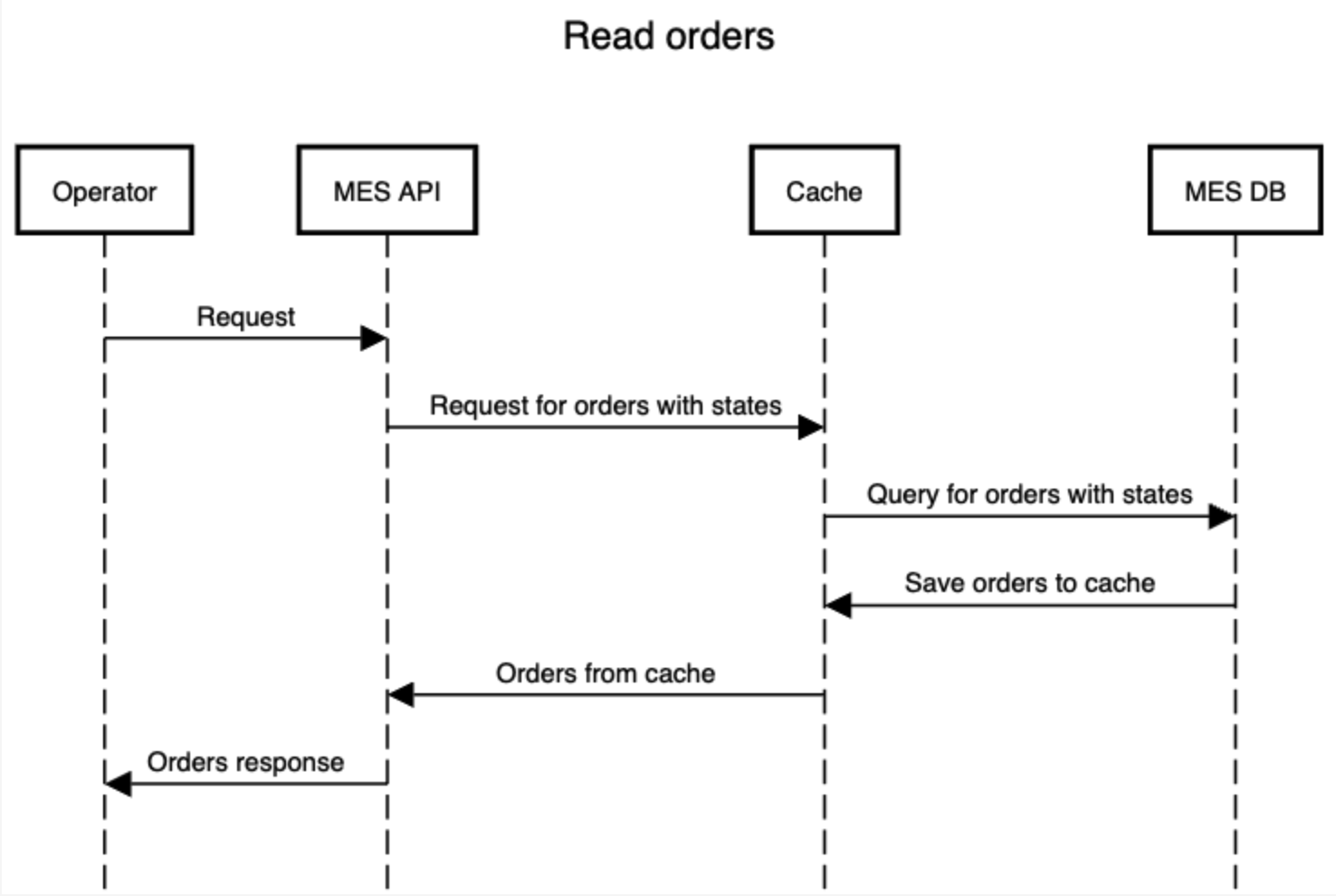
## 1. Мотивация

* По мере увеличения количества запросов к различным элементам системы время отклика может замедлиться;
* со стороны операторов:
  + операторы используют дашборд со статусами заказов для работы с подтвержденными заказами;
  + дашборд использует MES API;
  + чтобы уменьшить время ответа MES API на запросы операторов можно добавить кеширование для заказов;
* со стороны покупателей (customer):
  + при увеличении количества покупателей может возрасти нагрузка на Shop API и, соответственно, и на Shop DB;
  + чтобы избежать чрезмерных запросов к Shop DB и уменьшить время ответа Shop API можно внедрить кеширование для Shop API;
  + клиентское кеширование - для корзины покупателей - для товаров, добавленных в корзину имеет смысл клиентское кеширование;

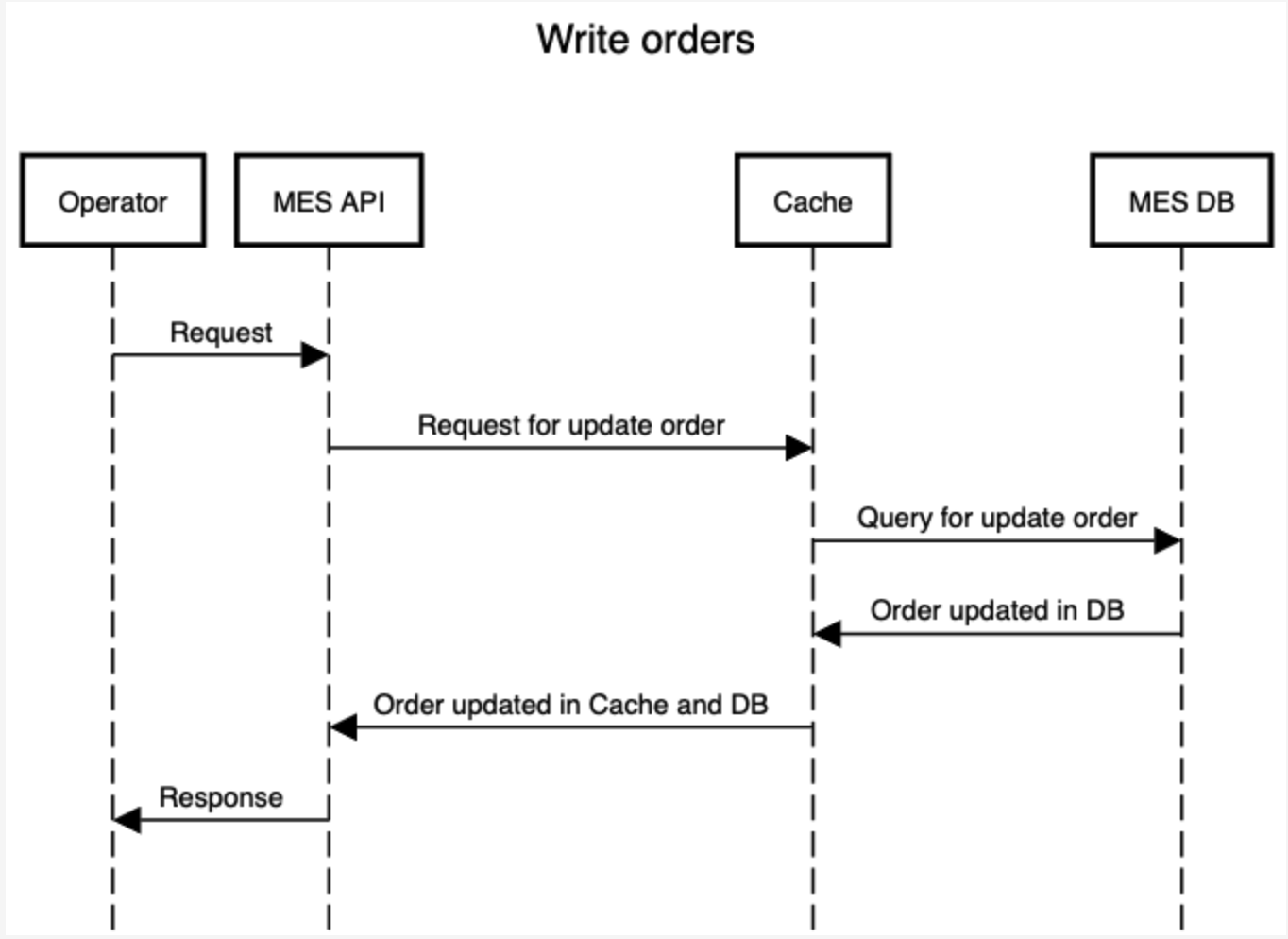
## 2. Предлагаемое решение

1. MES API - кеширование Read-Through и Write-Through:
   1. почему именно эти виды кеша:
      1. Read-Through - позволит получать данные с заказов не обращаясь к БД MES DB;
      2. Write-Through - так как статусы заказов обновляются и операторам необходимо понимать какой актуальный статус у текущего заказа, то при записи обновляется кеш, а кеш обновляет БД. Таким образом, при обновлении статуса заказа одним оператором второй оператор не сможет обновить статус еще раз;
   2. почему другие не подходят:
      1. Write-Behind - работает быстрее, но нужно будет накручивать логику вокруг кейса, когда один оператор обновил статус “Заказа1” и после него второй оператор тоже обновил статус “Заказа1”. Нужно будет думать над блокировками, но в текущем виде нет предпосылок к тому, что MES API будет высоконагруженным, поэтому обновлять данные в кеше и в БД можно синхронно (используя Write-Through)
      2. Другие Read-паттерны - Cache Aside - несогласованность в кеше и в БД, Refresh-ahead - в сложность в реализации, которая для нас избыточна;
2. Shop API:
   1. клиентское кеширование + инвалидация при заказе товара:
      1. таким образом, после оплаты/подтверждения заказа кеш будет инвалидироваться, но до тех пор клиент будет хранить свою корзину товаров локально - на своей стороне
   2. кеширование Refresh-Ahead + Write-Behind для товарных остатков:
      1. чтобы каждый запрос клиентов не приводил к запросам в БД можно добавить кеширование;
      2. почему именно эти виды кеша:
         1. Refresh-Ahead и Write-Behind являются асинхронными подходами к кешированию, таким образом значительно уменьшается количество операций с БД;
      3. почему другие не подходят:
         1. другие виды кешей являются синхронными и могут привести к замедлению времени ответа, что приведет к ухудшению пользовательского опыта;

* **Sequence Diagram** для операция чтения списка заказов - это диаграмма отображающая взаимодействие операторов с MES API



* **Sequence Diagram** для записи об изменении статуса заказа - это диаграмма отображающая взаимодействие операторов с MES API



* **Стратегии инвалидации кеша** - одновременное использование двух инвалидаций: по ключу и временная инвалидация:
  + временная инвалидация - подходит из-за того, что необходимо очищать кеш от неиспользуемых данных;
  + инвалидация по ключу - нужна, чтобы обновлять данные в кеше при update-запросе статуса заказа
* другие стратегии инвалидации меньше подходят:
  + инвалидация основанная на запросах - более масштабная инвалидация, чем требуется для нашей задачи в MES API;
  + инвалидация на основе изменений - нам не требуется полная инвалидация всех данных при обновлении - нам достаточно обновить только определенные заказы;
  + программная инвалидация - позволяет добавить дополнительную логику, но нам в нашей задачи для MES API это не нужно;