Pricing Data API

Overview

Build a service “**PricingDataAPI**” based on **NodeJS / Express**. The service will be responsible for providing the access to a pricing API. With help of the API, 3rd-party applications can retrieve an actual pricing data in the **JSON** format (can be chosen at will).

NOTES:

- Except “Express” framework there is no other limitations on dependency packages to use. Feel free to choose your best known or favorite tools;
- “**Advanced**” blocks are not mandatory, but a plus;

API

“GET /pricing?coupon”

The service must include the endpoint “**/pricing**” which 3rd-party applications can access using a **GET** request over HTTP protocol. Response must be in JSON format:

```
{
   “couponName” : ””,
   “pricingData”: {},
   “createdAt”: “ISO8601 format”
}
```

The endpoint must include an optional parameter “coupon”. Pricing data should be shown according to a coupon attribute’s value, if it specified. See: Database section.

Limitations:

This endpoint must considered as “highly loaded”, therefore required a caching strategy implementation in order to prevent a frequent reading of the database. Let’s assume that each request w/ reading from the database would take over 250ms, when the average request rate is >250 req/sec.
Suitable cache solution must be found. Ex: redis, LRU-cache etc..

Advanced: Since this endpoint is public and can be called by any 3rd-party application without additional authorization, the endpoint must implement a solution that will prevent it from potential DoS attack, where “hacker” can issue requests with auto generated coupon names thereby trigger a database reading all the time.

Database

Utilize MongoDB database. Pricing data stored in “**PricingData**” collection with following schema:

```
{
  couponName: String,
  pricingData: Object,
  createdAt: Date
}
```

“**coupon**” - as a string value. Ex: “winter-2018”
“**data**” - a pricing data. Large JSON object. (structure doesn’t matter)

Pricing data object is unique to each coupon. The collection should also provide a one record w/o a coupon. This record must be fetched when no **coupon** parameter found in a request.

“**Coupon**” collection stores coupon names. Schema:
{
  “Name”: “String”
}

Logs

Logs are essential. Suitable solution must be found in the npm library. Ex: morgan, winston, node-bunyan etc..
Advanced: Ability to see an endpoint time execution

Unit tests

Service must be supplied with ability to execute unit tests by running “**npm run test**” command in the terminal. Unit tests should cover only essential logic.
Advanced: Ability to see a coverade (not mandatory)

Start Service

This service must be ran by simply type “**npm run**” command in the terminal.