# **Задание 5. Кеширование**

## **Анализ системы в контексте планирования кеширования**

Учитывая, что "операторы жалуются на низкую скорость работы со страницей, а новых клиентов не устраивает скорость выполнения заказа" можно предположить, что улучшения производительности требует та часть MES, которая отвечает за управления заказами. Другая часть MES, отвечающая за расчёт стоимости заказа, жалоб не вызывает, но может влиять на производительность первой части, если используются общие ресурсы. Поэтому одна из первоочередных инициатив по устранению нежелательных ситуаций предполагает поднятие дополнительных инстансов под MES API и выполнение расчёта стоимости заказа только на этих инстансах.

Однако, производительность MES API применительно к управлению заказами также можно улучшить за счёт кеширования. Если выделение отдельных инстансов MES API под управление заказами не улучшит производительность сервиса, необходимо применить кеширование для ускорения основной и наиболее трудоёмкой операции для управления заказами — отображения новых поступивших заказов.

## **Мотивация**

Новые заказы, поступающие из очереди, по определению формируют выборку, необходимую операторам для быстрого приёма заказов в работу. Сохранение их в MES DB для последующей выборки даже с фильтрами и пагинацией создаёт дополнительную нагрузку на MES DB и негативно сказывается на времени выполнения запроса в MES API. Использование кеша для промежуточного хранения новых заказов позволит операторам сразу получать необходимую выборку в готовом виде без обращения к базе данных.

Внедрение кеширование решает несколько ключевых задач:

1. **Ускорение обработки заказов**: Заказ быстрее будет принят в работу оператором и успеет выполниться к сроку.
2. **Снижение времени ожидания в MES API**: Запрос на получение новых поступивших заказов будет выполнен быстрее, без обращения к базе данных.
3. **Снижение нагрузки на MES DB**: Базе данных не придётся выполнять запрос на фильтрацию и выборку новых заказов из всего существующего набора заказов.
4. **Улучшение пользовательского опыта**: Быстрое выполнение заказов напрямую влияет на качество обслуживания клиентов, сокращая недовольство клиентов.

## **Предлагаемое решение**

Предлагается использовать серверное кеширование, т.к. оно позволит иметь доступ к актуальной выборке данных большого объема сразу всем операторам.

1. **MES API** читает сообщения о новых заказах из **MQ** и записывает их в кеш, откуда они в соответствии с паттерном Write-Through попадают в **MES DB** со статусом MANUFACTURING\_STARTED.
2. Оператор запрашивает выборку новых заказов через дашборд **MES UI** и **MES API**.
3. **MES API** запрашивает новые заказы непосредственно из кеша и передаёт в **MES UI**.
4. Оператор выбирает заказ и берёт его в работу через дашборд **MES UI** и **MES API**.
5. **MES API** инвалидирует заказ в кеше по ключу и меняет статус заказа в **MES DB** на MANUFACTURING\_STARTED.

Решение предполагает использование паттерна **Write-Through**, т.к. запись новых запросов сначала в кэш, а оттуда в базу данных потребует наименьшего количества операций (сетевых взаимодействий).

* **Cache-Aside** НЕ подходит, т.к. он потребует постоянного обновления выборки через соответствующий запрос к базе данных, как при поступлении новых заказов, так и при взятии их в работу.
* **Read-Through** НЕ подходит, т.к. он тоже потребует постоянного обновления выборки через соответствующий запрос к базе данных, только запросы будет отправлять сам кеш.
* **Refresh-Ahead** НЕ подходит, т.к. он тоже потребует постоянного обновления выборки через соответствующий запрос к базе данных, либо вероятность получения неконсистентных данных будет очень велика (операторы будут часто брать один и тот же заказ).
* **Write-Behind** НЕ подходит, т.к. последующие точечные изменения статуса заказа будут выполняться MES API непосредственно в MES DB и могут произойти раньше, чем асинхронное создание заказа со статусом MANUFACTURING\_STARTED.

Решение предполагает использование стратегии **инвалидации по ключу**, т.к. она позволяет наиболее точно определить когда и какие данные в кеше можно считать невалидными.

* **Временная инвалидация** НЕ подходит, т.к. устаревание конкретных данных в кеше определяется конкретным событием, до наступления которого кеш сам является наиболее актуальным источником данных.
* **Инвалидация на основе запросов** НЕ подходит, т.к. для любых запросов, связанных с новыми заказами, кеш сам является наиболее актуальным источником данных.
* **Инвалидация на основе изменений** НЕ подходит, т.к. все изменения, касающиеся новых заказов, выполняются в первую очередь непосредственно в кеше.
* **Программная инвалидация** НЕ подходит, т.к. предполагает массовую инвалидацию кеша, тогда как эффективнее инвалидировать конкретные данные, если есть такая возможность.

В качестве кеша можно использовать Redis, развёрнутый в кластере. Redis предлагает различные структуры данных, позволяющие как получить весь набор элементов, так и добавить/изменить/удалить конкретные элементы в этом наборе, а также поддерживающий взаимодействие с PostgreSQL для реализации паттерна Write-Through.

Решение по кешированию отражено на схеме 5.1

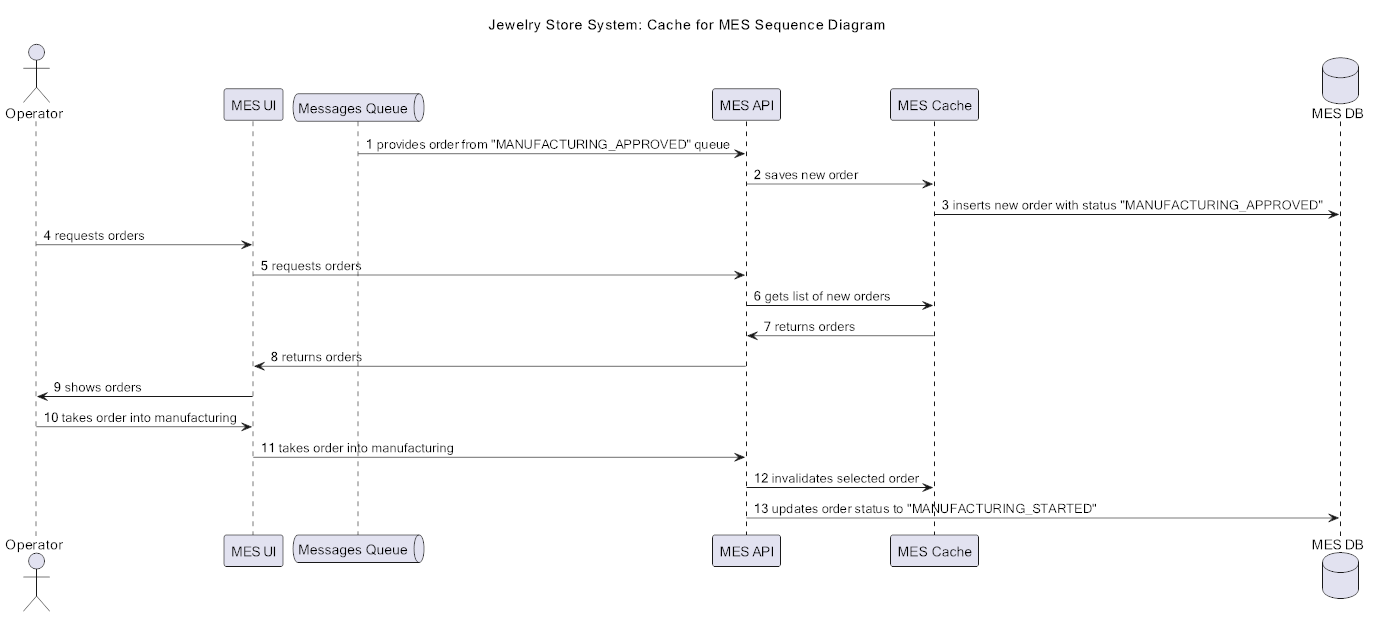


Схема 5.1 Диаграмма последовательности с MES Cache для системы Jewelry Store System