Solution Review - Controle Financeiro

1. Visão Geral

Este documento descreve a arquitetura da solução para o desafio de controle financeiro, abordando os aspectos técnicos e estratégicos para garantir **escalabilidade, resiliência, segurança e desempenho**.

2. Arquitetura da Solução

A solução foi desenhada utilizando uma **arquitetura serverless** baseada em **AWS Lambda** e **API Gateway** para garantir escalabilidade automática e otimização de custos. Os principais componentes são:

2.1. Serviços Serverless

- **Lambda de Processamento de Transações (lambda-transaction)**
  - **Trigger:** SQS (sqs-transactions).
  - **Responsabilidade:**
    - Processar mensagens da fila, validar e armazenar no **DynamoDB** (Transactions table).
    - Enviar eventos para a fila **ConsolidationQueue** para consolidação diária.
    - Registrar logs no **Datadog**.
- **Lambda de Consolidação Diária (ConsolidationProcessor)**
  - **Trigger:** SQS (sqs-consolidation).
  - **Responsabilidade:**
    - Ler transações e calcular saldo diário.
    - Atualizar o **DynamoDB** (ConsolidatedReports table).
    - Monitorar processamento com **Datadog**.
- **Lambda de Reprocessamento das DLQ(s)**
  - **Responsabilidade:**
    - Reprocessar mensagens falhas e reenviar para as filas principal.
    - Se falhar novamente, gerar **alertas no Datadog e mantem elas na DLQ com trava pra não ser reprocessada evitando looping**.
    - Registrar logs detalhados para análise de erros.
- **Lambda de Segurança (SecurityLambda)**
  - **Trigger:** API Gateway / Lambda@Edge (CloudFront).
  - **Responsabilidade:**
    - Integrar com **PerimeterX** para mitigação de ataques de bots.
- **Função Lambda de Lançamentos (Transaction Lambda)**
  - Registra transações financeiras (débitos e créditos).
  - Independente da função de consolidação.
  - **Endpoints via API Gateway:**
    - POST /transactions → Criar lançamento.

        - GET /transactions/{date} → Buscar lançamentos do dia.
- **Função Lambda de Consolidação (Daily Report Lambda)**
    - Consolida e calcula saldo diário.
    - **Endpoints via API Gateway:**
        - GET /consolidated-report/{date} → Retorna saldo diário.

2.2. Fila de Mensagens (AWS SQS)

- **Monitoramento via Datadog:**
    - Configuração de alertas para mensagens acumuladas na fila.
    - Análise do tempo médio de processamento de mensagens.
    - Monitoramento de falhas e eventos na DLQ.
- **Fila principal (**sqs-transactions**)**: Processa eventos de transações.
- **Dead Letter Queue (**sqs-transactions-dlq**)**: Captura mensagens que falharam após 5 tentativas.
- **Monitoramento:**
    - CloudWatch Alarms para tempo de processamento e falhas recorrentes.

2.3. Base de Dados (AWS DynamoDB)

- **Tabela Transactions**:
    - PK: TransactionID (UUID).
    - SK: Date.
    - Atributos: Amount, Type, Category.
- **Tabela ConsolidatedReports**:
    - PK: Date.
    - Atributos: TotalDebits, TotalCredits, Balance.
- **Boas Práticas:**
    - **GSI(Global Secundar Index)** para consultas rápidas.
    - **Streams** acionando **AWS Lambda** para processamento automático.
    - **TTL** em registros antigos para otimizar armazenamento.

2.4. Autenticação e Segurança (AWS Cognito, AWS WAF e PerimeterX)

- **User Pool** para gerenciamento de usuários.
- **MFA (Multi-Factor Authentication)** ativado.
- **Integração com .NET** via JWT Bearer Authentication.
- **AWS WAF** para proteção contra ataques como SQL Injection e DDoS.
- **PerimeterX** para mitigação avançada de ameaças e detecção de bots maliciosos.

2.5. Monitoramento e Observabilidade (Datadog)

- **Logs:** Integração com Serilog para rastreamento detalhado de requisições e falhas.
- **Métricas:**
    - Número de transações por segundo.

    - Erros HTTP e tempo médio de resposta.
    - Consumo de filas do SQS e falhas na DLQ.
- **Dashboards:**
    - Visualização de tempo de resposta das funções Lambda.
    - Monitoramento de consumo e escalabilidade do DynamoDB.
    - Análise de eventos de autenticação no Cognito.
- **Alertas:**
    - Picos anormais de requisições.
    - Falhas de autenticação no Cognito.
    - Lentidão no DynamoDB ou indisponibilidade de serviços.

2.6. CDN e DNS (Route 53 e CloudFront)

- **AWS Route 53** para gerenciamento de DNS e roteamento inteligente de tráfego.
- **AWS CloudFront** com cache configurado para endpoints estáticos, garantindo baixa latência e redução de custos.

3. Infraestrutura e Deploy

- **Hospedagem**: AWS Lambda para Serverless.
- **Gerenciamento de Configuração**: AWS Parameter Store / Secrets Manager.
- **Pipeline CI/CD**: GitHub Actions para automação de testes e deploy.
- **Containerização**: Docker para testes locais.

4. Testes e Qualidade

- **Testes Unitários**: xUnit + Moq.
- **Testes de Integração**: Validação de endpoints e comunicação entre serviços.
- **Testes de Carga**: k6 para simular 50 RPS na API Gateway.
- **Análise de Código**: SonarQube para verificação de qualidade e detecção de vulnerabilidades.

5. Conclusão e Próximos Passos

Esta arquitetura proporciona **alta escalabilidade, segurança e resiliência**, garantindo um fluxo financeiro confiável e bem monitorado.

Evoluções Futuras

- Implementação de um **Event-Driven Architecture** com AWS EventBridge.
- Otimização do DynamoDB com **Adaptive Capacity**.
- Expansão do monitoramento com **tracing distribuído** via Datadog.
- Implementação de um mecanismo de **reprocessamento automático** para mensagens na **Dead Letter Queue (DLQ)** utilizando AWS Lambda.
- O Lambda consumirá mensagens da DLQ e tentará reenviá-las para a fila principal (sqs-transactions).
- Em caso de falha persistente, as mensagens serão logadas e monitoradas no Datadog para análise.
- Alternativa para reprocessamento manual via AWS CLI, caso necessário.
- Implementar uma arquitetura hexagonal , com componentes externos via nuget melhorando o acomplamento e implementação do mesmo ex: datadog , envio de email com uma abstratação maior.