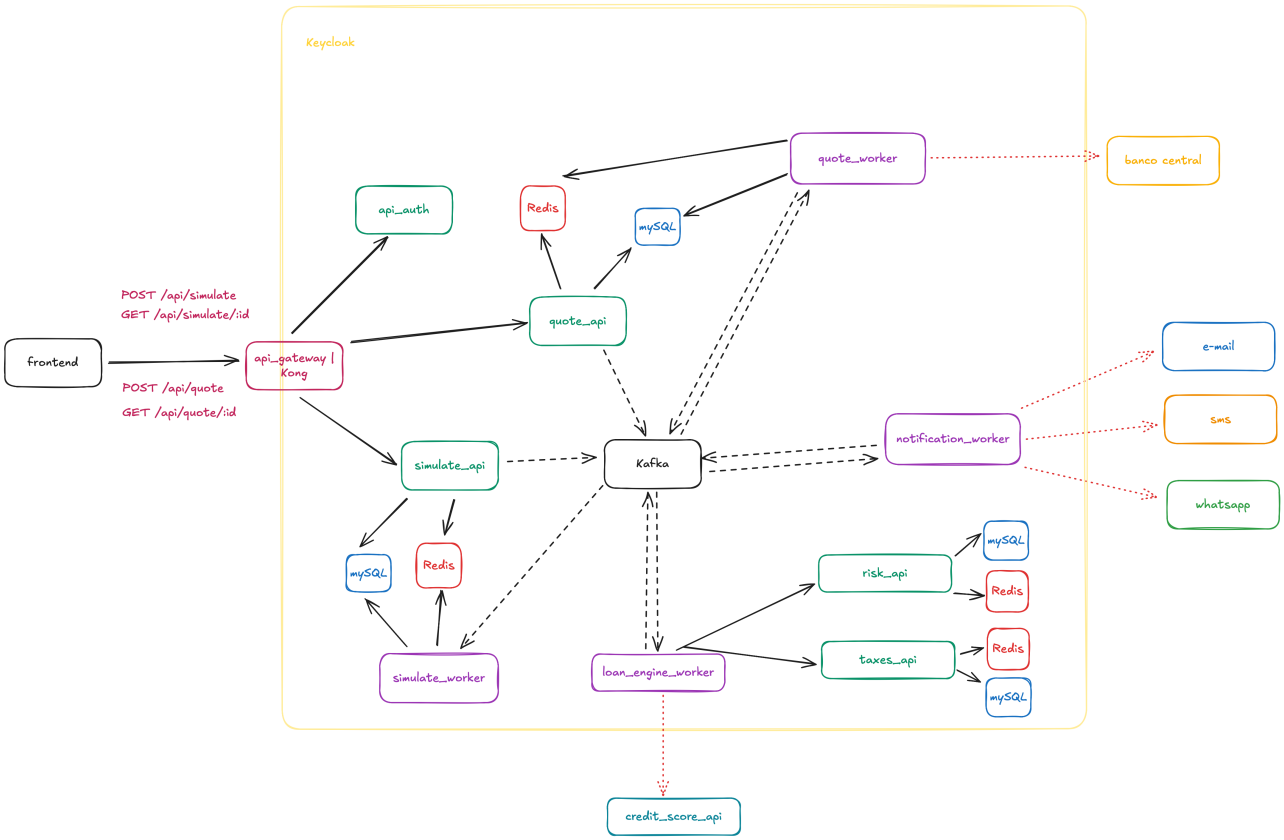
Arquitetura para um sistema de simulação e proposta de empréstimos



Link para o diagrama

Escolhas Tecnológicas

KEYCLOAK

Essa ferramenta é open-source e bastante robusta para o gerenciamento de identidade e acesso de usuário e serviços. Ela oferece algumas funcionalidades como Single Sign-On (SSO), autenticação multifator e suporte aos padrões OAuth2, OpenID etc.

Sendo assim, conseguimos criar um sistema de autenticação e autorização sem precisar ser do zero, com as devidas integrações e uma interface de gerenciamento amigável.

REDIS

O Redis é um banco de dados em memória bastante utilizado no mercado e não atua apenas como Engine de cache chave-valor. Podemos utiliza-lo para salvar informações críticas de forma temporária e também ter pub/sub para comunicação entre serviços, além da baixa latência.

KAFKA

O Kafka possui um design baseado em partições para processamento em tempo real de alto thoughtput, onde serviços podem publicar mensagens em tópicos e partições e outros serviços podem “assinar" este tópico para receber informações.

Ele também possui política de retenção de mensagens de acordo com políticas configuradas, ajudando a reter a mensagem em um tópico por mais tempo. E por conta da sua arquitetura, podemos transmitir milhões de mensagens em tempo real.

MYSQL

É um dos bancos de dados mais utilizados no mundo. Suportando transações ACID, replicação e um vasto ecossistema de ferramentas, garantindo segurança e integridade dos dados (ponto que é importante em um sistema financeiro).

SERVIÇOS

Para APIs e Workers a escolha seria Golang por ser uma linguagem simples e robusta, com recursos nativos de concorrência (goroutines e channels), permitindo a construção de serviços escaláveis de forma bastante fácil. Além disso, possui toda a suite de testes integrada na linguagem, favorecendo a construção de testes em código para garantir qualidade e segurança.

Outra escolha em serviços, seria a utilização do Docker para facilitar a implantação do serviço em qualquer nuvem, garantindo escala e tolerância a falhas de infraestrutura.

FRONTEND

No frontend, acredito que uma boa escolha seria o NextJS (React e Tailwind). O NextJS é um framework bastante consolidado e com diversas opções de integração. Aliado ao React e ao Tailwind, ele é uma opção robusta e que permite processamento client e server-side de forma fácil.

API-GATEWAY

O Api-Gateway é a porta de entrada da nossa aplicação, aqui podemos utilizar ferramentas que já possuem também um grande repositório para integrações, com boa documentação e que seja fácil de gerenciar. Pensando nisso, a escolha foi o Kong.
Nele, podemos integrar ferramentas via OAuth2, JWT etc, configurar rotas de fallback, cache, cors etc.

Padrões, Boas Práticas, Escala e Resiliência

MANUTENÇÃO

Relacionado aos padrões aplicados, iria por adicionar a parte de separação de domínio do **Domain Driven Design (DDD)** para facilitar o isolamento de regras complexas em entidades e serviços para ter coesão e separação de responsabilidade, facilitando o entendimento do negócio.

Adicionaria **injeção de dependência** para desaclopar os componentes dentro do código, permitindo que dependências sejam facilmente gerenciadas e substituídas caso seja necessário. Além de padrões como **Strategy** e **Factory** para trabalharmos com diversos tipos de simulações de empréstimos.

SEGURANÇA

Dois padrões bem interessantes e que podem nos ajudar com relação aos controles de acesso, auditoria, logging e monitoramento são: **Proxy** e **Decorator**.

O **Proxy** pode atuar como uma camada de controle de acesso, interceptando e validando credenciais, tokens etc, antes de encaminhar as requisições ao serviço destinatário. E o **Decorator**, permite realizar a parte de log e monitoria sem modificar a lógica dos objetos.

E para auxiliar toda a parte de roteamento e validações, a utilização de um **API-Gateway** é essencial, ainda mais atuando com o Keycloak. Aqui, podemos definir rate-limit, centralizar autorização e autenticação de forma integrada e desaclopar ainda mais nossa arquitetura.

DESEMPENHO E RESILIÊNCIA

A utilização do **Redis** facilita a aplicação de um padrão chamado **Cache-Aside,** onde fazemos operações de leitura/escrita em cache para então fazer no banco de dados subjacente, reduzindo o tempo de leitura e mantendo um desempenho satisfatório.

Com a utilização do **Kafka**, temos uma arquitetura **Event-Driven** que permite processamento assíncrono e comunicação desaclopada entre serviços. Essa abordagem permite escalar componentes específicos de acordo com a demanda, sem comprometer minha arquitetura como um todo.

E por fim, podemos adicionar **Circuit breaker** para que falhas em serviços possam comprometer outros, como num efeito dominó, cortando as comunicações entre serviços e permitindo a recuperação da arquitetura.

API Design

POST /api/simulate

Endpoint para simulação de empréstimos dado informações do solicitante, tais como: data de aniversário, renda, valor solicitado, número de documento e endereço.

GET /api/simulate/:id

Endpoint para obter o resultado de uma simulação dado o ID retornado pelo endpoint anterior.

POST /api/quote
Endpoint para efetivação de propostas de empréstimos.

GET /api/quote/:id
Endpoint para obter a proposta feita para o cliente.

Com relação a versionamento, acredito que o cliente dizer via header (accept-version ou custom header) qual versão ele está requerendo seja o ideal. Assim, conseguimos manter diversas versões de uma API e liberar contratos de forma específica sem alteração de URL (path ou parâmetros).

Motor de Simulação de Empréstimos

Nessa arquitetura, o motor de simulação de empréstimos é um worker que está recebendo e enviando atualizações para o kafka. Basicamente, ele é o ponto central da arquitetura onde são montados os cenários de simulação.

O ideal é que seja utilizado processamento paralelo para que sejam executados diversos cenários (what-if) e reduzir o tempo de resposta total. Além disso, podemos agregar outros serviços para que as análises sejam feitas de forma mais confiável possível: serviços de taxas, riscos e análise de crédito.

O motor de simulação pode se beneficiar de regras pré-definidas para análise e também para arredondamento de valor (para cima, para baixo, casas decimais etc).