.NET Interview Task – Microservices with PostgreSQL, preferrably

Task description

The system simulates a stock trading environment where users can place orders and their portfolios are managed in a distributed manner. There should be seamless and constant communication between the stock order processing and portfolio management components. There should be a periodic task to keep track of the latest stock prices in the user's portfolio.

There should be three different projects (microservices) independent from each other:
Order, Portfolio and Price.

Order:

1. Expose an endpoint POST - “/api/order/add/{userId}” that allows clients to place orders by sending JSON payloads containing order details:
- Ticker (e.g. Stock tickers like AAPL, TSLA, NVDA or whatever…)
- Quantity
- Side (buy/sell).

We don’t expect you to make login, registration and authentication functionalities, but everytime a request is made with a new userId we just save it in the system that user X made an order.

2. Upon receiving a stock order, use the latest price received from Price Service to simulate execution of the order and store the order details in a PostgreSQL database.

3. Notify the other microservices that an order has been filled and the order execution details.
It would be best to NOT do this via HTTP, but instead use other solutions for eventing.

Portfolio:

1. Should listen for events from the other microservices.

2. When a new event comes from the OrderService, process the stock order details and create or update a user’s account value (portfolio) in the database based on the quantity and price of order execution.

3. Subscribe to updates from the PricesService.

4. Update the user’s portfolio based on the received prices.

5. Expose an endpoint for fetching the user’s portfolio (the response can only contain the user’s id and his current portfolio value).

Price:

1. Implement a service which generates random prices every second for the stocks you have in the system.

2. Notify the other microservices everytime a new price is generated.

Here is the summarized flow:

1. A user (id) sends an order request via POST endpoint.

2. The Order Service processes the order using the latest price from PriceService.

3. The Order Service notifies the other services that an order was executed.

4. The Portfolio Service creates or updates, if it exists, the user’s portfolio.

5. Every time a new price is generated from the Price Service, the Portfolio Service must update the user’s portfolio accordingly.

6. There should be an endpoint for fetching the portfolio value at any given time so that the user can see what is his portfolio value.

Submission:
Share your code repository on GitHub.