Design Exercise

The following is a requirements document for a web service that we'd like to build. It is meant to serve as a starting point for a design discussion. Please read the following, take a few minutes to understand the requirements and think about it, and we will discuss implementation options in person. There is no need to write anything (unless you wish to jot down notes for yourself).

Background

A *stock*, for the purpose of this exercise, is a product that users can buy or sell and has continuous price updates. Stocks have a symbol (name), e.g. `IBM`, and a current price, e.g. $120.4.

A *stock basket* (or just *basket*), for the purpose of this exercise, is a collection of stocks with quantities. For example, a basket can include 30 `IBM` stocks, 50 `MSFT` stocks and 10 `GOOG` stocks. The *current price* of a basket is the total cost of the stocks in it. For example, if the price of `IBM` is $120.4, the price of `MSFT` is $110.8 and the price of `GOOG` is $1080, the price of the basket in this example is 30 * 120.4 + 50 * 110.8 + 10 * 1080 = $19952.

However, it is often more interesting not to look at only the latest price for this calculation, but rather an average price over a time period, e.g. the average price of `IBM` over the past hour, the average price of `GOOG` over the past hour etc.

Baskets Service

In this exercise you are given the task of designing a **baskets web service**. This service will be used by clients to define baskets that they want to track and get their updated price. The baskets service needs to have the following REST API:

```
/baskets POST - define a new basket and save it persistently.
    PARAM name: custom name of the basket
    PARAM stocks_and_quantities:
        [
            {
                "stock": ...,
                "quantity": ...
            },
            ...
        ]
    Returns a persistent ID of the new generated basket.

/baskets/{id}/total GET - gets the current price of a basket by its
persistent ID. Prices are averaged over the past hour.
```

To be able to calculate the current price of a basket, our service needs updated prices for all stocks. For this, assume we have a provider sending to us over WebSocket a feed of the following form:

```
[
    timestamp: "2018-10-05T14:31:12.482917",
    {
        "stock": "IBM",
        "price": 120.4
    },
    {
        "stock": "MSFT",
        "price": 110.0
    }
]
```

The WebSocket feed provides continuous updates, so a moment after the example above, another message may arrive with an update for stock `IBM` for example.

Your task is to design this service, taking the following points into account:

1. How do we listen to the price updates and where do we store them? Take into account there are thousands of stocks with thousands of updates per second each **(you can assume there are 10,000 stocks with average of 10,000 updates per second per stock)**
2. How do we represent the baskets, how do we store them?
3. How will you implement **`GET /baskets/{id}/total`** efficiently?
   To clarify, this needs to calculate the price of the basket (sum of price * quantity for all of the basket's stocks), when 'price' is the average of the stock's price in the last hour.
4. How will this all scale to any number of concurrent users?

Reminder: You do not need to write anything right now. Just read and think it through. This is a base for discussion we'll do on a whiteboard.