Recruiting Challenge

For Tech Lead

Introduction

When evaluating your response to the challenge, we will look at:

1. how you have solved the tasks
2. how you structure your code
3. which solutions you have chosen to solve the different task
4. We will also appreciate seeing examples on how you would ensure that your code works as intended
5. Ideally, you will use Typescript and NestJS framework. However, feel free to use any framework that best displays your skills.

Be creative and feel free to extend the tasks with additional functions, for example for sorting, filtering or searching in data.

As the cases only contain a limited amount of details, feel free to specify any additional assumptions, which are relevant for your answers.

Your code and written answers will be used for our internal evaluation and the subsequent interview.

Please submit a link to a GitHub repository containing your code. The repository should include a README file with instructions on how to run the application.

Task

Let's imagine you play a game where you can earn scores.
With scores, you can then buy some products in the game or transfer them to your bank account.

Whether you earn, spend, or request a payout, a new transaction is created in the transaction service that is available by API.

Write an MVP of data aggregation microservice that will collect transactions from transaction API and expose its own API endpoints:

1. Get aggregated data by user Id: balance, earned, spent, payout, paid out
2. Get list of requested payouts (user ID, payout amount), if there are several payouts requested by a user, then the amount must be aggregated into one.

Pre-conditions:

1. Service will have millions of requests per day
2. Data must be up to date with less than 2 minute's delay

3. You have limited access to transaction API (5 requests per minute, with limit 1000 transactions)
4. You can use NestJS or any other framework
5. You can mock transaction API entirely so that we can run your app
6. Exchange rate is 1 SCR = 1 EUR

Transaction API

GET /transactions?startDate=2023-02-01 00:00:00&endDate=2023-02-01 00:00:00

Response:


```
{
  "items": [
    {
      "id": "41bbdf81-735c-4aea-beb3-3e5f433a30c5",
      "userId": "074092",
      "createdAt": "2023-03-16T12:33:11.000Z",
      "type": "payout",
      "amount": 30
    },
    {
      "id": "41bbdf81-735c-4aea-beb3-3e5fasfsdfef",
      "userId": "074092",
      "createdAt": "2023-03-12T12:33:11.000Z",
      "type": "spent",
      "amount": 12
    },
    {
      "id": "41bbdf81-735c-4aea-beb3-342jhj234nj234",
      "userId": "074092",
      "createdAt": "2023-03-15T12:33:11.000Z",
      "type": "earned",
      "amount": 1.2
    }
  ],
  "meta": {
    "totalItems": 1200,
    "itemCount": 3,
    "itemsPerPage": 3,
    "totalPages": 400,
    "currentPage": 1
  }
}
```


Describe Testing Strategy

Please describe the optimal strategy for quality assurance of an application consisting of a backend as in the above example.

How would you have implemented automatic testing using a TDD approach if you have had more time?

Leadership Case: Technical Alignment

Different senior developers have opposing strong opinions about handling queues in our microservices as we are trying to choose one way. In your experience, what are the typical ways queues are handled, and how would you help the Engineering department align and move forward?