

# Sistema Fluxo de Caixa (SFC)

## **Desenho de Solução**

**Arquiteto: Jesse Leandro Leoni**

## Descritivo do Problema:

Um **comerciante** precisa controlar o seu **fluxo de caixa diário** com:

- os lançamentos de débitos e créditos;
- e um relatório que disponibilize o **saldo diário consolidado**;

Esta é a introdução e desenho inicial para entender o problema e como criar a solução (RASCUNHO):

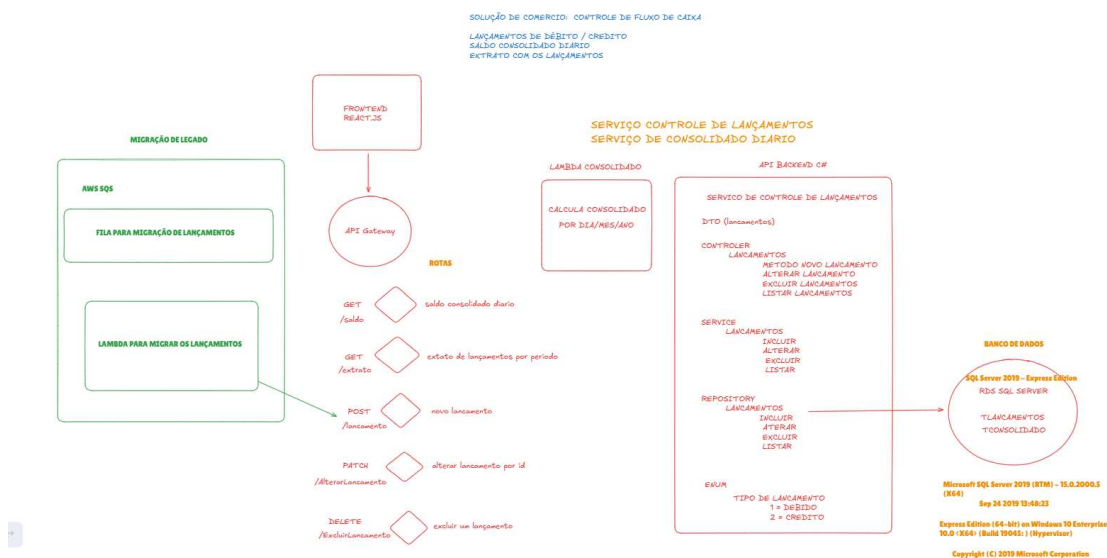
Ao ler o enunciado fui montando no **Exlidraw** o meu entendimento do projeto como um todo para depois poder desenhar a solução com mais detalhes.

Em verde logo no canto esquerdo pensei em como **migrar o legado** utilizando para isso uma fila **AWS SQS** que recebe os lançamentos do legado e uma função **AWS Lambda** lê a fila SQS e envia a requisição para o **AWS Gateway** chamando a rota **/Lancamento** passando o lançamento no **body** em formato **JSON**.

No meio em na cor vermelha, desenhei o **AWS Gateway** com as rotas de lançamento (CRUD) para os métodos, **GET / PUT / POST / DELETE**. Assim como, a rota **/Saldo** que irá buscar o Saldo Consolidado do Dia, passando a data no request.

Ainda em vermelho, temos o serviço de consolidação do saldo, que através de uma **lambda** acionada pelo **CRON** (agendada n vezes ao dia), faz o calculo do saldo e persiste no endpoint de **/Saldo** passando como parâmetros a data do dia e o valor do saldo calculado.

Temos o desenho inicial da **API** que fará o **CRUD** dos lançamentos e a persistência no banco de dados **SQL SERVER** (utilizei o SQL Express).

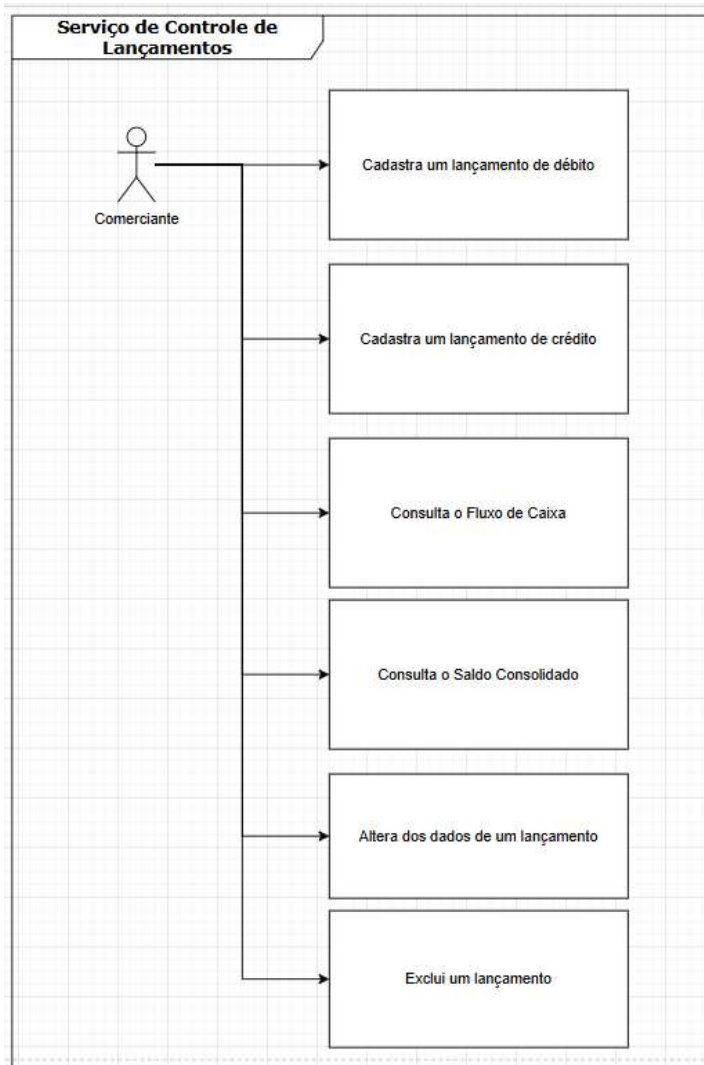


## Sumário

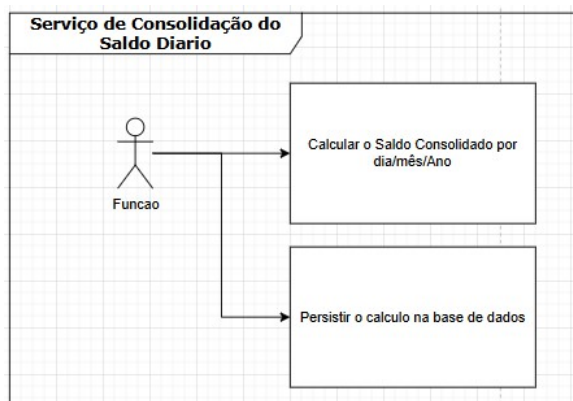
Requisitos de negócio: .....	4
Requisitos obrigatórios: .....	5
Mapeamento de domínios funcionais: .....	6
Capacidades de negócio: .....	7
Refinamento do Levantamento de requisitos funcionais e não funcionais: .....	8
Requisitos funcionais: .....	8
Requisitos não funcionais: .....	10
Desenho da solução completo (Arquitetura Alvo): .....	13
C4 – Model da Aplicação: .....	14
C4 – Model da Aplicação c/ a migração: .....	15
Justificativa na decisão/escolha de ferramentas/tecnologias e de tipo de arquitetura: .....	17
Infraestrutura: .....	17
Linguagens de Programação: .....	17
Ferramentas AWS: .....	18
Tipo de arquitetura da API: .....	20
Testes na API: .....	21
Testes no Visual Studio: .....	21
Readme .....	24
GIT .....	25
CUSTOS (AWS) .....	26
Monitoramento e Observabilidade: .....	27
Critérios de segurança para consumo (integração) de serviços: .....	28
DER – Diagrama de Entidade e Relacionamento .....	29
API – Serviço de Controle de Lançamentos .....	30
Chamando os métodos de Saldo Consolidado: .....	31
Chamando os métodos de lançamentos: .....	33
Agradecimentos .....	37

## Requisitos de negócio:

- Serviço que faça o **controle de lançamentos**:



- Serviço do **saldo consolidado diário**:



## Requisitos obrigatórios:

- Manter o cadastro de lançamentos de débito e crédito;
- Realizar o cálculo do saldo consolidado diário;

## Mapeamento de domínios funcionais:

### Modelo de Domínio: Fluxo de Caixa

#### Entidades principais:

##### 1. Lançamento

- `CodigoLancamento`: identificador único
- `DataLancamento`: data da transação
- `TipoLancamento`: entrada ou saída (crédito ou débito)
- `ValorLancamento`: valor da transação
- `DescricaoLancamento`: descrição da transação
- `CategoriaLancamento`: referência à categoria (sugestão futura - melhoria)

##### 2. Categoria

- `CategoriaLancamento`: identificador único
- `CategoriaNome`: nome da categoria (ex: "Salário", "Aluguel", "Venda", "Marketing")

##### 3. SaldoConsolidado

*(gerado a partir dos lançamentos)*

- `Data`: dia/mês/ano
- `Total_entradas`: soma dos lançamentos do tipo crédito
- `Total_saidas`: soma dos lançamentos do tipo débito
- `Saldo`: (entradas – saídas) + saldo do dia anterior

#### Relacionamentos

- Uma **Categoria** pode ser usada por muitos **Lançamentos**
- Um **SaldoConsolidado** é calculado a partir dos **Lançamentos** de uma determinada data escolhida.

#### Exemplo prático

Lançamento: {id: 1, data: 01/04/2025, tipo: "débito", valor: 3000, categoria: "Aluguel"}

Lançamento: {id: 2, data: 02/04/2025, tipo: "crédito", valor: 12000, categoria: "Salário"}

**OBS: Nesse teste removi a categoria e implementei o campo livre descrição.**

## Capacidades de negócio:

1. **Entradas de Caixa (Crédito):** Dinheiro recebido de vendas, investimentos, financiamentos, etc.
2. **Saídas de Caixa (Débito):** Pagamentos de despesas operacionais, investimentos, financiamentos, etc.
3. **Previsão de Fluxo de Caixa:** Estimativas de entradas e saídas futuras para ajudar na tomada de decisões. (No futuro seria legar implantar)
4. **Análise de Fluxo de Caixa:** Avaliação da saúde financeira da empresa com base nos fluxos de caixa. (No futuro seria legar implantar)
5. **Relatórios e Monitoramento:** Ferramentas para gerar relatórios e monitorar o desempenho financeiro ao longo do tempo. (No futuro seria legar implantar)
6. **Consultar o Consolidado de Saldo Diário:** Consultar o saldo consolidado diário.
7. **Calcular o Consolidado de Saldo Diário por Data:** Verificar o total de crédito de um dia, menos o total de débitos e adicionar o saldo do dia anterior.

Refinamento do Levantamento de requisitos funcionais e não funcionais:

## Requisitos funcionais:

**Eu como comerciante quero cadastrar um lançamento de crédito/débito no sistema;**

1. Entrar no tela inicial do sistema, e no menu Lançamentos, escolher a opção Novo Lançamento;
2. Abrir o formulário para a inserção de dados na tela;
3. Selecionar o Tipo de Lançamento ( 1 – Crédito / 2 – Débito);
4. Informar a Descrição do lançamento até 50 caracteres;
5. Informar o Valor do Lançamento até 99.999.999,99;
6. Escolher no calendário a Data do Lançamento;
7. O Sistema persiste o novo registro de lançamento e retorna a mensagem 200. (No futuro podemos melhorar, colocar 204 e a descrição "Sucesso ao incluir um novo Lançamento!". Assim como em caso de erro, podemos colocar um 400, dados informados com erro e colocar um fluent validation para dizer qual informação esta com problema. E se a API estiver fora do ar retornar um erro 500.)

### Exceções:

1. Tipo de lançamento diferente de 1 ou 2 deverá apresentar a exceção "Tipo de lançamento inexistente!";
2. Descrição do Lançamento deverá ter mais que 3 caracteres e menos que 51 caracteres, senão mostrar a exceção: "Descrição invalida!";
3. O Valor do lançamento deverá ser maior que zero e menor que 99.999.999,99. Caso contrário mostrar a mensagem de exceção: "O valor digitado está incorreto!";
4. A Data do Lançamento não poderá estar no futuro. Caso esteja, informar: "A data informada está incorreta!".

**Eu como comerciante quero verificar o saldo consolidado do dia no sistema;**

1. Entrar no tela inicial do sistema, e no menu Lançamentos, escolher a opção Saldo Consolidado;
2. Escolher no calendário a Data do Lançamento;
3. O Sistema exibe o Saldo Consolidado do Dia escolhido;

### Exceções:

A Data do Lançamento não poderá estar no futuro. Caso esteja, informar: "A data informada está incorreta!".



#### Eu como **comerciante** quero **excluir** um **lançamento** no sistema;

1. Entrar no tela inicial do sistema, e no menu Lançamentos, escolher a opção Listar Lançamentos;
2. Na grid de lançamentos na tela, escolha o lançamento a ser deletado, ou use a paginação para verificar mais lançamentos. Utilize os filtros do tipo de lançamento e data para facilitar a busca;
3. Selecionar o Lançamento;
4. Clique no botão “Excluir” na frente do lançamento desejado;
5. Confirme se deseja excluir o lançamento selecionado;
6. O Sistema irá excluir o lançamento desejado e vai retornar um 200 true no body. (No futuro podemos melhor a resposta e a exceção para a exclusão.)

#### Exceções:

1. 404 (not found), registro não encontrado.
2. 400 não foi possível excluir.

#### Eu como **comerciante** quero **alterar** um **lançamento** no sistema;

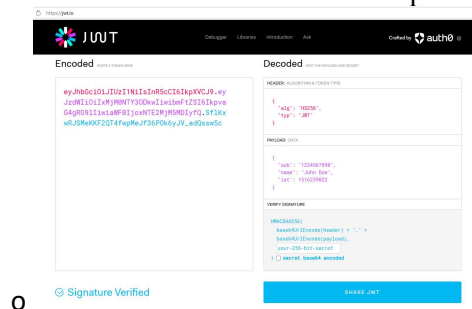
1. Entrar no tela inicial do sistema, e no menu Lançamentos, escolher a opção Listar Lançamentos;
2. Na grid de lançamentos na tela, escolha o lançamento a ser alterado, ou use a paginação para verificar mais lançamentos. Utilize os filtros do tipo de lançamento e data para facilitar a busca;
3. Selecionar o Lançamento;
4. Clique no botão “Alterar” na frente do lançamento desejado;
5. O sistema irá recuperar os dados do lançamento na tela;
6. Alterar as informações;
7. O Sistema irá persistir o lançamento desejado e vai retornar um 200 true no body. (No futuro podemos melhor a resposta e a exceção para a alteração.)

#### Exceções:

1. 404 (não encontrado), registro não encontrado.
2. 400 não foi possível alterar.

## Requisitos não funcionais:

- **Desempenho:** O sistema deve ser capaz de processar 50 transações por segundo.
- **Usabilidade:** A interface do usuário deve ser intuitiva e fácil de usar.
- **Escalabilidade:** O sistema deve suportar um aumento de 50% no número de usuários sem degradação de desempenho.
- **Manutenibilidade:** O sistema deve permitir atualizações e correções de bugs com o mínimo de interrupção.
- **Disponibilidade:** Colocamos o load balance no Cluster e também o auto scaling. Além disso, colocamos a aplicação em várias regiões, podendo colocar o CDN para melhorar o tempo de acesso a aplicação Web.
- **Segurança\*:** O sistema deve proteger dados sensíveis e prevenir acessos não autorizados. O serviço de controle de lançamento não deve ficar indisponível se o sistema de consolidado diário cair, para isso, as aplicações deverão estar desacopladas. Em dias de picos, o serviço de consolidado diário recebe 50 requisições por segundo, com no máximo 5% de perda de requisições, pensando nisso, que resolvemos adotar resiliência de filas **AWS SQS** para que os lançamentos não sejam perdidos se o banco de dados cair. O Controle de ataque por força bruta e **SQL Injection**, deverão estar habilitados no **WAF**, e as configurações de **permissões** no **IAM** deverão controlar os acessos entre no **API Gateway** e na **API**. Através da geração de **token JWT** iremos controlar a **autorização** nos métodos da API. Na aplicação WEB, implantar o **Captcha** e o controle de usuário e senha não permitindo o ataque por **User Enumeration**.



**Observar na API os seguintes itens:**

ID	Criticidade	Ponto de melhoria
01	Alto	Bypass no token de segurança
02	Alto	Acesso a dados sensíveis sem autenticação (API)
03	Médio	Ambiente staging acessível para internet
04	Baixo	Envio em massa do token de segurança

#### **01- Alto - Bypass no token de segurança**

Durante os testes, verificar o bypass nas operações/transações que exigem confirmação com a utilização do token de segurança sem necessariamente informar o token. Ao realizar um lançamento, o token informado foi 111111 que não é um token válido vindo do app, mas a transação foi realizada com sucesso devido a alteração na flag X-Channel da requisição. Na requisição original o valor da flag é X-Channel: 2 o que retorna token inválido, porém se o valor da flag for alterado para X-Channel: 3 ou X-Channel: 4 a transação ocorre normalmente e a aplicação ignora o valor informado do token.

Vale ressaltar que essa vulnerabilidade irá se repetir em qualquer operação/transação onde seja possível alterar o valor da flag X-Channel e onde o token seja enviado via flag X-Token juntamente com o parâmetro token no corpo da requisição.

#### **Recomendação:**

É preciso que a equipe de desenvolvimento valide o porquê essa alteração no valor da flag X-Channel está podendo causar essa anomalia e com isso permitir com que operações/transações sejam realizadas mesmo que o token seja inválido.

#### **02 – Alto - Acesso a dados sensíveis sem autenticação (API)**

É preciso implementar o mecanismo de autenticação e autorização na API.

#### **03 – Médio - Ambiente staging acessível para internet**

Durante a fase de testes, foi identificado que o ambiente staging da aplicação está acessível para a internet sem nenhum tipo de restrição de acesso. Os ambientes de teste devem ser bem protegidos e não acessíveis aos usuários porque podem ser vulneráveis a ataques de hackers e conter bugs que podem prejudicar o funcionamento do ambiente caso seja atacado. Se não forem devidamente protegidos, os ambientes de teste também podem se tornar pontos de entrada para acesso não autorizado a recursos e violações de dados. Se não forem isolados corretamente, os ambientes de teste inseguros podem até ameaçar o ambiente de produção.

#### **Recomendação:**

O ideal é que a aplicação tenha seu acesso restrito para apenas pessoas que estão envolvidos no desenvolvimento da aplicação. A restrição através de VPN é uma opção.

#### **04 - Baixo - Envio em massa do token de segurança**

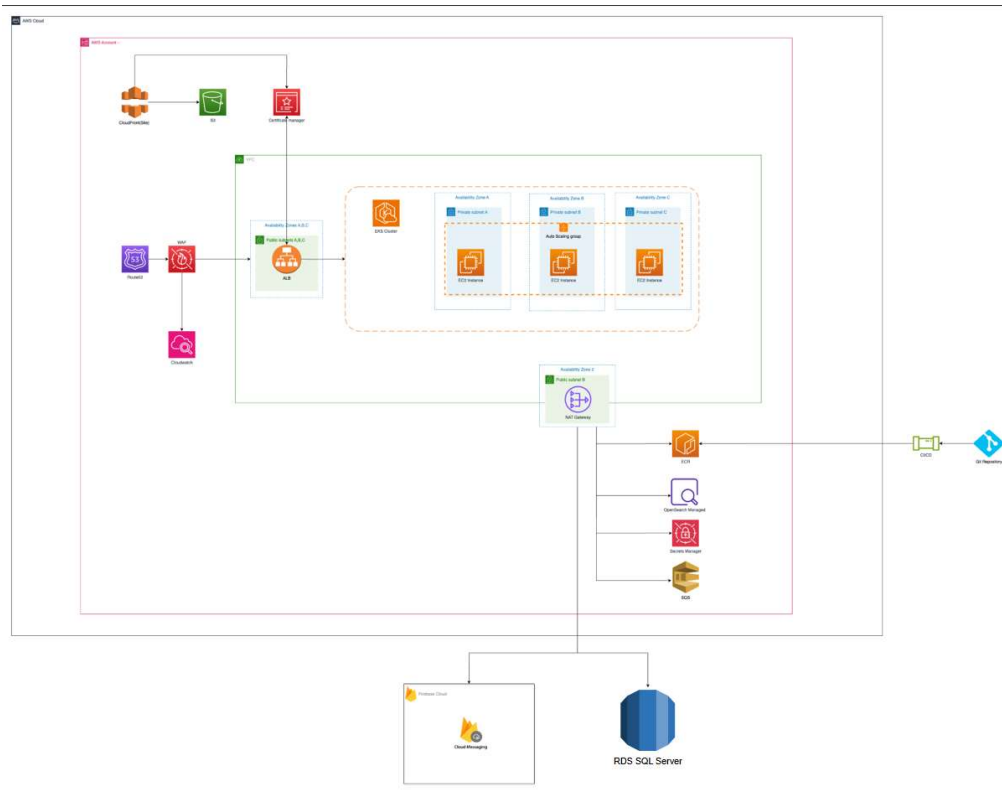
Durante a fase de testes, foi identificado que é possível gerar código de segurança que pode ser enviado para qualquer número independente se o CPF é válido, gerando com isso gastos com o envio de SMS. Basta o usuário informar o número do celular para o qual o código será enviado e automatizar a requisição. Com isso centenas de SMS poderão ser disparados.

#### **Recomendação:**

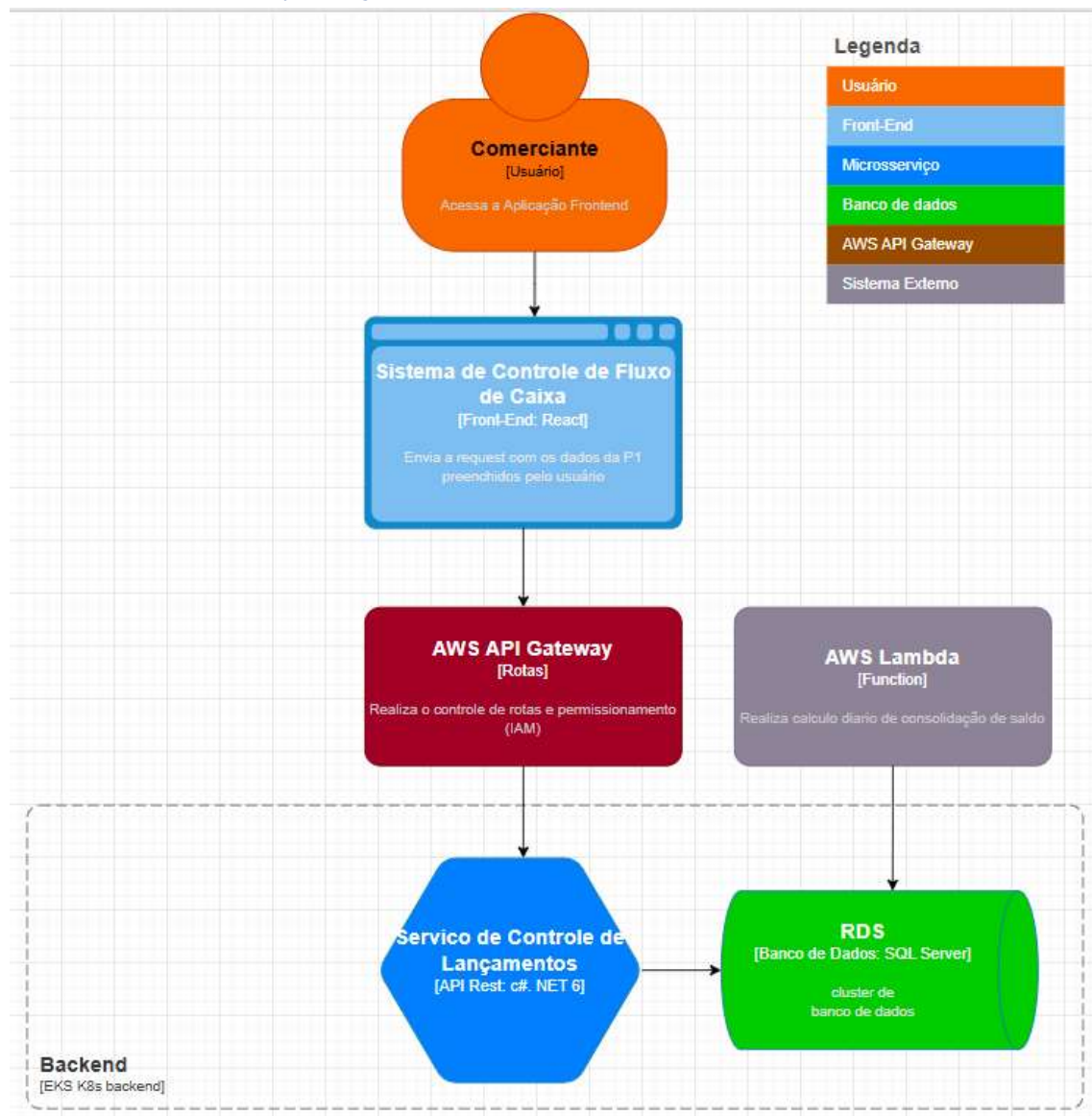
É preciso implementar o rate limit para um tempo maior no envio de SMS para evitar que um usuário malicioso consiga automatizar o envio desnecessário de tokens via SMS.

## Desenho da solução completo (Arquitetura Alvo):

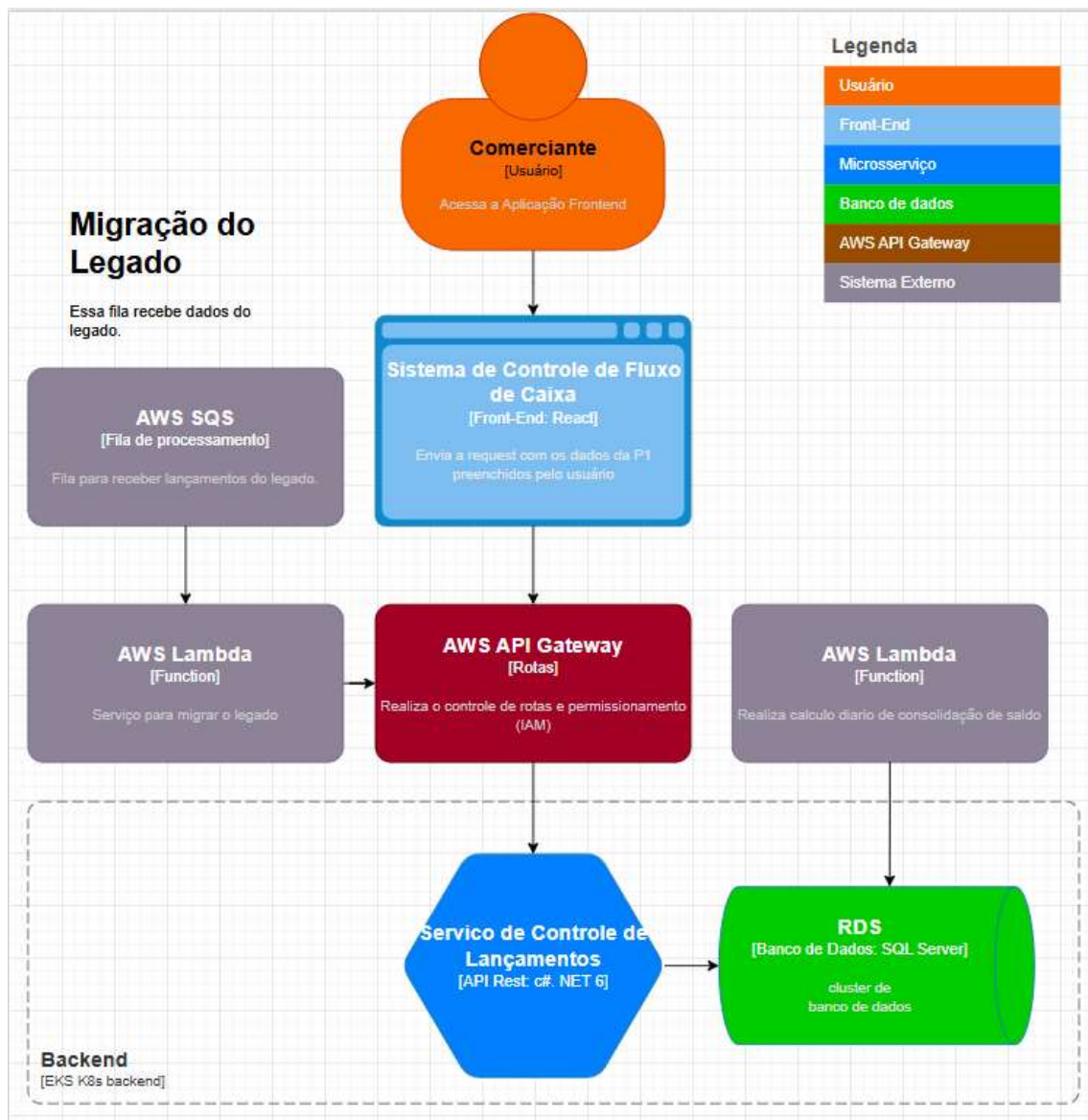
Esta documentação consiste em especificar recursos utilizados na **infraestrutura** do projeto de **Fluxo de Caixa**, listando-as em modo geral e como se integram. Também aborda brevemente a stack de tecnologias utilizadas no backend e frontend.



## C4 – Model da Aplicação:

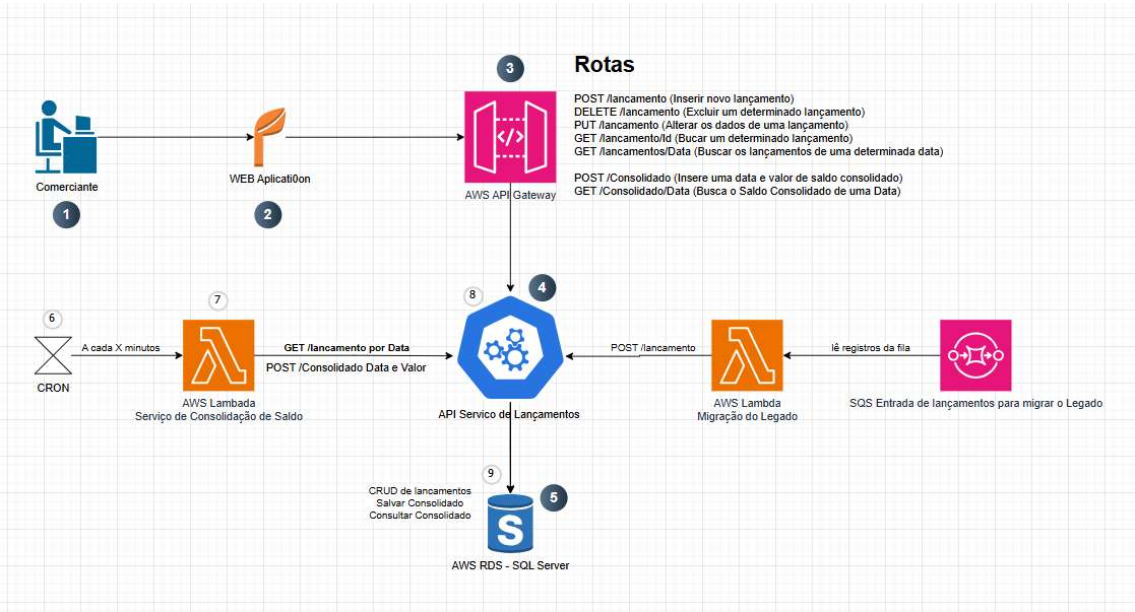


## C4 – Model da Aplicação c/ a migração:



**OBS:** Originalmente foi criado uma fila para a migração dos dados legados, mas com uma fila de **AWS SQS** podemos implementar resiliência na **API**, através do **Circuit Breaker Pattern** (Polly no C#), se o banco de dados estiver fora do ar (com toda a estrutura da nuvem, seria muito difícil ocorrer), temos opção de enviar os lançamentos para uma fila **SQS**, para não perder requisições. E podemos implantar uma fila de reproprocessamento e um **DLQ** para tratar do reenvio dos lançamentos e o caso de requisições perdidas. Podemos enviar um **SNS** para avisar sobre o período que o banco de dados esteve fora do ar, ou melhor, de todas as vezes que o circuito abriu e fechou.

Desenho de Arquitetura da Aplicação com a migração:





Justificativa na decisão/escolha de ferramentas/tecnologias e de tipo de arquitetura:

## Infraestrutura:

- 1x **WAF**(Web Application Firewall)
- 1x Cluster **EKS**
- 1x **Auto Scaling** Group
- 1x **EC2** t4g.large com volumes EBS gp2 20GB
- 1x Application **Load Balancer** em 3 Subnets publicas em A-Zs diferentes
- 1x **VPC**,
  - 3x Subnets públicas em A-Zs diferentes,
  - 3x Subnets privadas em A-Zs diferentes,
  - 1x NAT Gateway
- 1x Distribuições **Cloudfront**
- 1x **Bucket S3** privados
- 1x Repositório **ECR** privado para armazenar o código
- 1x Segredo no **AWS Secrets Manager**
- 1x Fila FIFO **SQS**
- 1x Registro **DNS** no **Route53**
- 1x Cluster OpenSearch gerenciado em duas A-Zs diferentes com 1 nó t3.medium.search, volumes EBS gp2 15GB e acesso público
- 1x Cluster **RDS SQL Server** com 3 nós dedicados da AWS e 10GB de disco
- 1x Firebase Cloud Messaging Especificações da stack de tecnologias Frontend (para mobile)

## Linguagens de Programação:

- ReactJS v20.0.0 e Typescript v4.9.5 Frontend WEB;
- C#.Net 9.0 na API e nas lambdas (Functions - Serveless) Backend;

# Ferramentas AWS:

## Calculadora de preços da AWS

▼

Mostrar cálculos

Conversões de unidades

Quantidade de armazenamento temporário alocada: 512 MB x 0.0009765625 GB em um MB = 0.5 GB

Calculos de definição de preço

5.000 solicitações x 30 ms x 0.001 fator de conversão de ms em segundos = 150,00 computação total (segundos)

0 GB/s - 400000 nível gratuito de GB/s = -400.000,00 GB/s

Max (-400000.00 GB/s, 0) = 0,00 total de GBs faturáveis

Tiered price for: 0,00 GB/s

Custo total do nível = 0,00 USD (cobranças mensais de computação)

Cobranças por computação mensal: 0,00 USD

5.000 solicitações - 1000000 solicitações de nível gratuito = -995.000 solicitações faturáveis mensais

Max (-995000 solicitações faturáveis mensais, 0) = 0,00 total de solicitações faturáveis mensais

Cobranças por solicitações mensais: 0 USD

0,50 GB - 0.5 GB (sem custo adicional) = 0,00 GB de armazenamento temporário faturável por função

Cobranças mensais de armazenamento temporário: 0 USD

Custos do Lambda - Com nível gratuito (mensal): 0.00 USD

## SQL Management Server para ambiente local:

SQL Server Enterprise - Localhost\SQLEXPRESS\SQLVCI (DEAD0D00209 BAC) - Microsoft SQL Server Management Studio

Server Explorer

Server Explorer Tree

Server Explorer Content

Query Editor - SQLVCI [DEAD0D00209 BAC]

Query Editor Content

Results

## Visual Studio 2022:

File Edit View Code Project Build Debug Test Analysis Tools Extensions Windows Help Search - SQLService

SQLService.WebApp

Code Editor

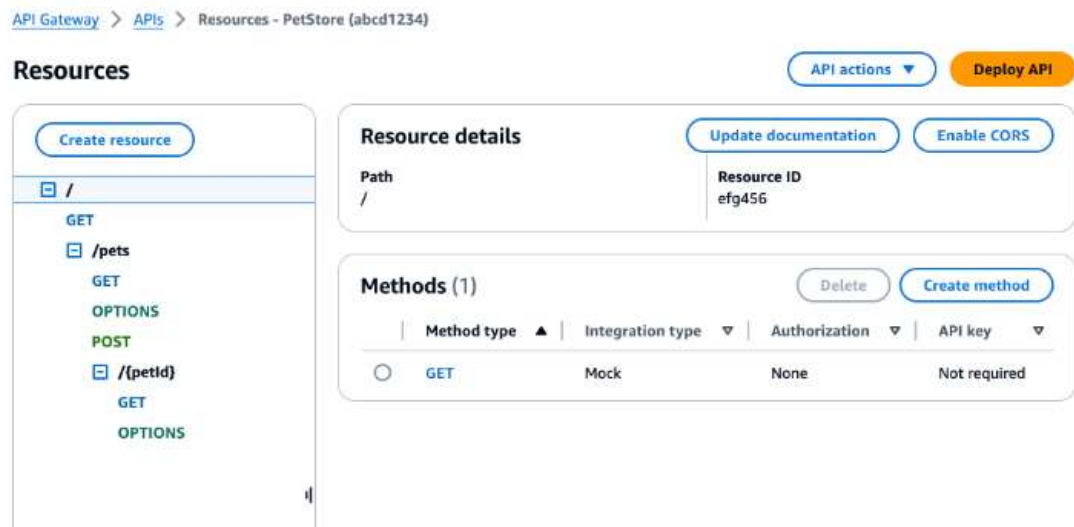
Solution Explorer

Output Window

## AWS API Gateway

Manual AWS passo a passo: [Tutorial: Criar uma API REST importando um exemplo - Amazon API Gateway](#)

Será necessário criar as rotas de lançamento e saldo e apontar para a API consumir, assim como as duas lambdas. Uma para migração do legado e outra para gerar o saldo consolidado.

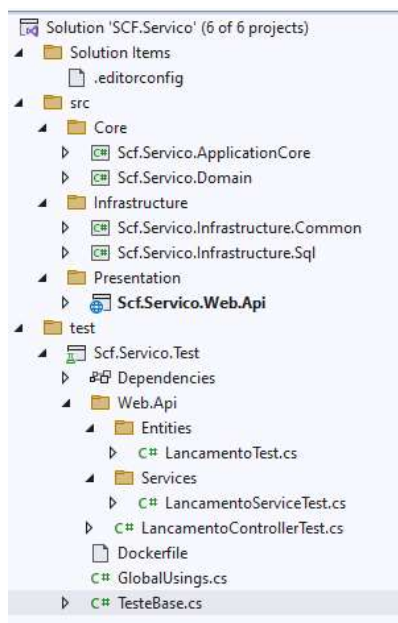


## Tipo de arquitetura da API:

Utilizei o **Domain-Driven Design (DDD)** é uma abordagem de desenvolvimento de software que se concentra em entender e modelar o domínio do problema em que o sistema de software opera. O DDD fornece uma estrutura para tomada de decisões, combinando práticas de design e desenvolvimento de software.

Utilizei os 5 princípios do **Solid** para o desenvolvimento da API:

- S – Single Responsibility Principle (Princípio da responsabilidade única)
- O – Open-Closed Principle (Princípio Aberto-Fechado)
- L – Liskov Substitution Principle (Princípio da substituição de Liskov)
- I – Interface Segregation Principle (Princípio da Segregação da Interface)
- D - Dependency inversion principle (Princípio da inversão de dependência)



A **Clean Architecture** representa uma abordagem estratégica no desenvolvimento de software que põe ênfase na manutenção da organização e estrutura do código de forma que permaneça resiliente a mudanças, seja em tecnologia, frameworks, ou requisitos de negócios.

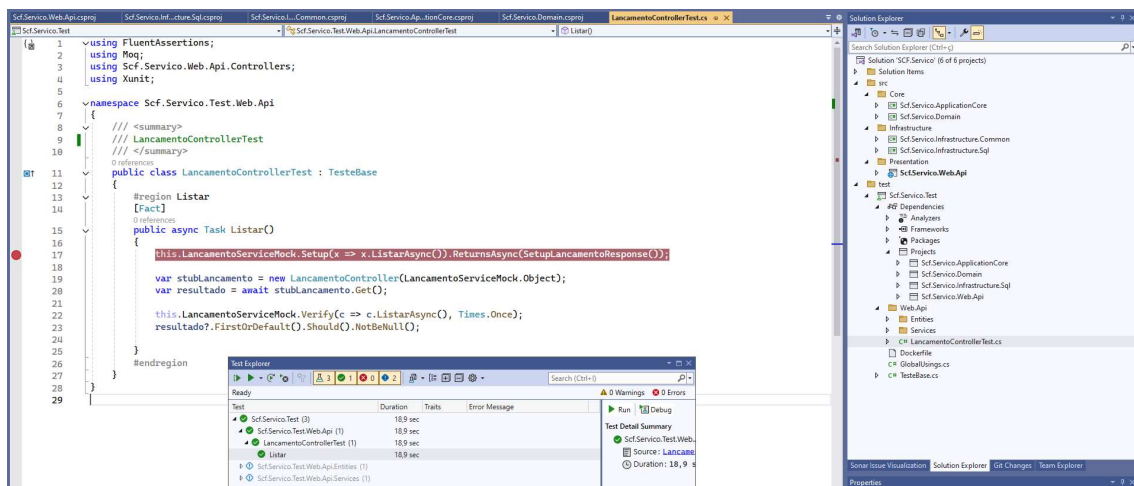
A comunicação será feita por **JSON**, que é uma formatação utilizada para estruturar dados em formato de texto e transmiti-los de um sistema para outro, como em aplicações cliente-servidor ou em aplicativos móveis. É um formato de arquivo de texto leve, compacto, no qual os dados são guardados no formato de par nome/valor, o qual também pode representar outras estruturas de dados, como arrays e objetos. JSON é um acrônimo de “Javascript Object Notation” ou simplesmente “Notação de objeto JavaScript”.

## Testes na API:

- Realizamos **testes unitários** na API com o **xUnit** e os **Mocks**;
- Realizamos testes na esteira de **CI/CD** com o **SonarQube** para avaliar no portão de qualidade possíveis problemas de **segurança**, **code smell** e **cobertura dos testes**;
- Testar na API o **health Check**:



## Testes no Visual Studio:



### Exemplo de testes unitários realizados:

```
using FluentAssertions;
using Moq;
using Scf.Service.Web.Api.Controllers;
using Xunit;

namespace Scf.Service.Test.Web.Api
{
    /// <summary>
    /// LancamentoControllerTest
    /// </summary>
    public class LancamentoControllerTest : TesteBase
    {
        #region Listar
        [Fact]
        public async Task Listar()
        {
            this.LancamentoServiceMock.Setup(x =>
            x.ListarAsync()).ReturnsAsync(SetupLancamentoResponse());
        }
    }
}
```

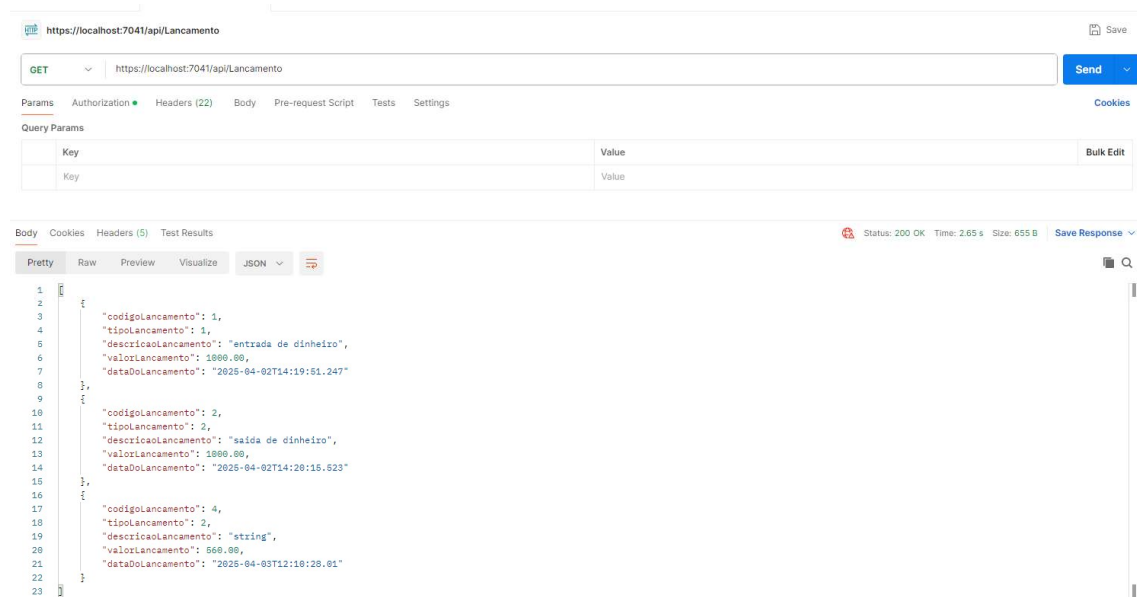
```

        var stubLancamento = new
LancamentoController(LancamentoServiceMock.Object);
        var resultado = await stubLancamento.Get();

        this.LancamentoServiceMock.Verify(c => c.ListarAsync(),
Times.Once);
        resultado?.FirstOrDefault().Should().BeNull();
    }
    #endregion
}
}

```

### Testes com o Postman para a API quando estiver rodando localmente:



```

curl --location 'https://localhost:7041/api/Lancamento' \
--header 'Accept: application/json' \
--header 'Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7' \
--header 'Access-Control-Allow-Origin: *' \
--header 'Authorization: Bearer ..j1X3OVab7kVXuUAHKgZaUQBED6FYTyE0Ry6Ftelpm--TEdZUfE6d4_t7PLRCCZekKw' \
--header 'Connection: keep-alive' \
--header 'Origin: http://localhost:3000' \
--header 'Referer: http://localhost:3000/' \
--header 'Sec-Fetch-Dest: empty' \
--header 'Sec-Fetch-Mode: cors' \
--header 'Sec-Fetch-Site: cross-site' \
--header 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36' \

```

```
--header 'sec-ch-ua: "Chromium";v="134", "Not:A-Brand";v="24", "Google Chrome";v="134"\' \
--header 'sec-ch-ua-mobile: ?0' \
--header 'sec-ch-ua-platform: "Windows"'
```

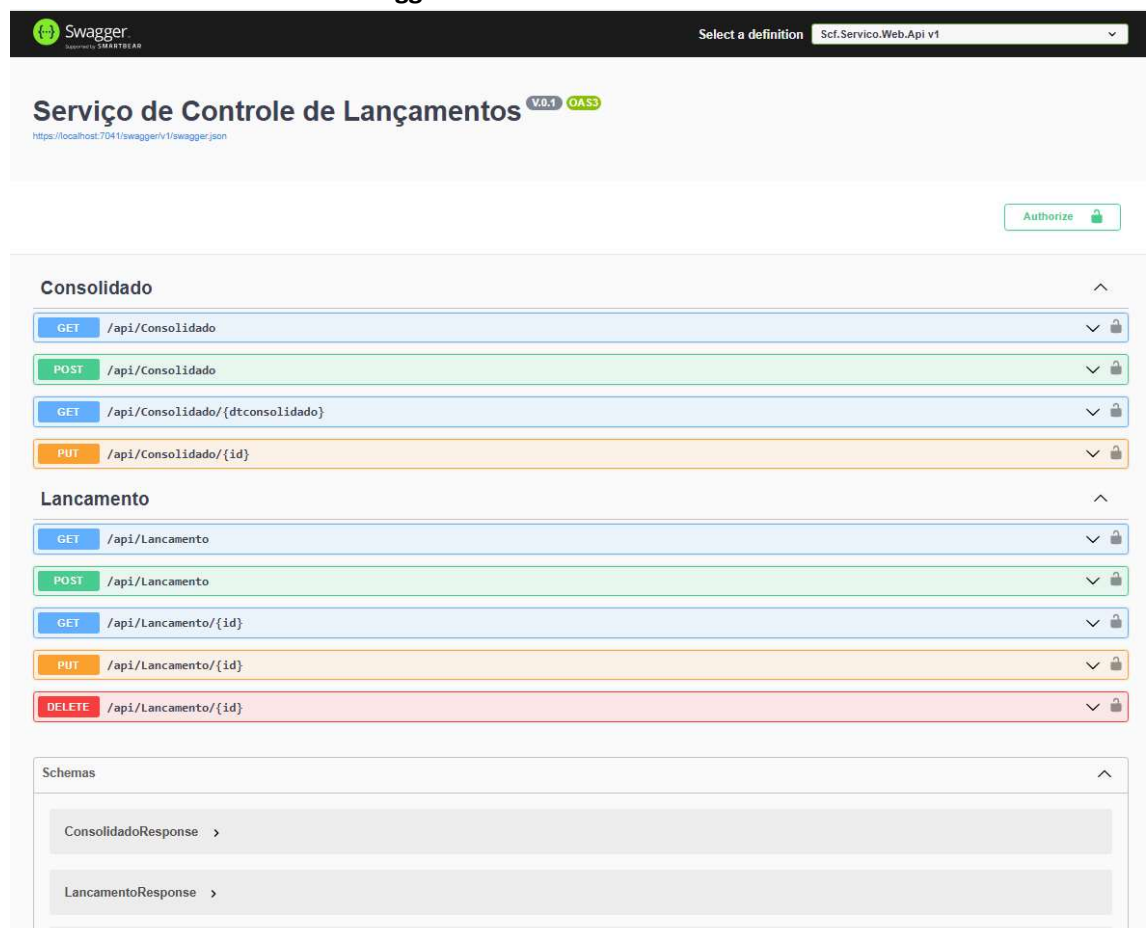
## Readme

Com instruções claras de como a aplicação funciona, e como rodar localmente:

1. Instalação do **SQL Server Express Edition**;
2. Criar o Banco de Dados **DBFLUXO1**;
3. Executar o **Script** para a criação dos objetos;
4. Instalar o **Visual Studio 2022**;
5. Abrir o projeto da **API**;
6. Configurar o arquivo **appsettings.json**:

```
"ConnectionStrings": {  
  // Defina a string de conexão  
  "Context":  
    "Server=localhost\\SQLEXPRESS;Database=DBFLUXO1;Trusted_Connection=Tr  
ue;"  
},
```

7. Executar o projeto da **API**;
8. Testes diretos na API com o **swagger**:



9. Configurar o **AWS API Gateway** com as rotas **/lançamento** e **/saldo**
10. Efetuar os testes no **AWS Gateway** e no **Postman**;



## GIT

Hospedar em repositório público (GitHub):

Acesse esse link para baixar os códigos:

<https://github.com/jesseopah/fluxoCaixa/tree/master>

# CUSTOS (AWS)

Estimativa de custos com infraestrutura e licenças para item mais sofisticados e com custo alto:

Adicionado com êxito Amazon OpenSearch Service estimar.

Resumo da estimativa

Informações

Custo inicial

0,00 USD

Custo mensal

27.947,89 USD

Custo total de 12 months

335.374,68 USD

Inclui um custo inicial

Conceitos básicos da AWS

Comece a usar gratuitamente

Entre em contato com a equipe de vendas

My Estimate

Duplicar

Excluir

Mover para

Criar grupo

Adicionar suporte

Adicionar serviço

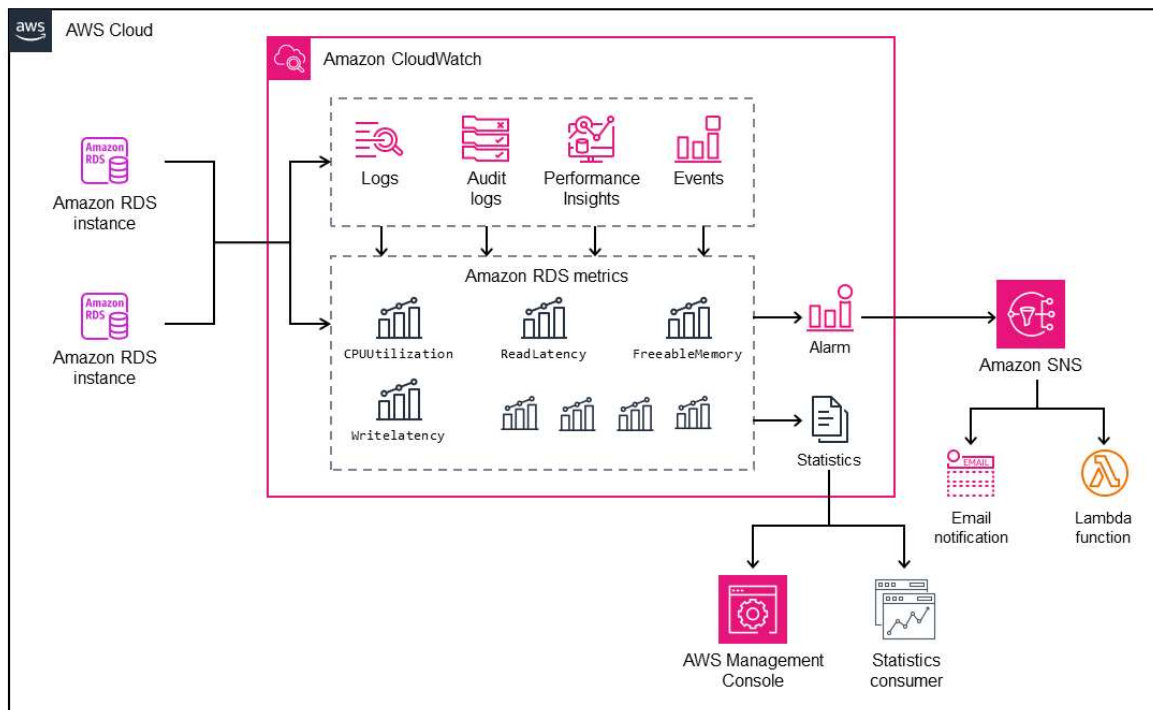
Localizar recursos

<input type="checkbox"/>	Nome do serviço	Status	Custo l...	Custo mensal	Descrição	Região	Resumo da configuração
<input type="checkbox"/>	AWS Web Application Firewall (WAF)	-	0,00 USD	1.756,00 USD	AWS Web Applicati...	América do Sul (Sã...	Número de listas de controle de acesso da web (ACLs da...
<input type="checkbox"/>	Amazon EKS	-	0,00 USD	73,00 USD	-	América do Sul (Sã...	Número de clusters do EKS (1), Número de nós híbridos ...
<input type="checkbox"/>	Amazon EC2	-	0,00 USD	199,88 USD	EC2 t4g.large com ...	América do Sul (Sã...	Locação (instâncias dedicadas), Sistema operacional (Lin...
<input type="checkbox"/>	Windows Server and SQL Server on Amazon EC2	-	0,00 USD	0,00 USD	Cluster RDS SQL S...	América do Sul (Sã...	EC2 Windows BYOL with SQL Server Standard BYOL
<input type="checkbox"/>	Amazon Route 53	-	0,00 USD	14.494,05 USD	Route53	América do Sul (Sã...	Zonas hospedadas (3), Registros adicionais em zonas ho...
<input type="checkbox"/>	Amazon CloudFront	-	0,00 USD	1,05 USD	Cloudfront	América do Sul (Sã...	Transferência de dados de saída para a Internet (10 GB ...
<input type="checkbox"/>	Amazon Simple Storage Service (S3)	-	0,00 USD	0,47 USD	-	América do Sul (Sã...	Armazenamento S3 Standard (10 GB por mês), Solicitaç...
<input type="checkbox"/>	AWS Secrets Manager	-	0,00 USD	0,40 USD	AWS Secrets Manager	América do Sul (Sã...	Número de segredos (1), Duração média de cada segred...
<input type="checkbox"/>	Amazon OpenSearch Service	-	0,00 USD	11.423,04 USD	Cluster OpenSearch	América do Sul (Sã...	Número de instâncias (1), Armazenamento para cada in...

Podemos diminuir itens a lista conforme o budget. Itens como o Opendsearch e configurações especiais do Route53 podem ser otimizados e o preço vai cair bastante.

## Monitoramento e Observabilidade:

O **Amazon CloudWatch** é um serviço que monitora aplicações, responde às mudanças de desempenho, otimiza o uso de recursos e fornece insights sobre a integridade operacional. Ao coletar dados de todos os recursos da AWS, o CloudWatch fornece visibilidade sobre o desempenho de todo o sistema e permite que os usuários definam alarmes, reajam automaticamente às mudanças e obtenham uma visão unificada da integridade operacional. E o **Grafana** para a observabilidade é uma excelente ferramenta.



## Critérios de segurança para consumo (integração) de serviços:

A Amazon Web Services (AWS) oferece vários critérios de segurança para a integração de serviços, incluindo padrões de conformidade, serviços de segurança e ferramentas de proteção.

### **Padrões de conformidade**

- A AWS oferece suporte a padrões de segurança e certificações de conformidade como PCI-DSS, HIPAA/HITECH, FedRAMP, GDPR, FIPS 140-2 e NIST 800-171
- A AWS garante que os padrões de conformidade sejam suficientes para atender aos requisitos de medidas de conformidade mais comuns

### **Serviços de segurança**

- O AWS Security Hub centraliza as verificações de segurança de outros serviços AWS, incluindo **AWS Config** regras
- A Proteção de redes e aplicações na AWS fornece controle em linha do tráfego para ajudar a proteger contra acesso não autorizado

### **Ferramentas de proteção**

- A autenticação multifatorial (**MFA**) pode ser ativada no usuário raiz da conta da AWS e nos usuários com acesso interativo ao AWS Identity and Access Management (**IAM**)

### **Responsabilidade compartilhada**

- A conformidade é uma responsabilidade compartilhada entre o cliente AWS e a AWS
- Os clientes podem usar a documentação sobre controle e conformidade da AWS para executar seus procedimentos de avaliação e verificação de controle

## DER – Diagrama de Entidade e Relacionamento

Construção de duas tabelas no Banco de Dados: **SQL Server Express**

**//Script de criação do Banco de Dados SQL Server**

```
USE [DBFLUX01]
GO
```

```
CREATE TABLE dbo.TLancamentos
(
Codigo int identity(1,1) primary key,
TipoLancamento int not null,
DescricaoLancamento varchar(50) null,
ValorLancamento decimal(10,2) not null,
DataDoLancamento datetime not null,
)
GO
```

```
CREATE TABLE dbo.TConsolidados
(
Codigo int identity(1,1) primary key,
DataConsolidado datetime not null,
TotalEntrada decimal(10,2) not null,
TotalSaida decimal(10,2) not null,
ValorConsolidado decimal(10,2) not null
)
GO
```

```
INSERT INTO [dbo].[TLancamentos]
([TipoLancamento]
,[DescricaoLancamento]
,[ValorLancamento]
,[DataDoLancamento])
VALUES
(1,'entrada de dinheiro',1000.00,getdate());
GO
```

```
INSERT INTO [dbo].[TLancamentos]
([TipoLancamento]
,[DescricaoLancamento]
,[ValorLancamento]
,[DataDoLancamento])
VALUES
(2,'saida de dinheiro',1000.00,getdate());
GO
```

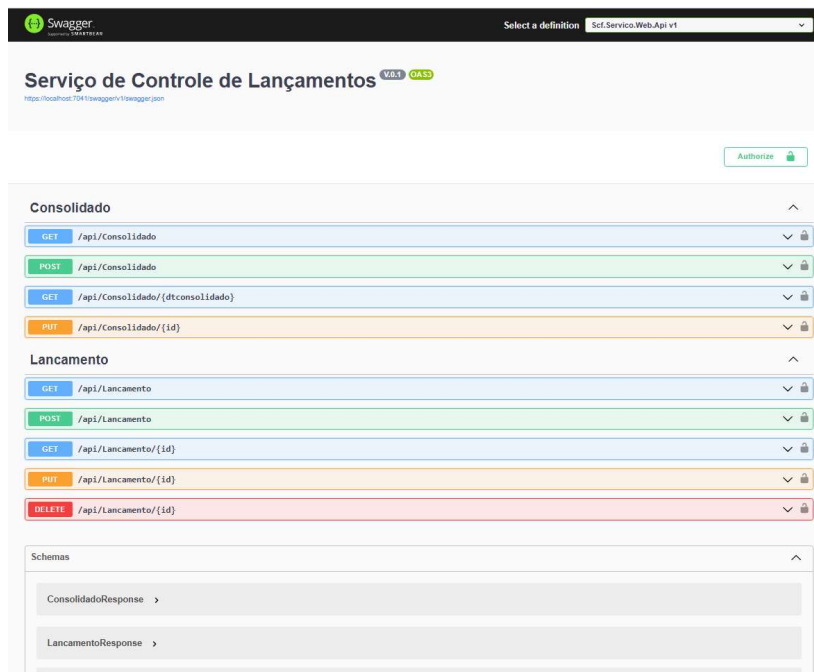
```
INSERT INTO dbo.TConsolidados
values ('02/04/2025',1000,1000,0);
```

## API – Serviço de Controle de Lançamentos

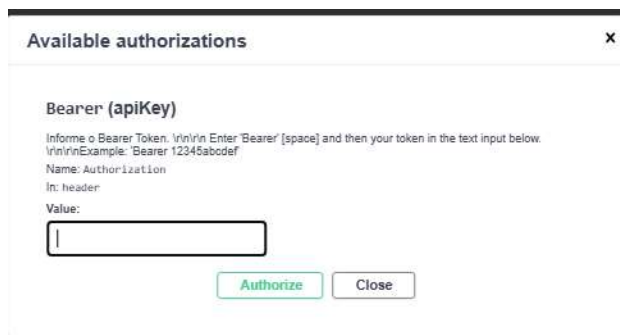
A API **Serviço de Controle de Lançamentos** será necessária para o CRUD de lançamentos de Débito e Crédito. Assim como, a lista de lançamentos (Extrato) e o Saldo Consolidado.

Executando a API localmente (Swagger): [Swagger UI](#)

Resultado esperado:

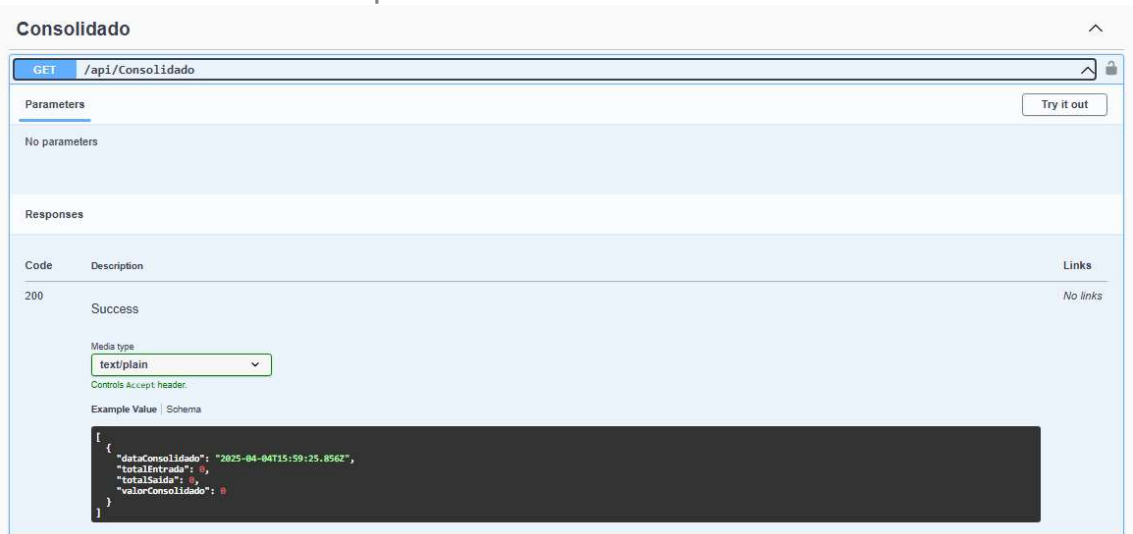


Incluir o Token JWT para a autorização conforme exemplo do Swagger:



# Chamando os métodos de Saldo Consolidado:

## 1. Listar Saldos Consolidados por data:



**Consolidado**

GET /api/Consolidado

Parameters

No parameters

Responses

Code	Description	Links
200	Success	No links

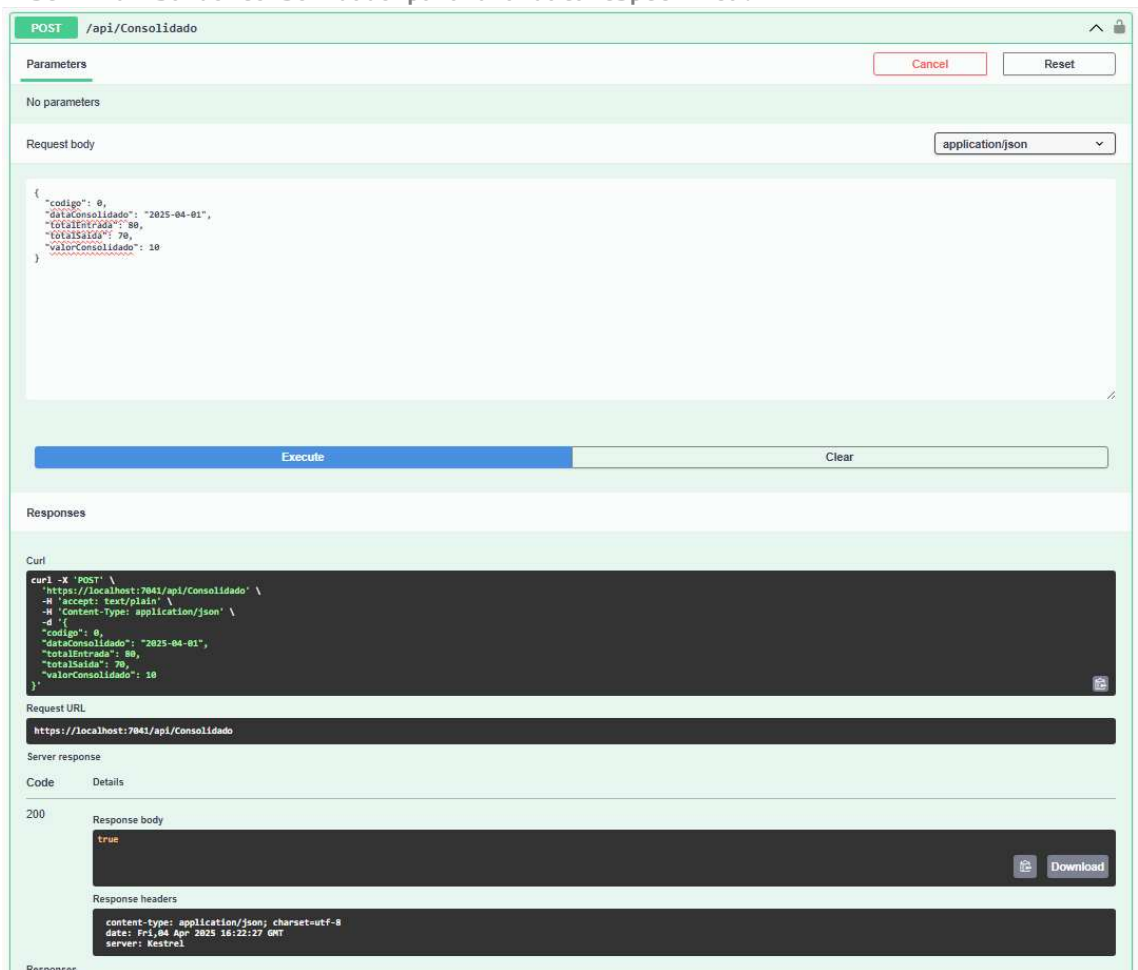
Media type: text/plain

Controls Accept header.

Example Value | Schema

```
{
  "dataConsolidado": "2025-04-04T15:59:25.856Z",
  "totalEntrada": 0,
  "totalSaida": 0,
  "valorConsolidado": 0
}
```

## 2. Inserir um Saldo Consolidado para uma data específica:



**POST** /api/Consolidado

Parameters

No parameters

Request body

application/json

```
{
  "codigo": 0,
  "dataConsolidado": "2025-04-01",
  "totalEntrada": 80,
  "totalSaida": 70,
  "valorConsolidado": 10
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'https://localhost:7041/api/Consolidado' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "codigo": 0,
    "dataConsolidado": "2025-04-01",
    "totalEntrada": 80,
    "totalSaida": 70,
    "valorConsolidado": 10
  }'
```

Request URL

https://localhost:7041/api/Consolidado

Server response

Code	Details
200	<p>Response body</p> <p>true</p> <p>Response headers</p> <p>content-type: application/json; charset=utf-8 date: Fri, 04 Apr 2025 16:22:27 GMT server: Kestrel</p>

Responses

### 3. Buscar Saldo Consolidado por Data: (2025-04-01) AAAA-MM-DD

GET /api/Consolidado/{dtconsolidado}

Parameters

Name	Description
dtconsolidado * required	
string (path)	2025-04-01

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7041/api/Consolidado/2025-04-01' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7041/api/Consolidado/2025-04-01

Server response

Code	Details
200	<p>Response body</p> <pre>{   "codigo": 1,   "dataConsolidado": "2025-04-01T00:00:00",   "totalEntrada": 10,   "totalSaida": 20,   "valorConsolidado": 0 }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Fri, 04 Apr 2025 15:25:04 GMT server: Kestrel</pre>

Download

### 4. Alterar um Saldo Consolidado por ID:

PUT /api/Consolidado/{id}

Parameters

Name	Description
id * required	
integer(\$int32) (path)	1

Request body

application/json

```
{
  "codigo": 1,
  "dataConsolidado": "2025-04-02",
  "totalEntrada": 30,
  "totalSaida": 20,
  "valorConsolidado": 10
}
```

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'https://localhost:7041/api/Consolidado/1' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "codigo": 1,
    "dataConsolidado": "2025-04-02",
    "totalEntrada": 30,
    "totalSaida": 20,
    "valorConsolidado": 10
  }'
```

Request URL

https://localhost:7041/api/Consolidado/1

Server response

Code	Details
200	<p>Response body</p> <pre>true</pre>

Download



# Chamando os métodos de lançamentos:

## 1. Listar lançamentos:

**Lancamento**

GET /api/Lancamento

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X GET: \
  "https://localhost:7041/api/Lancamento" \
  -H "accept: text/plain"
```

Request URL

https://localhost:7041/api/Lancamento

Server response

Code Details

200

Response body

```
{
  "codigo": 1,
  "tipoLancamento": 1,
  "descricaoLancamento": "entrada de dinheiro",
  "valorLancamento": 1000,
  "dataLancamento": "2025-04-04T12:12:02.933"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 04 Apr 2025 16:41:13 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	Success	No links

## 2. Buscar lançamento por ID:

GET /api/Lancamento/{id}

Parameters

Name Description

id \* required  
integer(\$int32)  
(path)

1

Execute Clear

Responses

Curl

```
curl -X GET: \
  "https://localhost:7041/api/Lancamento/1" \
  -H "accept: text/plain"
```

Request URL

https://localhost:7041/api/Lancamento/1

Server response

Code Details

200

Response body

```
{
  "codigo": 1,
  "tipoLancamento": 1,
  "descricaoLancamento": "entrada de dinheiro",
  "valorLancamento": 1000,
  "dataLancamento": "2025-04-04T12:12:02.933"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 04 Apr 2025 16:42:41 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	Success	No links

3. Alterar lançamento por ID:

PUT /api/Lancamento/{id}

Parameters

Cancel Reset

Name	Description
id * required Integer(\$int32) (path)	2

Request body

application/json

```
{  "codigo": 2,  "tipoLancamento": 2,  "descricaoLancamento": "SAIDA DE DINHEIRO",  "valorLancamento": 30,  "dataDoLancamento": "2025-04-04T16:43:54.147Z"}
```

Execute Clear

Responses

Curl

```
curl -X 'PUT' \  "https://localhost:7041/api/Lancamento/2" \  -H 'accept: text/plain' \  -H 'Content-Type: application/json' \  -d '{  "codigo": 2,  "tipoLancamento": 2,  "descricaoLancamento": "SAIDA DE DINHEIRO",  "valorLancamento": 30,  "dataDoLancamento": "2025-04-04T16:43:54.147Z"  }'
```

Request URL

https://localhost:7041/api/Lancamento/2

Server response

Code Details

200

Response body

true

Download

Resultado esperado:

Results		Messages			
	Codigo	TipoLancamento	DescricaoLancamento	ValorLancamento	DataDoLancamento
1	1	1	entrada de dinheiro	1000.00	2025-04-04 12:12:02.933
2	2	2	SAIDA DE DINHEIRO	30.00	2025-04-04 16:43:54.147

4. Excluir um lançamento pelo ID:

DELETE

/api/Lancamento/{id}

Parameters

Cancel

Name	Description
id * required	
Integer(\$int32)	
(path)	

4

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
'https://localhost:7041/api/Lancamento/4' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:7041/api/Lancamento/4
```

Server response

Code	Details
200	<div><div>Response body</div><div>true</div><div>Download</div></div> <div><div>Response headers</div><div>content-type: application/json; charset=utf-8 date: Fri, 04 Apr 2025 16:47:58 GMT server: Kestrel</div></div>

Responses

Code	Description	Links
200	Success	No links

Media type

text/plain

Controls Accept header

Example Value | Schema

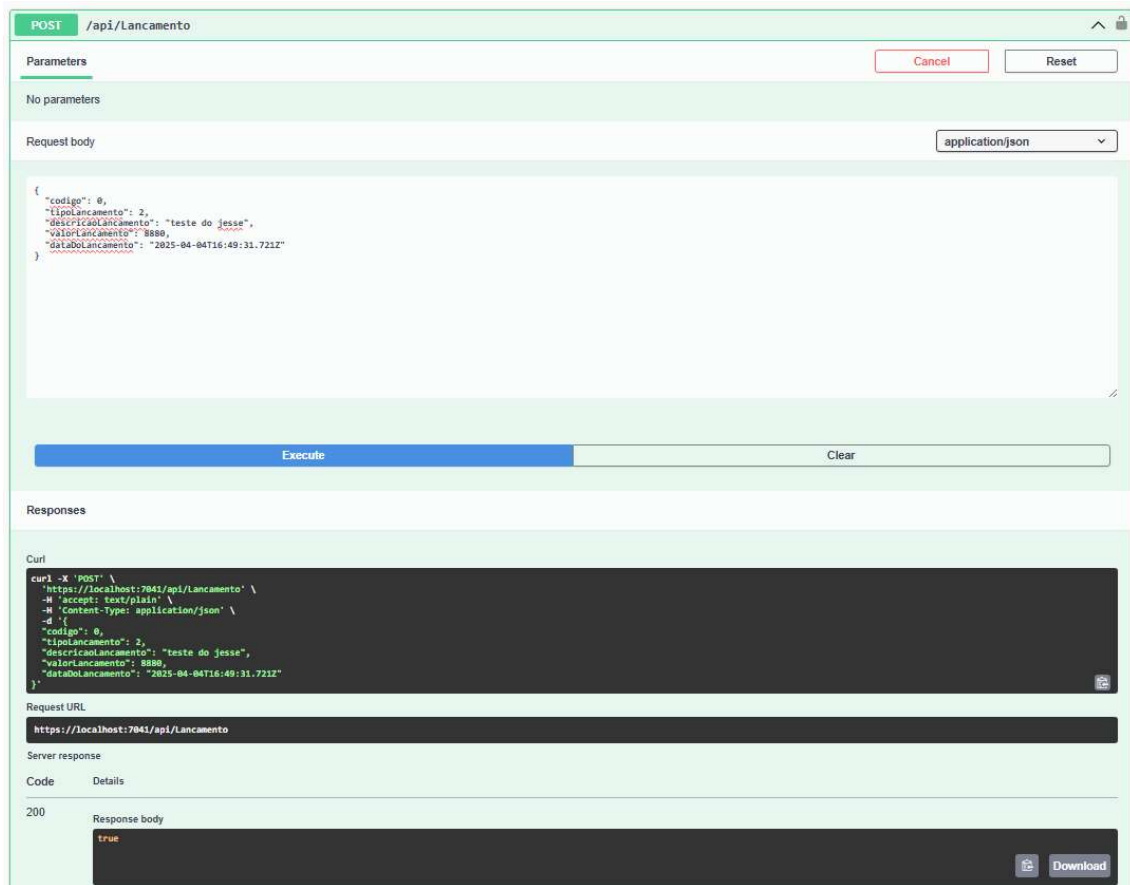
true

Resultado esperado:

O registro com ID = 3 foi removido:

	CodigoLancamento	TipoLancamento	DescricaoLancamento	ValorLancamento	DataDoLancamento
1	1	1	entrada de dinheiro	1000.00	2025-04-02 14:19:51.247
2	2	2	saida de dinheiro	1000.00	2025-04-02 14:20:15.523

5. Incluir um novo lançamento (Tipo de lançamento igual a 1 = Crédito / 2 = Débito):



Resultado desejado:

100 %					
Results		Messages			
	Codigo	TipoLancamento	DescricaoLancamento	ValorLancamento	DataDoLancamento
1	1	1	entrada de dinheiro	1000.00	2025-04-04 12:12:02.933
2	2	2	SAIDA DE DINHEIRO	30.00	2025-04-04 16:43:54.147
3	5	2	teste do jesse	8880.00	2025-04-04 16:49:31.720

## Agradecimentos

**Obrigado!**

**Jesse Leandro Leoni**

**Arquiteto de Solução**

[Jesse.leoni@opah.com.br](mailto:Jesse.leoni@opah.com.br)