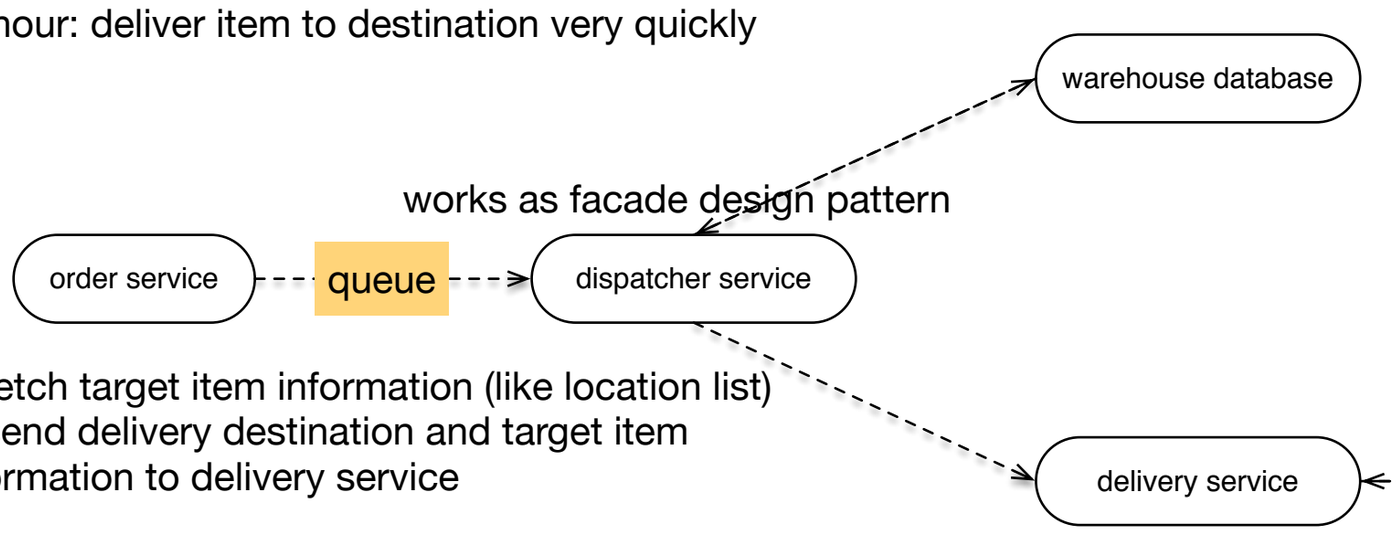
One-hour delivery system

Requirements:
1. one-hour: deliver item to destination very quickly



DB schema:
item: id, name, a list of {warehouse_id, quantity}
warehouse: id, location
trunk: id, name, location, status

1. fetch target item information (like location list)
2. send delivery destination and target item information to delivery service

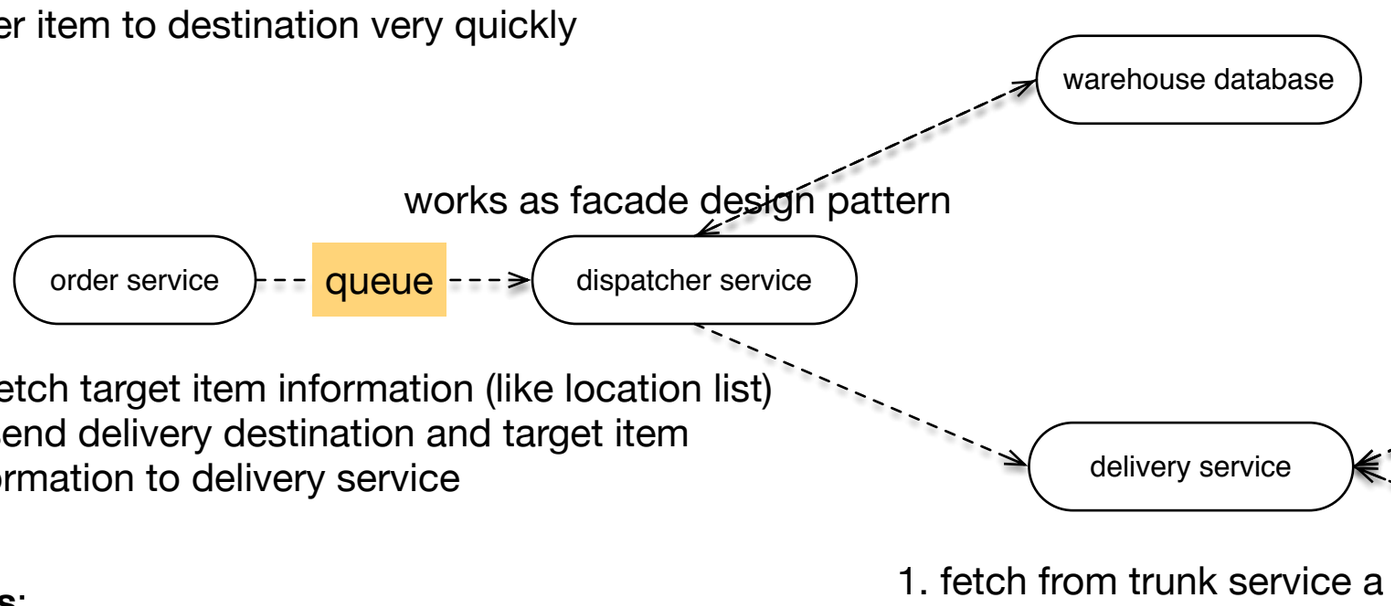
1. fetch from trunk service a list of available nearby trunks
2. find an optimal solution for picking up a trunk to delivery an target item from a warehouse to destination

1. add a trunk
2. remove a trunk
3. find nearby trunks given a location

Scenarios:
1. a large volume of orders - how to scale?
*** queue - asynchronous**
***** load balancer and service instances - add a load balancer between dispatcher service and order service. In the perspective of high availability and fault tolerance, we can deploy multiple instances for services.
***** improve data read (or/and write) efficiency - cache, multiple-master-multiple-slave database like mongoDB architecture, considering that the warehouse database could be very large (warehouses and items)
  * considering Amazon is a world wide company, there must be hundreds and thousands of warehouses all over the world. In the perspective of geography, we can do partition/sharding for warehouse database (warehouse and items)
2. how to schedule delivery - given 2 lists of locations of items, trunks and one destination?
  1. To ensure we can deliver within one hour, we can factor in the distance of delivery. Sort the list of locations of the target item based on the distance between warehouse and destination in ascending order. Find the one with shortest distance. Then assign an available trunk that is most close to the warehouse to deliver the item to our client.
3. what you can do to guarantee that delivery is within one-hour in the perspective of software?
  1. algorithms should not be simple. There could be multiple delivery strategy patterns to be used depends on different scenarios.
  2. the system should be **high available** and low latency!

Amazon Locker system

Requirements:
1. deliver item to destination very quickly



DB schema:
item: id, name, a list of {warehouse_id, quantity}
warehouse: id, location
trunk: id, name, location, status

1. fetch target item information (like location list)
2. send delivery destination and target item information to delivery service

1. fetch from trunk service a list of available nearby trunks
2. fetch from locker service a list of nearby locker (Top k closest)
3. find an optimal solution for picking up a trunk to delivery an target item from a warehouse to a locker
  1. criteria one: locker should be as close to client as possible.
  2. criteria two: locker candidate should have enough empty containers for delivery man to drop the package

我那时候一开始考虑也想是不是Order生成的时候就匹配了一个Locker，还有挑选哪个Location,但跟面试观官交流以后，他就说假设只有一个Location然后主要想知道送货小哥去的时候怎么分配，别的先不考虑。

Scenarios:
1. how to assign a container to a package in a locker

a list of containers, a package

```
class Container {
    int size; // int height, width, length;
    boolean available;
}

class Package {
    int size; // int height, width, length;
}
```

1. get a list of available containers
2. sort the list
3. binary search - search insert index
  1. binary search height of package on the entire list -> first valid container
  -> map.get() returns two lists of containers. Heights of containers in each list >= firstOne. These two lists are sorted in width (listA) and length
  2. binary search width of package on the listA -> first valid container -> map.get() return two lists of … length (listB) -> find that one.
  3. binary search length of package on the listB

Map<Height_Length, List_of_containers>
* List: container.height >= Height_Length, and sorted by width

Map<{Height_Length, Width_Length}, List_of_containers>
* List: container.height >= Height_Length,
and container.width >= Width_Length
and sorted by length