Problemática: Um comerciante precisa controlar o seu fluxo de caixa diário com os lançamentos (débitos e créditos), também precisa de um relatório que disponibilize o saldo diário consolidado.

1. Mapeamento de Domínios Funcionais e Capacidades de Negócio

Domínios Funcionais

1. Controle de Lançamentos

- Cadastrar Lançamento: Adicionar débitos e créditos com data e descrição.
- Atualizar Lançamento: Editar detalhes de lançamentos existentes.
- Excluir Lançamento: Remover lançamentos.
- Listar Lançamentos: Consultar lançamentos por data ou categoria.

2. Consolidado Diário

- Calcular Saldo Diário: Gerar o saldo diário com base em lançamentos.
- Gerar Relatórios: Criar relatórios diários ou mensais com saldos consolidados.
- Visualizar Relatórios: Exibir relatórios para análise.

Capacidades de Negócio

- Entrada de Dados: Interface para registrar e editar lançamentos.
- Processamento de Dados: Calcula saldos e gera relatórios.
- Visualização: Exibição de relatórios e balanços diários.
- Segurança: Controle de acesso e proteção de dados.

2. Refinamento do Levantamento de Requisitos Funcionais e Não Funcionais

Requisitos Funcionais

1. Cadastro e Gerenciamento de Lançamentos

- API para criar, atualizar, e excluir lançamentos.
- Armazenamento de lançamentos com data, descrição, débito e crédito.

2. Cálculo e Relatórios

- Serviço que calcula o saldo diário consolidado.
- Geração de relatórios diários e mensais.

3. Visualização e Acesso

- Interface para visualizar relatórios e saldos.

- API para consultar saldos diários e relatórios.

Requisitos Não Funcionais

1. Escalabilidade: A solução deve suportar crescimento de dados e número de usuários.
2. Desempenho: Consultas e relatórios devem ser rápidos e eficientes.
3. Segurança: Proteção de dados e controle de acesso.
4. Disponibilidade: A solução deve estar disponível e resiliente a falhas.
5. Manutenibilidade: Código e arquitetura devem ser fáceis de manter e atualizar.

3. Desenho da Solução Completa (Arquitetura Alvo)

Arquitetura Proposta

1. Frontend

- AWS Amplify ou Amazon S3 + CloudFront: Para hospedar a aplicação frontend.
- Framework: React, Angular ou Vue.js.

2. Backend

- AWS Lambda: Micro serviços para processamento de lançamentos e cálculos.
  - `CreateTransactionFunction`: Adiciona e atualiza lançamentos.
  - `CalculateBalanceFunction`: Calcula e retorna saldo diário.
  - `GenerateReportFunction`: Gera e armazena relatórios.

- Amazon API Gateway: Para expor as funções Lambda como APIs RESTful.

3. Banco de Dados

- Amazon DynamoDB: Armazenamento de lançamentos e saldos.
- Amazon RDS (opcional): Para dados relacionais, se necessário.

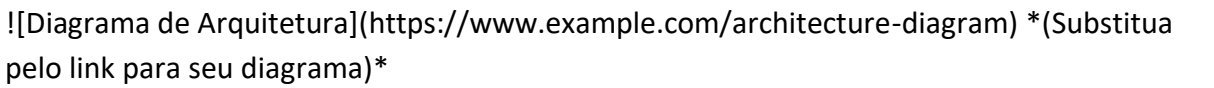
4. Relatórios

- Amazon QuickSight: Ferramenta de BI para criar e visualizar relatórios.
- AWS Lambda e Amazon S3: Para geração e armazenamento de relatórios em formatos como CSV ou PDF.

5. Notificações e Logs

- Amazon CloudWatch: Monitoramento e logs de serviços.
- Amazon SNS ou Amazon SQS: Para notificações ou mensagens entre serviços.

Diagrama de Arquitetura



4. Justificativa na Decisão/Escolha de Ferramentas/Tecnologias e Tipo de Arquitetura

Ferramentas e Tecnologias

- AWS Lambda: Permite execução de código sem necessidade de gerenciar servidores. Ideal para micro serviços com escalabilidade automática.
- Amazon DynamoDB: Banco de dados NoSQL altamente escalável e de baixa latência.
- Amazon API Gateway: Gerencia APIs RESTful, permitindo integração fácil com funções Lambda.
- Amazon QuickSight: Ferramenta de BI integrada à AWS para análises e relatórios.
- Amazon S3: Armazenamento de arquivos e relatórios.

Tipo de Arquitetura

- Arquitetura de Micro Serviços: Permite desacoplar funcionalidades, facilitando escalabilidade e manutenção.
- Serverless: Uso de AWS Lambda reduz a necessidade de gerenciamento de infraestrutura e suporta escalabilidade automática.

5. Testes

Testes Unitários

- Lambda Functions: Testar cada função Lambda individualmente para garantir que a lógica de negócios esteja correta.

Testes de Integração

- APIs: Testar a integração entre frontend, API Gateway, e funções Lambda.
- Banco de Dados: Verificar que os dados são corretamente armazenados e recuperados do DynamoDB.

Testes de Performance

- Carga e Estresse: Testar a aplicação com volume alto de dados e usuários para garantir desempenho adequado.