### **DESIGN STRATEGY**

Define a strategy to decide when to hit the Rapid API web service or Database while fetching the covid data.

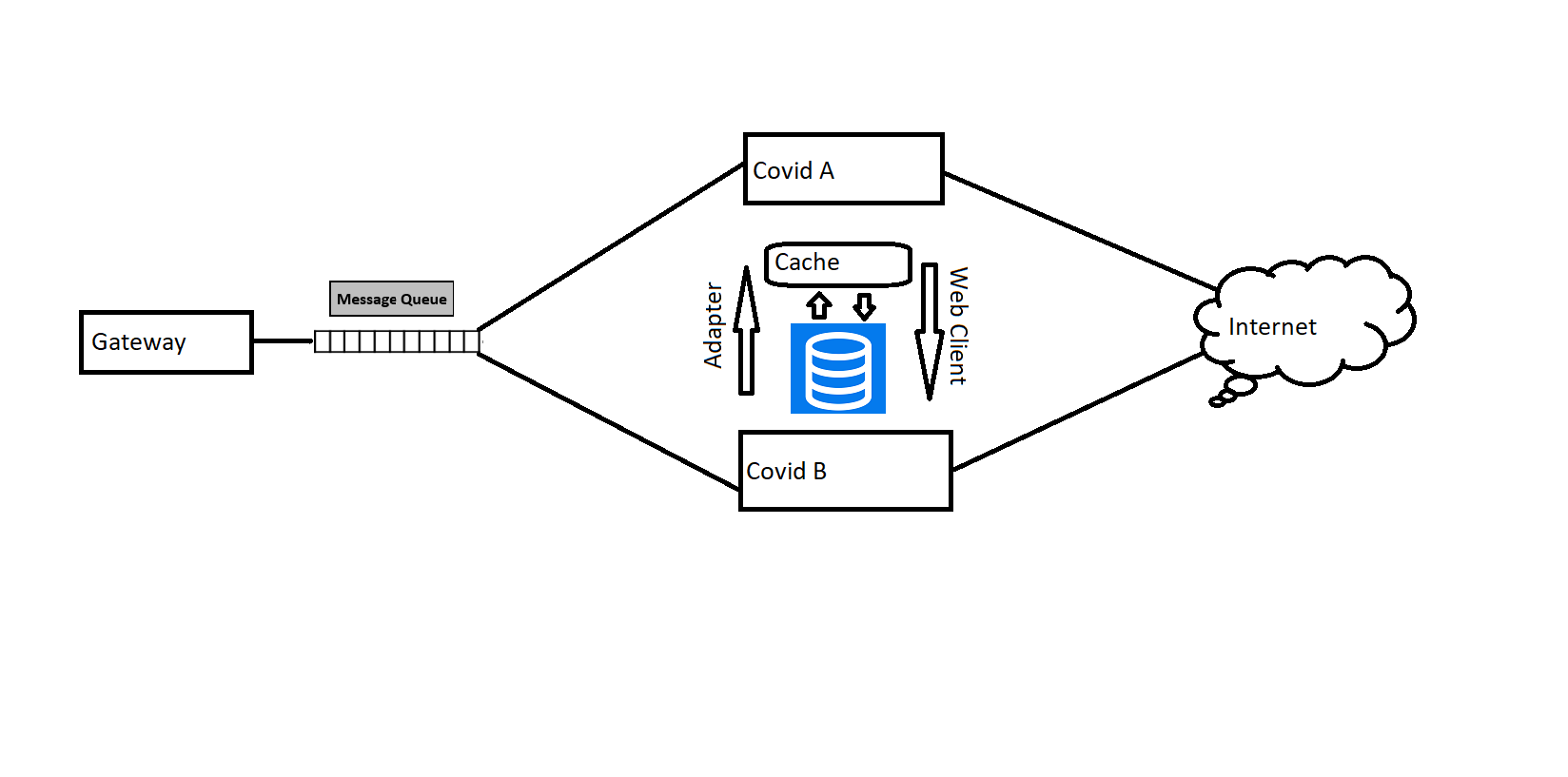
It depends upon the frequency of requests coming to the application. To decrease the number of API calls or database hits, we can implement caching.

We can implement application layer cache directly on the request layer. This way every time a request is made, it will quickly return the cached data, if exists.

If not found in cache(any level) then it will hit the database and if found it will update the cache and return the result. If not found in either, then it will make the API call.

We also need to decide on cache policy like LRU. Suppose if we are fetching the results for countries by name. Then we store them in cache with a timestamp or number of times it is requested. This will help us in cache eviction.

### **NEW REQUIREMENT**



Following are the details:

**Design pattern**

To integrate with one more web service having different format, we can use a combination of factory and Adapter design pattern. This way we can have factory pattern for compatible API’s and adapter pattern for different classes to work together even if they have different interfaces.

**Interaction layer and Caching**

Both API’s will have different controllers, and the API’s will interact with each other using RestTemplate or web client. We can have different or same repository for them, depending upon the requirements.

We will also need to implement both Global and local cache.

**Messaging Queue**

If we have many requests coming, then we can also use rabbitMq. In this, we can have a gateway microservice that’ll publish the event (request) on queue to which both the API’s (new and existing) will subscribe to. They will fetch the response and after necessary conversion using adapter pattern will show back the result.

**Fallback mechanism**

Suppose the client makes a request, the result is not found either in cache or in database. Then request will make an API call, considering the call would fail (service is down), the request will go to the other API that we integrated. However, this new API of ours has different request/response body, so we will use adapter pattern. Here the client would be our existing API and adaptee would be our New API.