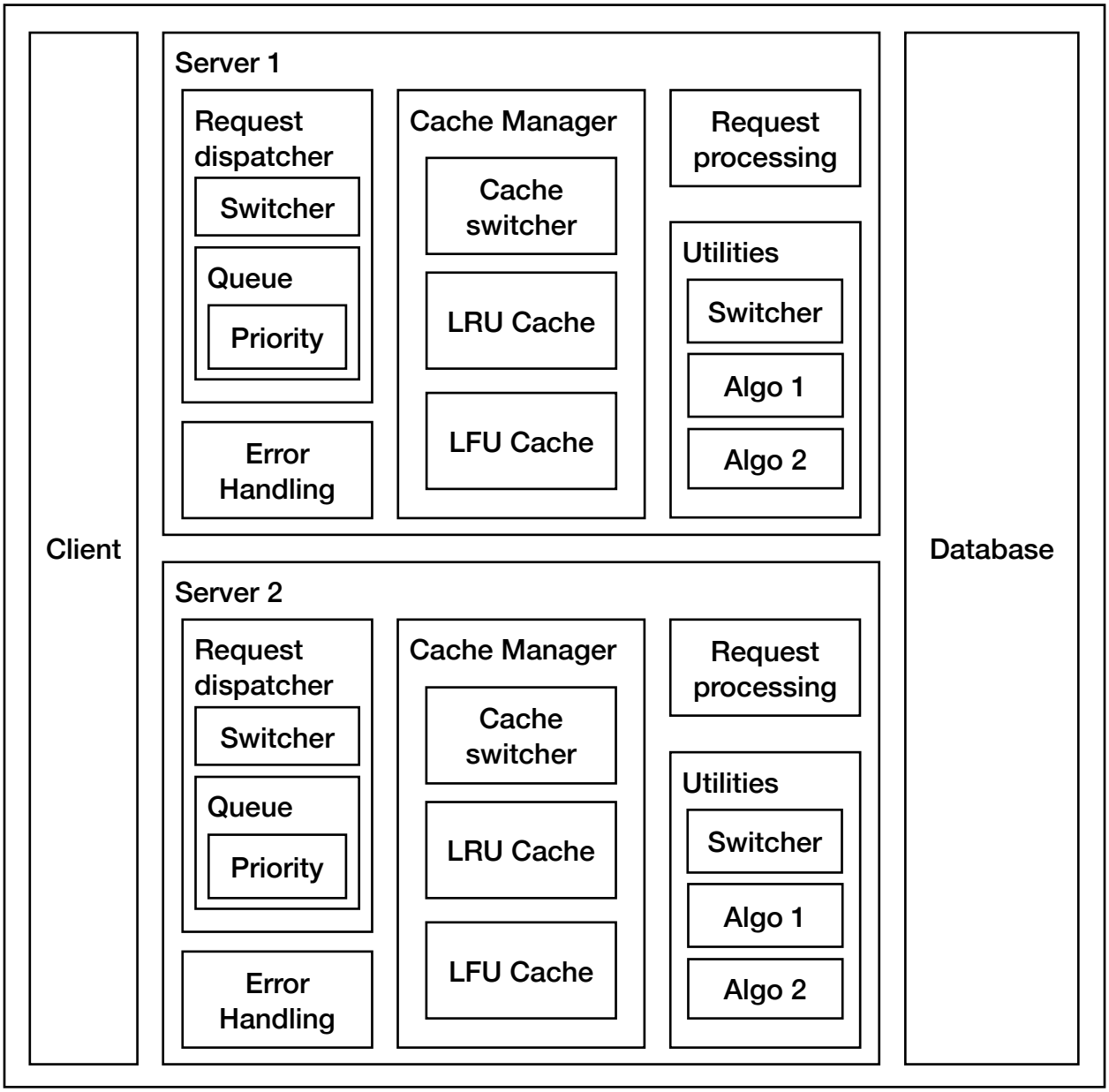
**Задача 5.** а) Разширена диаграма на декомпозиция на модулите




Диаграмата на декомпозиция на модулите ни разкрива, че системата се придържа към клиент-сървър архитектурата. такъв вид архитектура може да се претовари, ако много клиентски приложения пускат едновременно множество заявки към сървъра на приложението и по този начин образуват bottleneck – честотата на постъпващите заявки е значително по-голяма от честотата на получаване на отговор на заявка. Има множество стратегии за осигуряване на устойчивост към отказ от претоварване на системата, но най-добрите от тях използват и комбинират няколко от тях едновременно.

Request dispatcher очевидно е сървърният модул, който приема клиентските заявки. Да допуснем, че клиентските заявки образуват bottleneck. За да се справим с този проблем добавяме и правим следните модификации:

1. Добавяме подмодул Queue и Switcher в Request dispatcher модула.
   - В опашката ще се складират всички постъпващи заявки от клиентски приложения, за да гарантираме, че няма да остане такава без отговор от сървърното приложение. Queue ще съдържа подмодул Priority, който ще задава приоритета с който ще се взимат заявки от опашката. Опасностите тук са две:
     - Силна опашка: ако опашката е без приоритет или казано по друг начин – за приоритет използва времето на постъпване на заявката, то тогава няма да има

процеси, които ще гладуват, само ако времето за отговор на всяка заявка е нормирано. Тоест няма заявки, който отнемат много повече време от други.
    - Слаб семафор: ако опашката използва някакъв друг приоритет, при нормирано време за всички заявки, може да настъпи гладуване и един тип заявки да чакат отговор твърде много време за сметка на друг тип заявки.

      Валидна стратегия е за приоритет да се използва време за отговор на заявка, което е взето от динамично направена статистика. За нея ще трябва да се обособи нов специален модул, който прави тази статистика и по точи начин на заявките, които отнемат много малко време за отговор ще се отговаря още по-бързо и ще остава повече време за обработка на заявките, които така или иначе отнемат значително количество време за отговор.

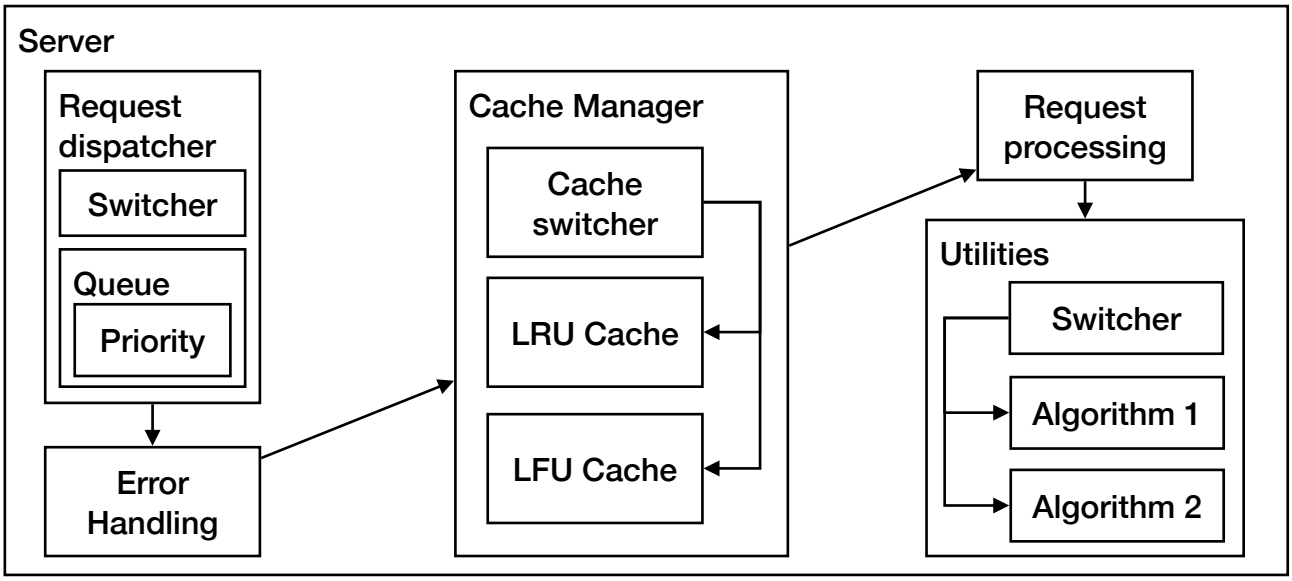
- Switcher-а може да прехвърля заявка, взета от опащката до друг сървър, като симулира заявка от клиент. Другият сървър няма да знае дали тази заявка идва от клиентско приложение или от сървър (за него ще е все едно). Този модул ще позволи лесно мултиплициране на сървърите. Логиката в него ще се базира на броя заявки, които са постъпили в момента и поради тази причина, той ще има достъп до размера на опашката във всеки един момент (дори може да получава информация за това при настъпване на някакво събитие като фиксиран лимит, дни в които се очаква масово използване на приложението (черен петък, празнични дни и т.н.) и други).

2. Добавяне на кеш в сървъра.
    - Нов CacheManager модул ще се грижи за забързване на процеса по извличане на отговор от сървъра. Кеш-а ще разполага с два вида стратегии, по които може да функционира и ще може да превключва между тях на база на някаква статистика за представянето на сървъра (която пак може да се извлича от друг модул, който да осигурява тази статистика). Двете стратегии ще са:
        - LRU (Least Recently Used) – от кеша се изтриват отговорите на най-НЕ наскоро постъпилите заявки при достигане на определен размер на кеш-а;
        - LFU (Least Frequently Used) – от кеша се изтриват отговорите на най-малко генерираните попадения заявки при достигане на определен размер на кеш-а;

      Това е много силна, широко разпространена и лесно приложима стратегия за спряване с отказа при отказ от претоварване. При извличане на отговор на заявката – първо се проверява за вече генериран отговор в кеша и ако такъв няма, чак тогава се извършват изчисления и се допитва до базата данни за да се създаде отговор, който може да се запази в кеша. При попадение в кеш-а ще се връща много по-бързо заявка, отколкото ако се процесира и допитва до базата данни.

3. Разширяване на Utilities модула, по начин, по който го прави динамично конфигируем.
    - Добавяме нови подмодули, в който са реализирани различни алгоритми, които ще отговарят за обработката на различни заявки с цел ускоряване на процесите по генериране на отговор.

б) Диаграма на структурата на употреба на модулите




в) Диаграма на структурата на внедряването