Xendit - Caching strategy

Assumption:

The Marvel characters will be never deleted, They only add new heroes only.
The Marvel APIs limit maximum 100 records per request.
In the response of /characters API they provided:

- **offset** (int, *optional*): The requested offset (number of skipped results) of the call.,
- **limit** (int, *optional*): The requested result limit.,
- **total** (int, *optional*): The total number of resources available given the current filter set.,
- **count** (int, *optional*): The total number of results returned by this call.,
- **results** (Array[Character], *optional*): The list of characters returned by the call.

Caching Strategy

At the first API call of /characters, we will get all the character IDs and store them in our application cache.
I'm using flat-cache lib to store the cache data to the file, so meaning when you restart the server but the cache still can load from the file.

I use 2 caches file to store the data, one for the list of character IDs and total record (***heroesCache***), another for storing the character details (***heroesDetailCache***)
It is how we handle the case that Marvel adds new heroes:

- I set an expired time for our cache (the time can confinable on .env) and stored to our ***heroesCache*** cache.
- From every **/characters** API call, I will do the check:
    - Checking the ***heroesCache*** cache, if no data on our cache then call the Marvel API to get it (I use some algorithm to divide the offset and make the async to get all characters from Marvel), store the ***list of characters IDs*** and the ***total of characters*** into ***heroesCache*** cache, in parallel I stored all character information into ***heroesDetailCache*** for each character, then send characters IDs to the client
    - If there is the data on the heroesCache cache, then we need to check the expired time, if it is not expired then we can get the data from the cache.
    - If there is the data on cache but the duration is expired then we need to check::
        - step1: I will call an API to get the information from Marvel (**/characters**), In their response, they have the ***total*** number of all characters and the number of how many data returned back (***count***) on this call.
        - step2: I will check the *margin* between the number of totals stored in our cache and the total return on the step1's API. If there is the margin here => yes, there is the new heroes added to Marvel, we need to update our cache:
            - If the ***margin*** is **less than or equal** to the **limit** of the record per request. meaning in the API call of step1 we are getting all the new changes heroes. Then we just need to **append** the new characters into our cache, update the total characters in our cache, then send back data to the client
            - If the **margin** is **greater than** the **limit**=> meaning the API on step1 can't getting all the new characters. We need to get all the character with the offset to make sure it is not getting the characters that we already stored in our cache *heroesCache*
    - If there is the data on cache but the duration is not expired then we just send the data of the cache to the client.

- From every **/characters/{characterId}** API call, I will do check:
    - Checking **heroesDetailCache** cache,
        - If no data on our cache then call an API to Marvel to get the Hero details and store to ***heroesDetailCache*** *then* send back to the client
        - If there is data from the cache, then return data to the client