: для анализа нагрузки по конкретным эндпоинтам, чтоб понимать источник нагрузки.
  + method: типы запросов.
* **Response time (latency) for MES API**  
  Зачем нужна: Высокая задержка в MES может затормозить работу операторов и привести к задержкам в производстве. Отслеживание этой метрики позволит вовремя выявлять проблемы с производительностью.  
  Ярлыки:
  + url: для отслеживания задержек по конкретным эндпоинтам.
  + method: типы запросов.
* **Number of HTTP 200/500 for MES API**

Зачем нужна: Мониторинг успешных и неудачных запросов показывает, как система справляется с задачами и выявляет сбои.  
Ярлыки:

* + url: для анализа сбоев и успешных запросов по конкретным эндпоинтам.
  + method: типы запросов.
* **CPU % for MES API / Memory Utilization for MES API**  
  Зачем нужна: MES выполняет сложные расчеты, и важно следить за тем, как эффективно используются ресурсы серверов.  
  Ярлыки:
  + instance: для анализа загрузки конкретных серверов.
* **Size of MES db instance**  
  Зачем нужна: Увеличение размера базы данных MES напрямую связано с объемом обрабатываемых заказов и хранением результатов расчета. Мониторинг объема базы данных поможет планировать масштабирование.  
  Ярлыки: не требуются.

1. **RabbitMQ**

* **queue\_totals.messages**Зачем нужна: Рост общего количества сообщений в очереди может указывать на то, что потребитель не справляется с нагрузкой.  
  Ярлыки:
  + queue\_name: для определения конкретного потребителя, не справляющегося с нагрузкой.
* **queue\_totals.messages\_unacknowledged**  
  Зачем нужна: Рост количества неподтверждённых сообщений в очереди может указывать некорректную работу потребителя.  
  Ярлыки:
  + queue\_name: для определения конкретного проблемного потребителя.
* **message\_stats.publish\_details.rate**  
  Зачем нужна: Для своевременного выявления роста нагрузки на очередь и потребителей.  
  Ярлыки:
  + queue\_name: для определения конкретных потребителей, которым предстоит справляться с растущей нагрузкой.
* **message\_stats.deliver\_get\_details.rate**  
  Зачем нужна: Для определения способности потребителей справляться с растущей нагрузкой.  
  Ярлыки:
  + queue\_name: для определения конкретных потребителей, которым предстоит справляться с растущей нагрузкой.
* **Number of dead-letter-exchange letters in RabbitMQ**  
  Зачем нужна: Эта метрика показывает, сколько сообщений не смогли быть обработаны и попали в dead-letter queue. Это важный индикатор проблем в системе обмена сообщениями между сервисами.  
  Ярлыки:
  + queue\_name: для отслеживания конкретных очередей, которые вызывают проблемы.

1. **Прочие метрики**

* **Size of S3 storage**  
  Зачем нужна: Хранение файлов 3D-моделей в S3 может занимать значительные ресурсы. Контроль за объемом хранилища позволит вовремя расширять или оптимизировать его.  
  Ярлыки: не требуются.

## **План действий**

1. **Создание инфраструктуры для сбора и хранения метрик**

* Создать инстанс time-series базы данных с использованием Prometheus или альтернативного решения для сбора метрик.
* Настроить хранилище метрик на основе выбранной time-series базы (Prometheus, InfluxDB) для долговременного хранения данных.
* Настроить сбор логов и метрик для пересылки метрик из RabbitMQ, API и баз данных в time-series хранилище.

1. **Интеграция с сервисами и сбор метрик**

* Добавить экспорт метрик в Shop API, CRM API и MES API.
* Настроить экспорт метрик из RabbitMQ.
* Настроить экспорт метрик из баз данных (Shop DB, MES DB).

1. **Настройка мониторинга инфраструктуры**

* Интегрировать мониторинг ресурсов Yandex Cloud для мониторинга состояния EC2 инстансов (CPU, память, дисковое пространство) и S3-хранилища.
* Подключить метрики с балансировщиков нагрузки и сетевой инфраструктуры, если она используется, для мониторинга сетевых запросов, времени ответа и загрузки каналов.

1. **Визуализация данных**

* Развернуть систему визуализации метрик с использованием Grafana или аналогов. Настроить дашборды для отображения ключевых метрик для каждой части системы (Shop, CRM, MES, RabbitMQ).
* Создать дашборды по API для мониторинга RPS, ошибок, задержек, использования ресурсов и других важных показателей.
* Создать дашборды по RabbitMQ для мониторинга состояния очередей.
* Настроить дашборды по базам данных для анализа использования ресурсов и соединений.

1. **Настройка алертов**

* Настроить правила оповещения (алерты) в системе мониторинга для ключевых метрик: ошибки 500, высокие задержки, превышение пределов использования CPU, памяти, переполнение очередей в RabbitMQ.
* Добавить алерты для RabbitMQ на количество dead-letter сообщений и долгую обработку сообщений в очереди.
* Настроить алерты по состоянию баз данных: превышение лимитов соединений, медленные запросы, увеличение объема данных.
* Настроить алерты для API на основе задержек ответов и ошибок.

1. **Оптимизация системы на основе метрик**

* Анализировать собранные данные для выявления узких мест (например, медленные запросы в API, перегруженные очереди RabbitMQ).
* Разработать план по масштабированию компонентов системы на основе анализа данных (увеличение количества инстансов API, базы данных или настроек очередей RabbitMQ).
* Провести оптимизацию баз данных: анализ и оптимизация медленных запросов, масштабирование хранилищ.

1. **Документация и обучение команды**

* Создать документацию по метрикам и алертам: описать, какие метрики собираются, что они означают и какие действия нужно предпринимать при срабатывании алертов.
* Провести обучение команды по работе с мониторингом: как использовать дашборды, анализировать данные и реагировать на алерты.

1. **Тестирование системы мониторинга**

* Провести нагрузочное тестирование системы для проверки работоспособности мониторинга под реальной или увеличенной нагрузкой.
* Тестировать все алерты: убедиться, что оповещения корректно срабатывают при достижении пороговых значений.

## **Работа с показателями насыщенности**

1. **Использование CPU для Shop API, CRM API и MES API**

* Пороговое значение: определить, исходя из типа нагрузки на приложение (80% использования для CPU-bound, 40% – для IO-bound).
* Обоснование: Если CPU превышает 80%, сервис может начать обрабатывать запросы медленнее, что скажется на задержках и производительности.
* Действия при превышении:
  + Завести тикет в системе управления задачами (например, Jira), чтобы провести анализ производительности.
  + Добавить новый инстанс для распределения нагрузки (автоскейлинг).
  + Отправить уведомление DevOps-инженеру по почте или через мессенджер.
  + Если превышение продолжается более 15 минут, настроить автоматический вызов (например, через SMS или звонок) ответственному DevOps-инженеру.

1. **Использование памяти (RAM) для Shop API, CRM API, MES API, баз данных**

* Пороговое значение: 85% использования памяти.
* Обоснование: Когда память приближается к максимальной емкости, это может привести к снижению производительности или сбоям из-за нехватки ресурсов для обработки новых запросов.
* Действия при превышении:
  + Создать тикет на анализ проблем с памятью, определить, какие запросы или процессы потребляют ресурсы.
  + Добавить новый инстанс с дополнительной памятью (автоскейлинг).
  + Оповестить команду через почту и мессенджер.

1. **Очереди в RabbitMQ (queue\_totals.messages)**

* Пороговое значение: определить, исходя из данных по скоростям отправки и потребления сообщений.
* Обоснование: Большое количество сообщений в очереди означает, что один или несколько потребителей не успевают их обрабатывать, что ведет к задержкам в выполнении заказов.
* Действия при превышении:
  + Завести тикет для проверки состояния потребителей и их производительности.
  + Увеличить количество потребителей в RabbitMQ, чтобы разгрузить очередь.
  + Отправить уведомление DevOps-инженеру и владельцу бизнес-процесса, если очередь переполнена более 5 минут.

1. **Количество dead-letter сообщений в RabbitMQ**

* Пороговое значение: определить, исходя из данных по пиковым нагрузкам.
* Обоснование: Сообщения попадают в dead-letter, когда их не удается обработать корректно, что указывает на возможные ошибки в работе системы или нарушенные бизнес-процессы.
* Действия при превышении:
  + Завести тикет для анализа причин, почему сообщения не были обработаны.
  + Если количество сообщений продолжает расти, уведомить DevOps и команду разработчиков, так как это может указывать на сбой в сервисах, которые читают из очереди.

1. **Время ответа (latency) для Shop API**

* Пороговое значение: 1 секунда (в 3 раза больше времени ожидания, приемлемого для пользовательского опыта).
* Обоснование: Высокая задержка в API негативно влияет на пользовательский опыт и может привести к отказам клиентов.
* Действия при превышении:
  + Завести тикет для анализа источников задержки: проблемы с базой данных, перегруженный CPU или сеть.
  + Оповестить DevOps и команду разработчиков через мессенджеры.
  + Если задержка продолжается более 5 минут, автоматически добавить инстансы API для распределения нагрузки.

1. **Количество соединений с базами данных (Shop DB, MES DB)**

* Пороговое значение: 90% от максимального допустимого количества соединений.
* Обоснование: Когда количество соединений близко к максимальному, новые запросы могут отклоняться, что приведет к отказам в работе API.
* Действия при превышении:
  + Завести тикет для анализа активности, чтобы понять, что вызывает рост числа соединений.
  + Увеличить максимальное количество соединений в настройках базы данных, если это возможно.
  + Оповестить DevOps и владельцев бизнес-процессов.

1. **Количество HTTP 500 ошибок для Shop API, CRM API и MES API**

* Пороговое значение: 5% от общего количества запросов за 5 минут (либо определить эмпирически).
* Обоснование: Ошибки 500 указывают на внутренние сбои в системе и могут быть связаны с ошибками в коде или инфраструктуре.
* Действия при превышении:
  + Завести тикет для анализа логов и поиска причин ошибок.
  + Оповестить DevOps-инженера и команду разработчиков.
  + Если ошибки сохраняются более 10 минут, включить автоматическое масштабирование (добавить инстансы) и провести рестарт сервисов.

1. **Объем используемого S3-хранилища**

* Пороговое значение: 90% от выделенного объема.
* Обоснование: Когда объем S3 близок к полному, загрузка новых файлов может быть ограничена, что нарушит процесс работы клиентов и операторов с 3D-моделями.
* Действия при превышении:
  + Завести тикет на расширение хранилища или удаление старых данных.
  + Оповестить DevOps и бизнес-аналитиков о необходимости расширения.
  + Выполнить автоматическое расширение S3-хранилища (если поддерживается).