Access Key Management and Token Information Retrieval System

Microservice 1: Access Key Management Service

Objective: This service is responsible for generating access keys for users, defining rate limits, expiration times, and providing administrative capabilities to create or delete keys. Users can retrieve their plan details using their key.

Core Features:
Key Generation: Allows admins to generate access keys for users, specifying request rate limits per minute and key expiration time.

Admin Commands: Simple commands for administrators to create, delete, and list keys. Also, functionality to update the rate limit or expiration of existing keys.

User Queries: Enables users to fetch the details of their access plan (rate limit, expiration, etc.) by making an API call with their key and disable the key.

NOTE: Admin auth happens with JWT token, but there is no need to have auth system

Microservice 2: Token Information Service
Objective: This service allows users to fetch token information based on their access keys. It respects the rate limits and expiration times defined in the keys generated by Microservice 1.

Core Features:
Token Information Retrieval: Provides an endpoint to fetch token information. The response should be a mock or static data, as the focus is on access control rather than the data itself.

Rate Limit Enforcement: Checks the user's request rate against their key's rate limit. If the rate is exceeded, it returns an error response.

Key Validation: Verifies if the key is valid and has not expired before allowing access to the token information.

Logging: Optionally, logs each request for token information, including key used, timestamp, and whether the request was successful or rate-limited.

Technologies:

- The only limitation is to use nest.js. besides that you can choose technologies and build the architecture.
- Bonus points for having tests

Hints:
For optimization you can use data duplication. There is no need to have any http/grpc requests in between the micrservices.
For communication you only need event streaming. Even redis pub sub is good enough if you haven't worked with any before.

Design

Requirements:

Functional:

1. Allow admin to create, view, and delete keys
2. Allow admin to set expiry and rate limit for the key
3. Allow users to disable the token*
4. Allow users to view token info

Non-Functional:

1. Info should be accessible based on expiry and rate limit defined for the given token
2. Info should be accessible only using the active token

—----------

Token Generation:

1. For token validation, we should avoid storing and comparing the token. Rather we should derive the information from request token
2. JWT is the suitable option for this as we can include user details as **claims**, which will help in easy validation. **-** stateless behavior
3. Assign a **role** for the token. Eg User shouldn't be able to generate token.

—----------

Microservice 1: Access Key Management Service handles commands
Microservice 2: Token Information Service acts as **View Module** and handles user queries

How should the two microservices interact?

- **Request-Response Pattern** - **User view** on every API request sends a validation/query request to the **Token Management** service for token info**.**

  It can be implemented using separate events, but in this case, event-based communication seems overkill. **"In event-based communication, events are business facts that have value to more than one service and are worth keeping around"**

- **CQRS Pattern** must be followed.

In CQRS pattern the admin/user commands trigger some synchronous operation, which will generate events for User View. The User View processes the events and updates the token state.

Approach-1: Manual

Token generation microservice manually triggers events after successfully performing CRUD operation (updating the token state). The User View service processes the triggered events and updates the internal state.

Approach-2: **C**hange **D**ata **C**apture

Rather than manually triggering the event, the Token Management service only updates the database state. The User View service monitors the database events and accordingly updates its state. **(Seems to be better approach) (Solves dual write problem)**

Token Properties:

1. UserId
2. Token (used by token management service and not by user view)
3. Role
4. RateLimiter
5. Expiry
6. CreatedAt / IssuedAt
7. isDeleted

UserId and CreateAt combination can be used to check if req token has expired or not.