Desafio de segurança e performance

Abaixo neste documento, estarei deixando a descrição do desafio, apresentarei quais são seus objetivos e também irei descrever como irei resolver cada situação proposta, contemplando as tecnologias utilizadas e os processos envolvidos.

Descrição do problema:

1. Armazenamento

Vamos supor que existam três grandes bases de dados externas que organizam nossas informações. A primeira delas, que chamamos de Base A, é extremamente sensível e deve ser protegida com os maiores níveis de segurança, mas o acesso a esses dados não precisa ser tão performática.

A segunda, é a Base B que também possui dados críticos, mas ao contrário da Base A, o acesso precisa ser um pouco mais rápido. Uma outra característica da Base B é que além de consultas ela é utilizada para extração de dados por meio de algoritmos de aprendizado de máquina.

A última base, é a Base C, que não possui nenhum tipo de dado crítico, mas precisa de um acesso extremamente rápido.

2. Tráfego

Cada uma das bases existentes, são acessadas por sistemas em duas diferentes arquiteturas: micro serviços e nano-serviços. Vale salientar que essas bases de dados são externas, Quantos aos payloads retornados por esses recursos, pode usar sua criatividade e defini-los, imaginando quais dados seriam importantes de serem retornados por sistemas como esses.

O primeiro sistema, acessa os seguintes dados da Base A:

- CPF
- Nome
- Endereço
- Lista de dívidas

O segundo, acessa a Base B que contém dados para cálculo do Score de Crédito. O Score de Crédito é um rating utilizado por instituições de crédito (bancos, imobiliárias, etc) quando precisam analisar o risco envolvido em uma operação de crédito a uma entidade.

- Idade
- Lista de bens (Imóveis, etc)
- Endereço
- Fonte de renda

O último serviço, acessa a Base C e tem como principal funcionalidade, rastrear eventos relacionados a um determinado CPF.

- Última consulta do CPF em um Bureau de crédito.

• Movimentação financeira nesse CPF.
• Dados relacionados a última compra com cartao de crédito vinculado ao CPF.

3. Disponibilização dos Dados

Agora que os dados desejados já foram consumidos, processados e armazenados, é necessário que eles sejam disponibilizados. Será necessário também desenvolver um meio pelo qual esses dados estarão disponíveis. É interessante imaginar os possíveis interessados em consumir esses dados para que uma única solução possa ser construída de modo a atender o máximo de situações possíveis.

Solução proposta

Bom, analisando o contexto acima, temos a clareza que nesta implementação iremos precisar trabalhar em diferentes frentes, sendo desempenho e segurança. Ao finalizar a análise deste problema conseguimos imaginar diversas formas de resolver os problemas descritos. Abaixo segue minha análise para resolução de cada um.

Antes de iniciarmos o destrinchamento da solução, estarei explicando um pouco sobre as tecnologias que serão utilizadas para termos clareza no momento da leitura.

Tecnologias utilizadas:

-Utilizaremos um banco de dados relacional sendo ele o MySql;

-Utilizaremos uma ferramenta de armazenamento/gerenciamento de cache que será o Redis;

-Utilizaremos um banco não relacional para armazenamento de log`s que será o mongoDB;

-A linguagem de programação utilizada será JAVA, utilizando dois framework`s diferentes, sendo o SpringBoot e o Quarks, no qual suas funções serão melhor explicadas no decorrer da apresentação.

-JWT (json web token) para autenticação dos usuário.

-Spring Security para facilitar a implementação.

-Os payloads de consumação desta API e de retorno serão sempre no formato JSON, facilitando o acesso e o tratamento destes dados, assim sendo possível serem consumidos por micro serviços e nanos serviços.

Para consumação das bases de dados A e B, iremos criar uma API RestFul utilizando o framework SpringBoot. Para segurança desta aplicação utilizaremos a forma de autenticação JWT, isso significa que para ser possível acessar os endpoints desta aplicação o usuário terá que ter um token, que somente será gerado a partir de um endpoint de login, quando informado um usuário e uma senha válida. Então terá esta aplicação terá a necessidade possuir um endpoint de cadastro de usuários. Além de cadastro do usuário, teremos a necessidade de um endpoint para cadastro de rotas, onde será informado quais endpoints o usuário tem permissão de acesso.

Para o endpoint de login, além de solicitar o “Login”, a “Senha” também iremos solicitar qual a rota(Endpoint) que este usuário quer acessar, pois para cada endpoint o token gerado será diferente, determinando seu tempo de duração.

Acesso a Base A: O token de acesso gerado para acessar a “**Base A”** terá validade de 30 segundos, isso fará com que seja mais custoso acessar este endpoint, visto que sua expiração é breve. Além de possuir o token válido, o usuário somente poderá acessar este endpoint caso o seu em seu perfil tenha esse acesso cadastrado, caso ao contrário o seu acesso será negado. Quando acessado este endpoint, será gravado um Log no banco de dados MongoDB, com as informações do usuário que efetuou a solicitação e também quais os dados foram solicitados (Exemplo número de conta,CPF ou um ID unico..), para fins de rastreabilidade, sendo possível identificar quem solicitou os dados e também teremos um registro inicial de quais foram dados foram solicitados.

Ideia extra para o acesso a Base A: Também existe a possibilidade de fazer este token ser único por requisição, porém para isso terá que ser feito um trabalho com a utilização de Cache. Exemplo: Ao gerar o token guardamos ele em Cache. Quando o usuário efetua o acesso ao endpoint, verificamos se ele ainda está guardando em cache e caso esteja, efetuamos a ação proposta pelo Endpoint do contrário não efetuamos e barramos o acesso. A exclusão deste registro do cachê sempre será efetuada ao finalizar as ações do endpoint.

Acesso a Base B: O token da Base B terá a validade 1 minuto, pois precisamos que o acesso a essa base seja mais rápido e precisamos de uma menor burocratização, além disto neste endpoint teremos o retorno da consulta gravado em cache para ser possível aproveitar este dados em novas requisições. Para o armazenamento em cache utilizaremos o redis, assim as próximas consultas serão mais rápidas pois, se o usuário estiver buscando os dados de uma pessoa que já foi procurada anteriormente, a aplicação não irá buscar no Mysql mas, sim do Redis, ganhando um tempo considerável nesta busca. Além disso, este endpoint também contará com o registro de log de acesso gravando os dados do usuário solicitante e também uma informação para rastreabilidade dos dados acessados, exemplo um número de conta ou um ID único.

Acesso a Base C: Referente a busca de dados da Base C, mesmo não possuindo dados críticos, ela apresenta dados pessoais, então iremos manter o token de validação porém, o tempo de validade do token será de 10 minutos. A gravação dos dados retornados da consulta em cache resolveria boa parte dos nossos problemas referente a velocidade porém, além disso poderíamos implementar este endpoint, utilizando o framework Quarks, que vem crescendo bastante no mercado, é estável, conseguimos aplicar o mesmo método de segurança e tem um desempenho muito maior que o SpringBoot. Neste endpoint eu não guardaria os dados do solicitante no Mongo para ganho de desempenho.

Linha de raciocínio para o desenvolvimento desta solução nesta primeira versão:

Nesta primeira versão de resolução deste desafio tratei como prioridade a segurança pensando em diversas formas de deixar o acesso a esse endpoint seguro e escalável. Em segundo momento pensei em como resolver o tempo de retorno das requisições feitas focando em descrever como as tecnologias seriam utilizadas. Não levei em consideração a aplicabilidade dos recursos como MySQL, MongoDB e Redis no Docker na aplicação desenvolvida neste momento, este será o próximo passo, no qual escreverei melhor no Readme.