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):**

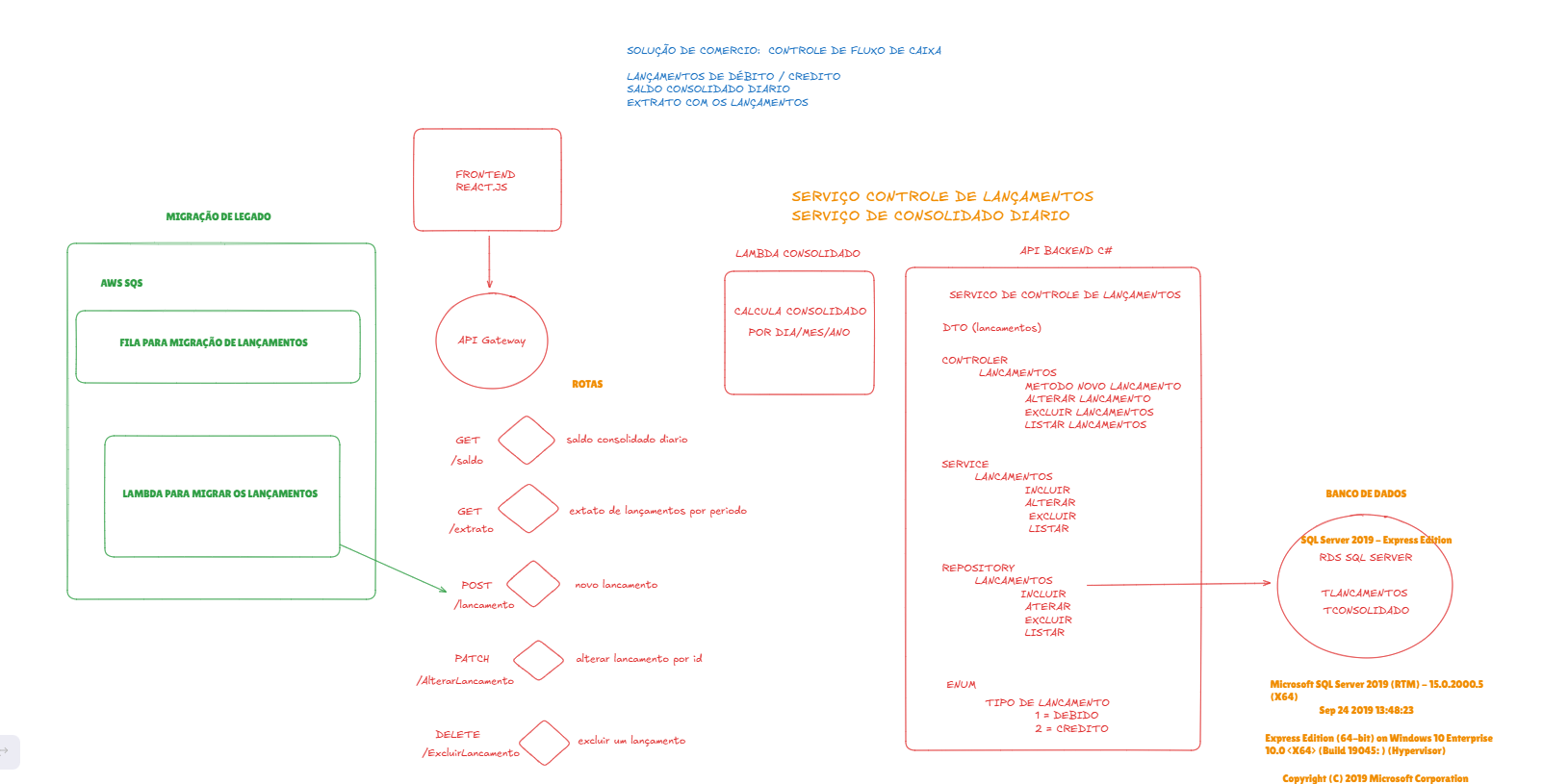
Ao ler o enunciado fui montando no **Exlidraw** o meu entendimento do projeto como um todo para depois poder desenhar a solução com mais detalhes.

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](#_Toc194667488)

[Requisitos obrigatórios: 5](#_Toc194667489)

[Mapeamento de domínios funcionais: 6](#_Toc194667490)

[Capacidades de negócio: 7](#_Toc194667491)

[Refinamento do Levantamento de requisitos funcionais e não funcionais: 8](#_Toc194667492)

[Requisitos funcionais: 8](#_Toc194667493)

[Requisitos não funcionais: 10](#_Toc194667494)

[Desenho da solução completo (Arquitetura Alvo): 13](#_Toc194667495)

[C4 – Model da Aplicação: 14](#_Toc194667496)

[C4 – Model da Aplicação c/ a migração: 15](#_Toc194667497)

[Justificativa na decisão/escolha de ferramentas/tecnologias e de tipo de arquitetura: 17](#_Toc194667498)

[Infraestrutura: 17](#_Toc194667499)

[Linguagens de Programação: 17](#_Toc194667500)

[Ferramentas AWS: 18](#_Toc194667501)

[Tipo de arquitetura da API: 20](#_Toc194667502)

[Testes na API: 21](#_Toc194667503)

[Testes no Visual Studio: 21](#_Toc194667504)

[Readme 24](#_Toc194667505)

[GIT 25](#_Toc194667506)

[CUSTOS (AWS) 26](#_Toc194667507)

[Monitoramento e Observabilidade: 27](#_Toc194667508)

[Critérios de segurança para consumo (integração) de serviços: 28](#_Toc194667509)

[DER – Diagrama de Entidade e Relacionamento 29](#_Toc194667510)

[API – Serviço de Controle de Lançamentos 30](#_Toc194667511)

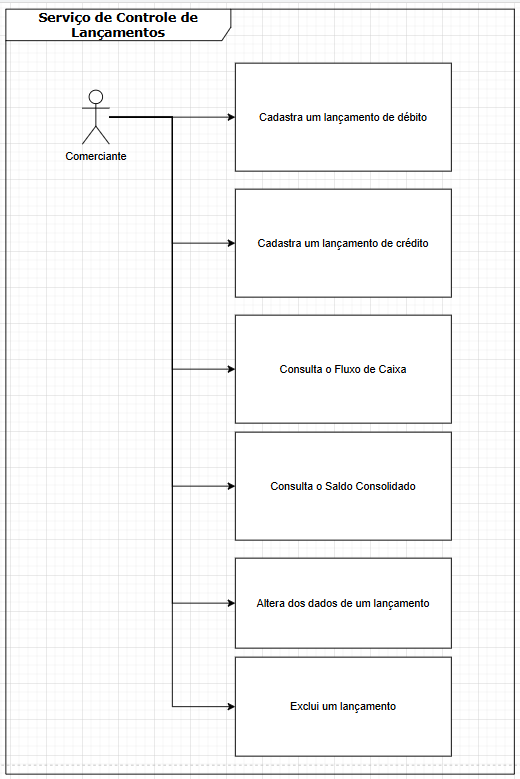
[Chamando os métodos de Saldo Consolidado: 31](#_Toc194667512)

[Chamando os métodos de lançamentos: 33](#_Toc194667513)

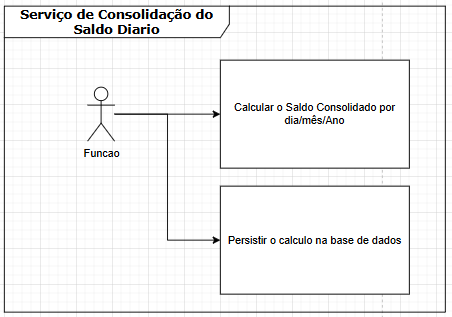
[Agradecimentos 37](#_Toc194667514)

# 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 foud), 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.](https://www.bing.com/ck/a?!&&p=fcc3be3c4259febfecffec20e4381409b97b27334d3740f31591ce9c10a8cf5bJmltdHM9MTc0MzYzODQwMA&ptn=3&ver=2&hsh=4&fclid=0b359b30-f598-6b4b-1b7d-8febf4986adc&psq=exemplo+de+requisitos+n%c3%a3o+funcionais&u=a1aHR0cHM6Ly93d3cuY29vcGVyc3lzdGVtLmNvbS5ici9yZXF1aXNpdG9zLWZ1bmNpb25haXMtZS1uYW8tZnVuY2lvbmFpcy1vLXF1ZS1zYW8tZS1xdWFsLWUtYS1kaWZlcmVuY2Ev&ntb=1)
* **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**.

o Interface gráfica do usuário, Texto, Email, Site

O conteúdo gerado por IA pode estar incorreto.

**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 pode 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**

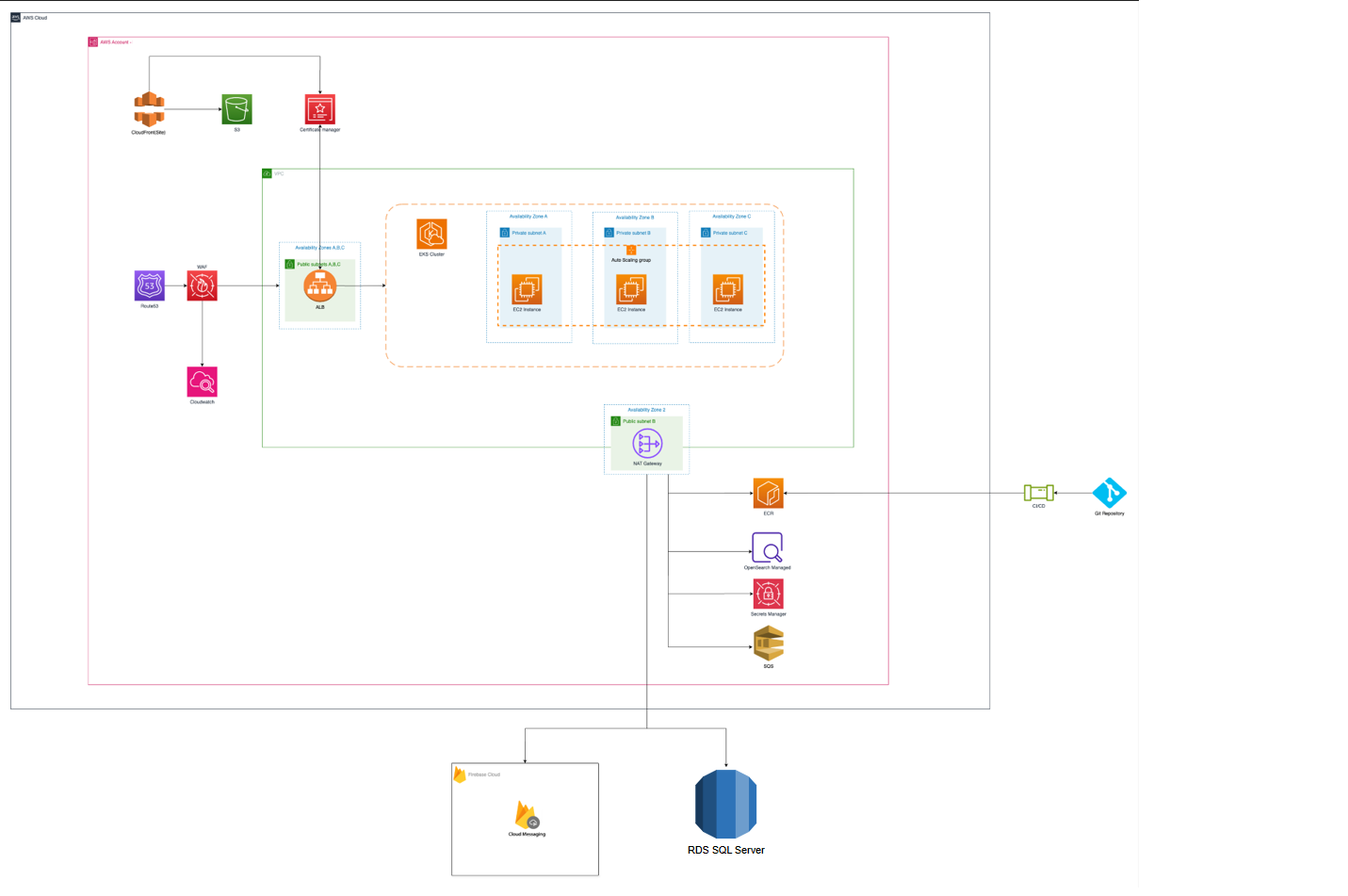
Durante a fase de testes, foi identificado que é possível gerar código de segurança que pode ser enviado para qualquer número independente se o CPF é válido, gerando com isso gastos com o envio de SMS. Basta o usuário informar o número do celular para o qual o código será enviado e automatizar a requisição.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

**Recomendação:**

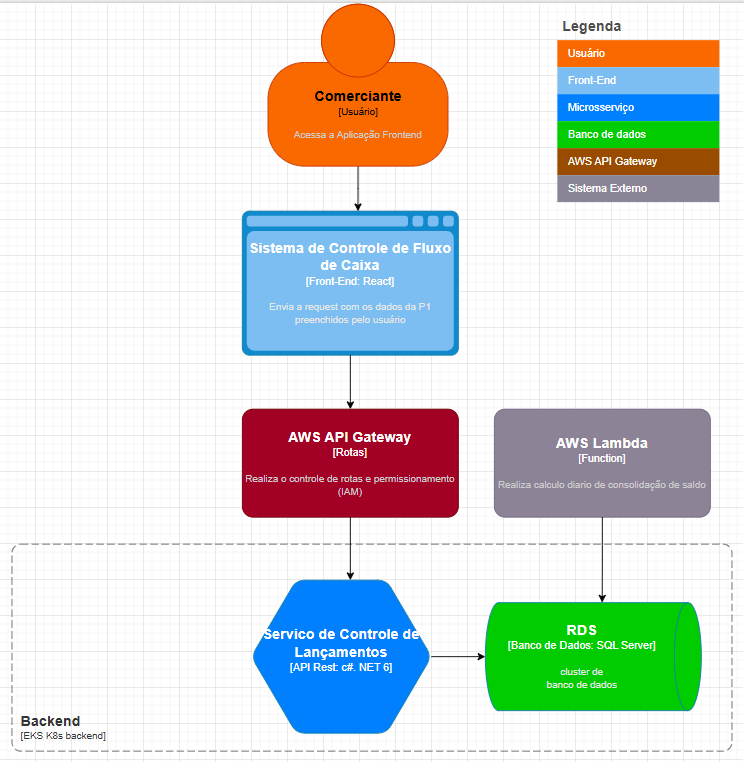
É preciso implementar o rate limit para um tempo maior no envio de SMS para evitar que um usuário malicioso consiga automatizar o envio desnecessário de tokens via SMS.

# Desenho da solução completo (Arquitetura Alvo):

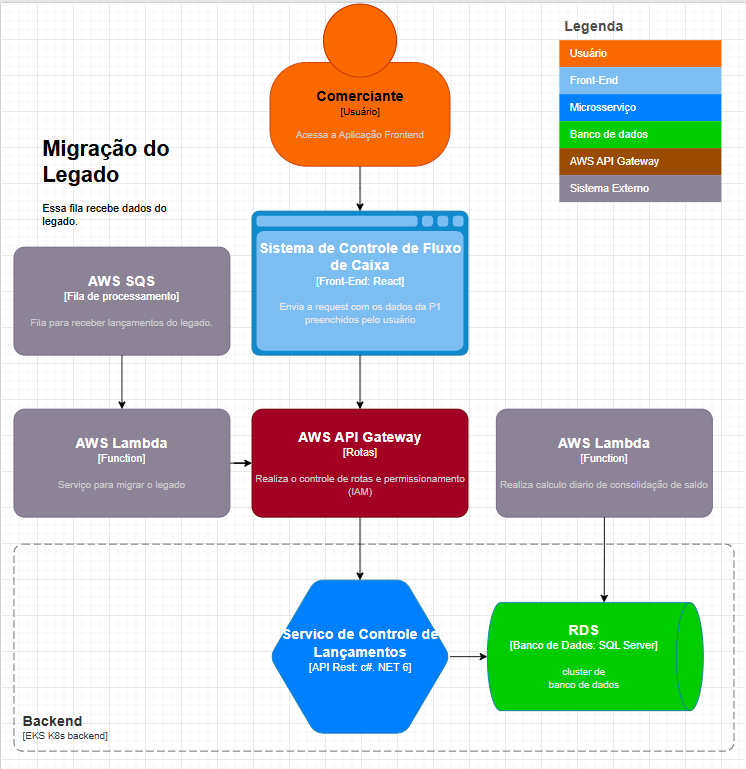
Esta documentação consiste em especificar recursos utilizados na **infraestrutura** do projeto de **Fluxo de Caixa**, listando-as em modo geral e como se integram. Também aborda brevemente a stack de tecnologias utilizadas no backend e frontend.

****

# 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 reprocessamento 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:**

Diagrama

O conteúdo gerado por IA pode estar incorreto.

# 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 Distruibuições **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)

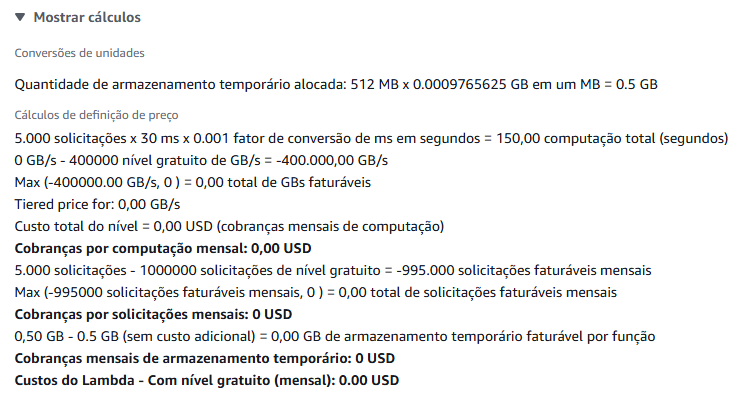
## Linguagens de Programação:

• ReactJS v20.0.0 e Typescript v4.9.5 Frontend WEB;

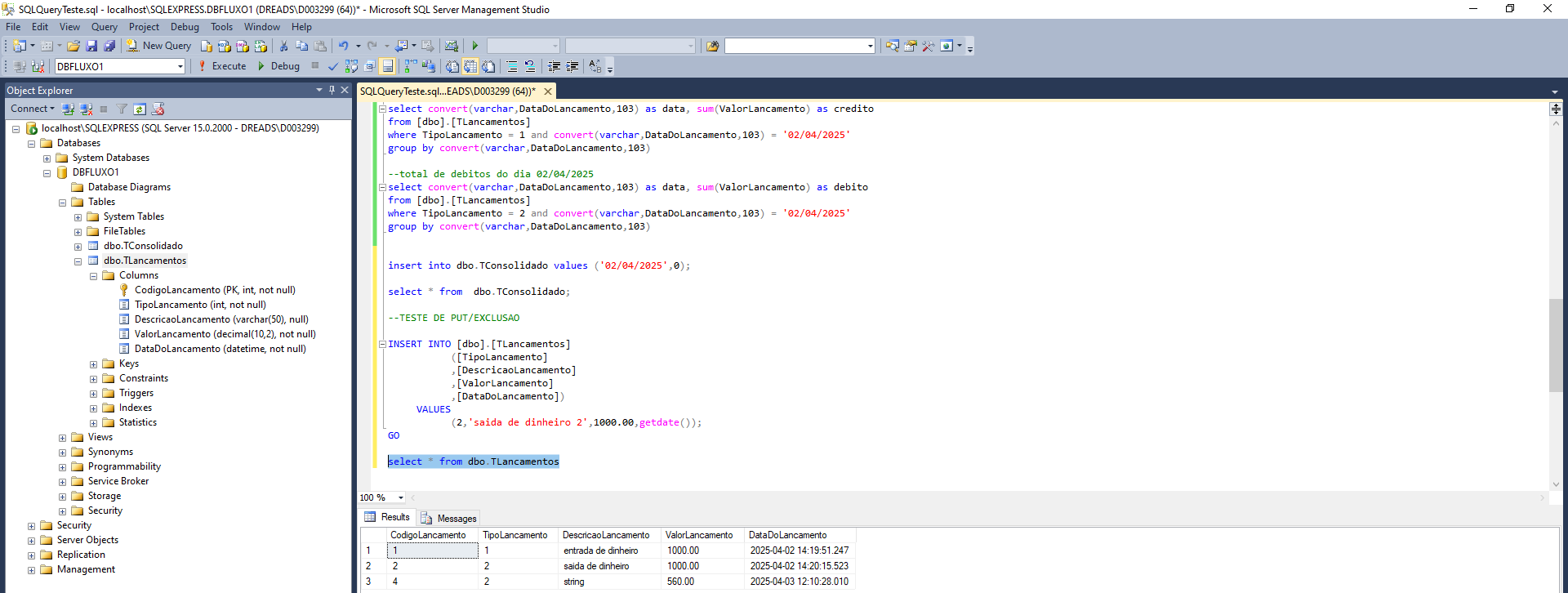
• C#.Net 9.0 na API e nas lambdas (Functions - Serveless) Backend;

Ferramentas AWS:

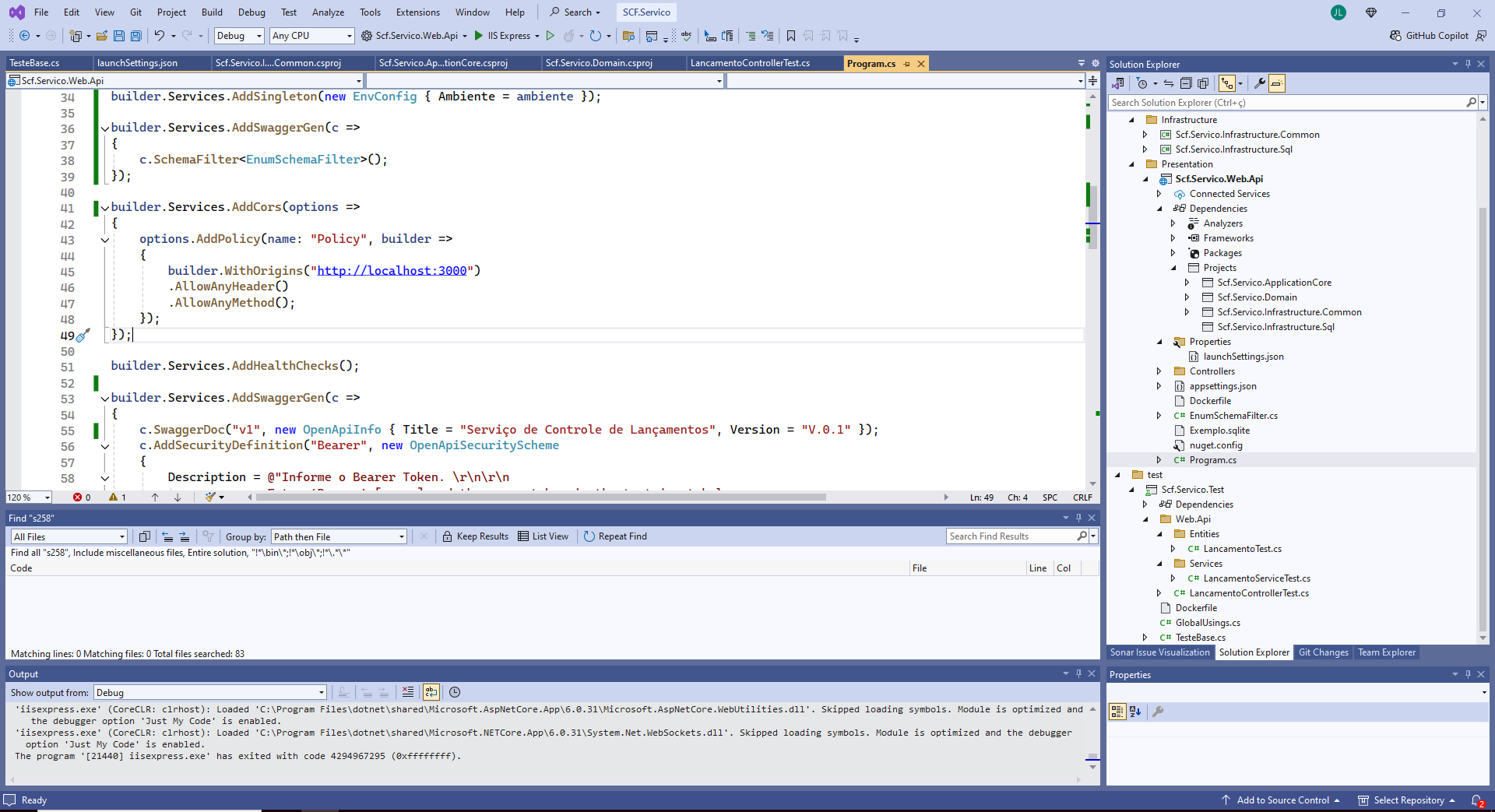
[Calculadora de preços da AWS](https://calculator.aws/#/)



**SQL Managment Server para ambiente local:**



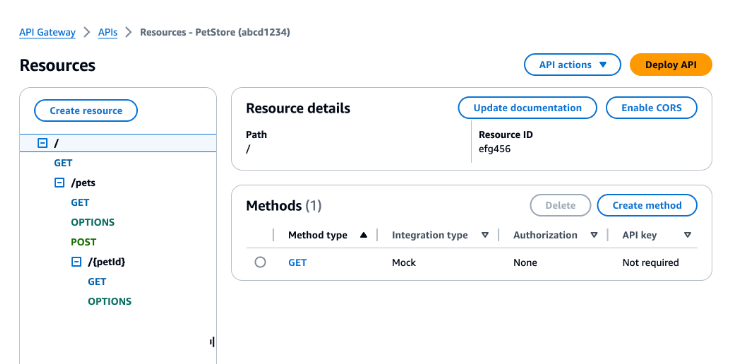
**Visual Studio 2022:**



**AWS API Gateway**

**Manual AWS passo a passo:** [Tutorial: Criar uma API REST importando um exemplo - Amazon API Gateway](https://docs.aws.amazon.com/pt_br/apigateway/latest/developerguide/api-gateway-create-api-from-example.html)

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.

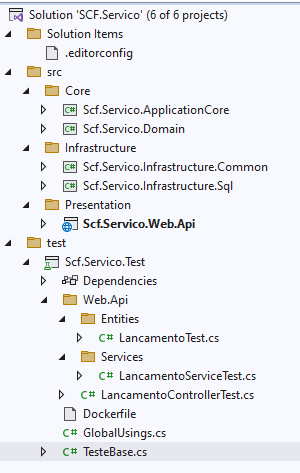


# Tipo de arquitetura da API:

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](https://www.bing.com/ck/a?!&&p=3f57e732a270aef0b2a77276e75a436e5e0ce0297b559fe0c09e15f139573d4dJmltdHM9MTc0MzYzODQwMA&ptn=3&ver=2&hsh=4&fclid=0b359b30-f598-6b4b-1b7d-8febf4986adc&u=a1aHR0cHM6Ly9wdC5zdGFja292ZXJmbG93LmNvbS9xdWVzdGlvbnMvMTk1NDgvby1xdWUtcmVhbG1lbnRlLSVDMyVBOS1kZGQtZS1xdWFuZG8tZWxlLXNlLWFwbGljYQ&ntb=1). [O DDD fornece uma estrutura para tomada de decisões, combinando práticas de design e desenvolvimento de software](https://www.bing.com/ck/a?!&&p=1a17cc217931cbb12a489268a47b8e478667ee01455f3423d8356ee134e13ec9JmltdHM9MTc0MzYzODQwMA&ptn=3&ver=2&hsh=4&fclid=0b359b30-f598-6b4b-1b7d-8febf4986adc&u=a1aHR0cHM6Ly96dXAuY29tLmJyL2Jsb2cvZG9tYWluLWRyaXZlbi1kZXNpZ24tZGRk&ntb=1).

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)

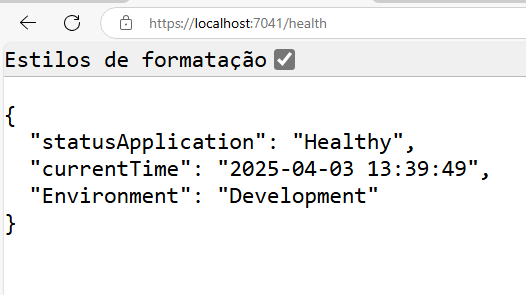


A **Clean Architecture** representa uma abordagem estratégica no desenvolvimento de software que põe ênfase na manutenção da organização e estrutura do código de forma que permaneça resiliente a mudanças, seja em tecnologia, frameworks, ou requisitos de negócios.

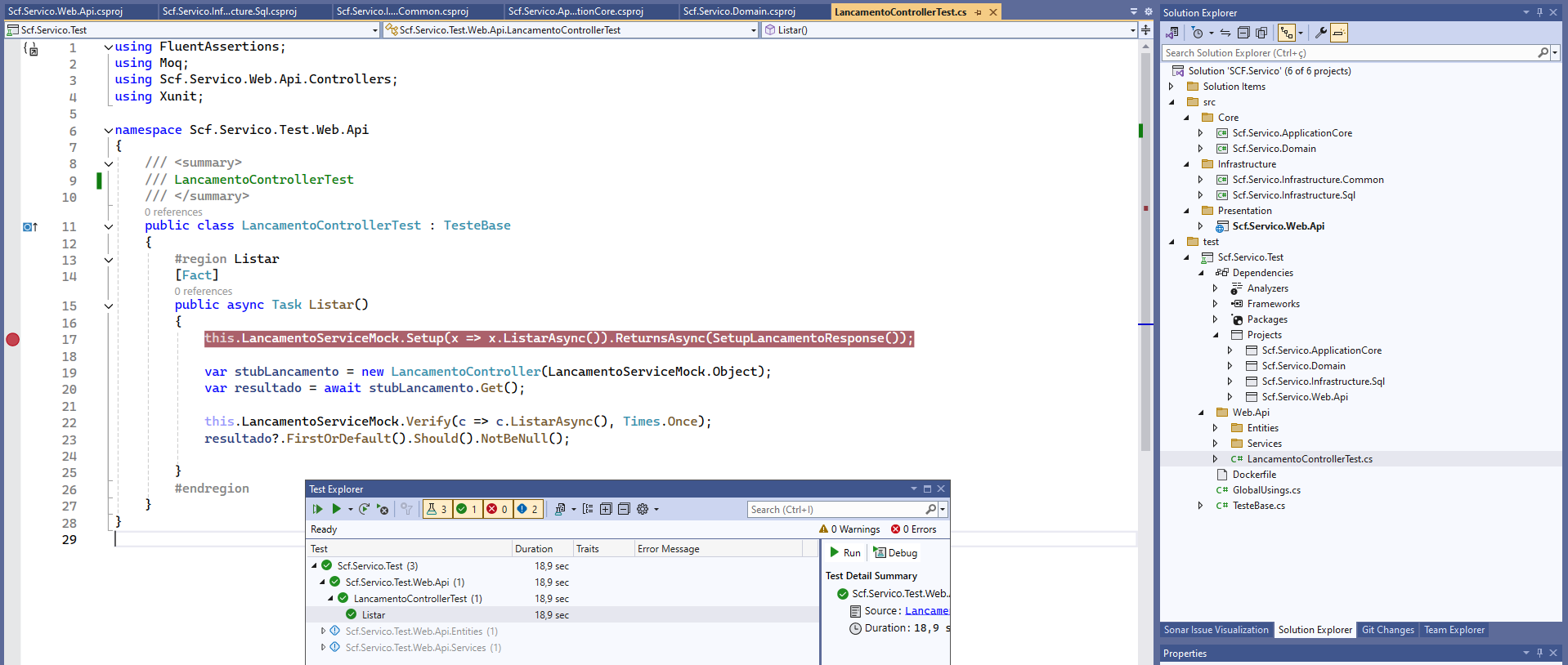
A comunicação será feita por **JSON**, [que é uma formatação utilizada para estruturar dados em formato de texto e transmiti-los de um sistema para outro, como em aplicações cliente-servidor ou em aplicativos móveis. É um formato de arquivo de texto leve, compacto, no qual os dados são guardados no formato 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”.](https://www.bing.com/ck/a?!&&p=a3ff7c380615af12894ed6510c2081117368371f6048f95f457501a71e4c8574JmltdHM9MTc0MzYzODQwMA&ptn=3&ver=2&hsh=4&fclid=0b359b30-f598-6b4b-1b7d-8febf4986adc&psq=o+que+%c3%a9+o+json&u=a1aHR0cHM6Ly9yb2NrY29udGVudC5jb20vYnIvYmxvZy9qc29uLw&ntb=1)

# 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.Servico.Web.Api.Controllers;

using Xunit;

namespace Scf.Servico.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().NotBeNull();

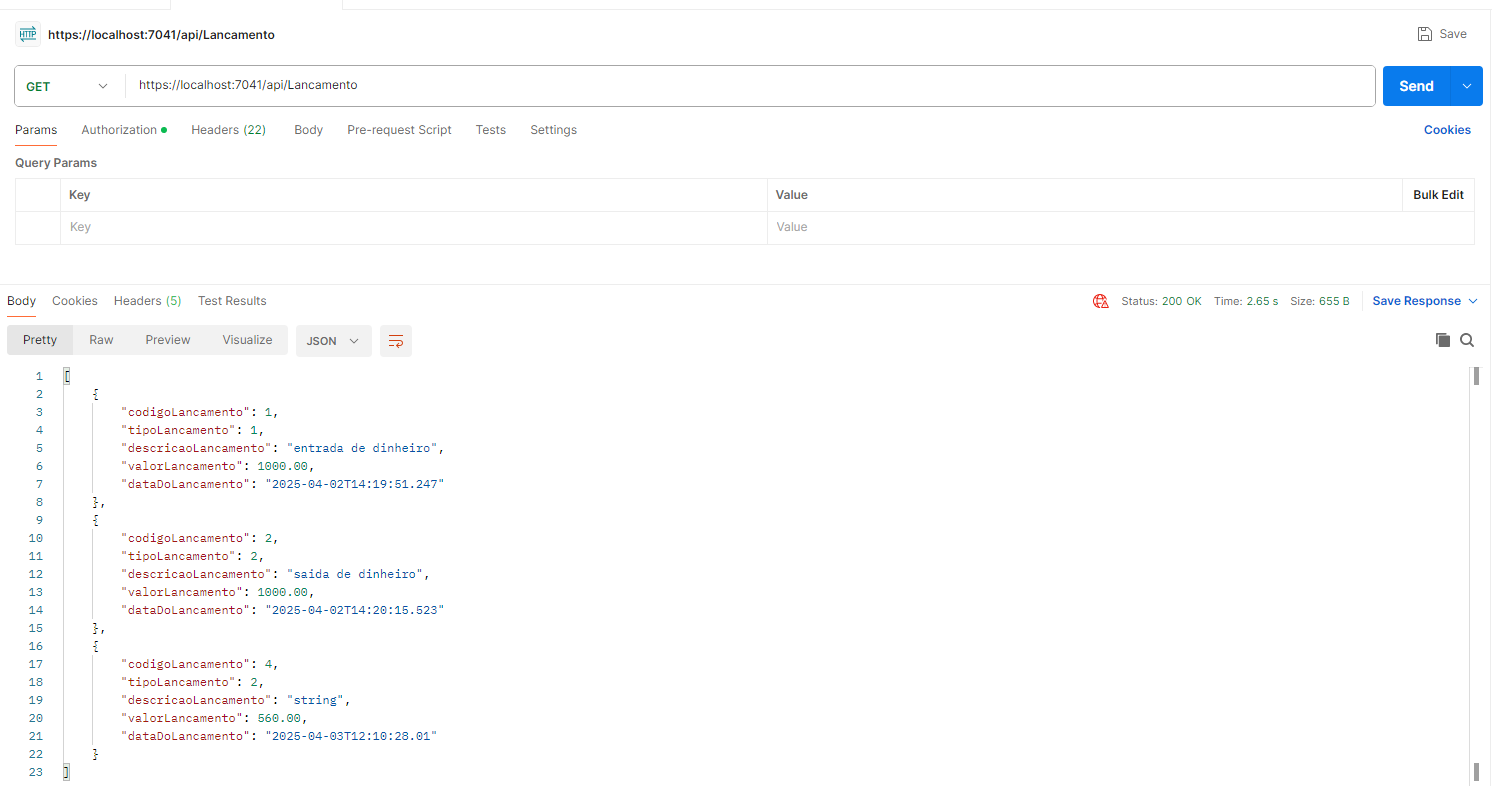
}

#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 ..j1X3OVab7kVXuUAHKgZaUQBED6FYTye0Ry6FteIpm--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**:

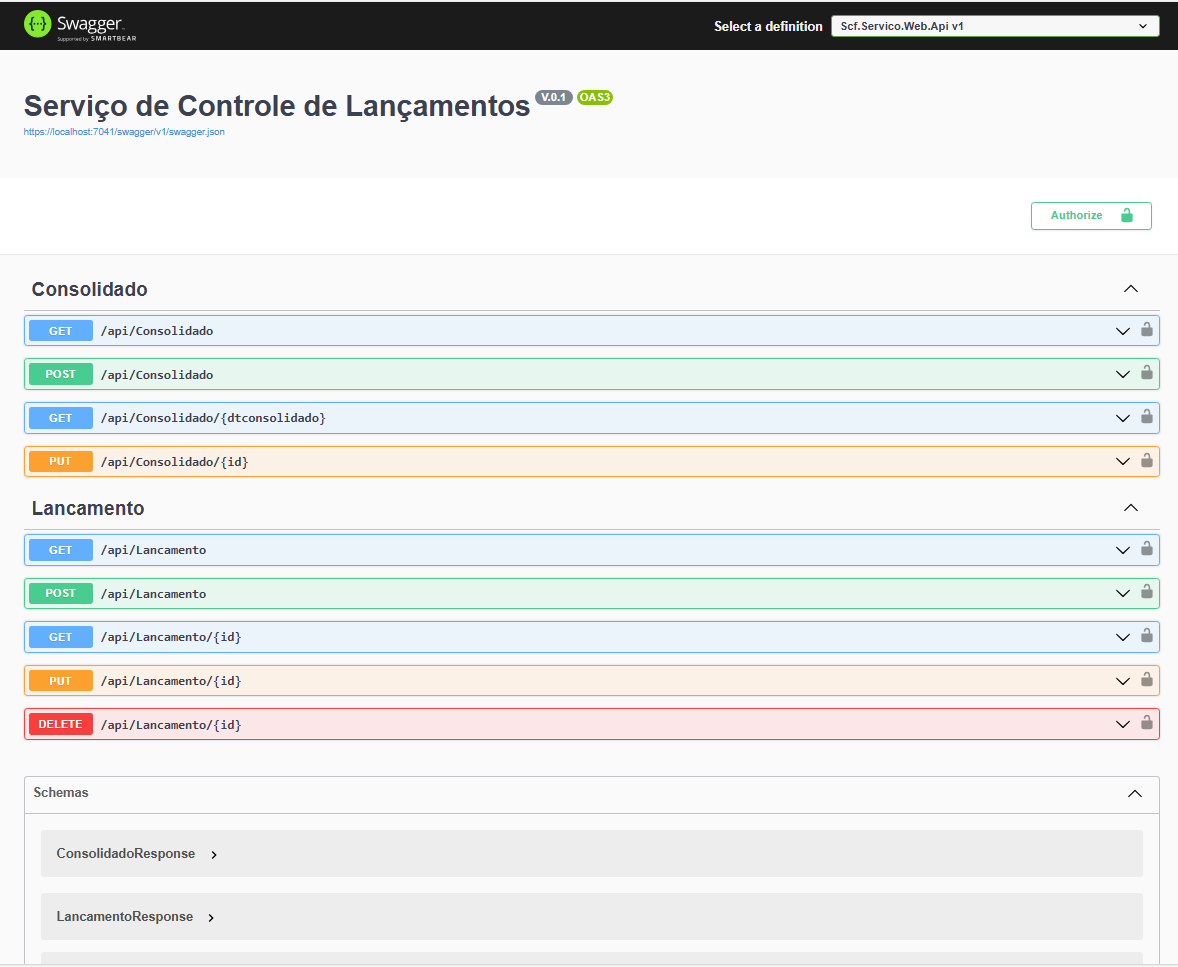
"ConnectionStrings": {

// Defina a string de conexão

"Context": "Server=localhost\\SQLEXPRESS;Database=DBFLUXO1;Trusted\_Connection=True;"

},

1. Executar o projeto da **API**;
2. Testes diretos na API com o **swagger**:



1. Configurar o **AWS API Gateway** com as rotas /lançamento e /saldo
2. 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**](https://github.com/jesseopah/fluxoCaixa/tree/master)

# CUSTOS (AWS)

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

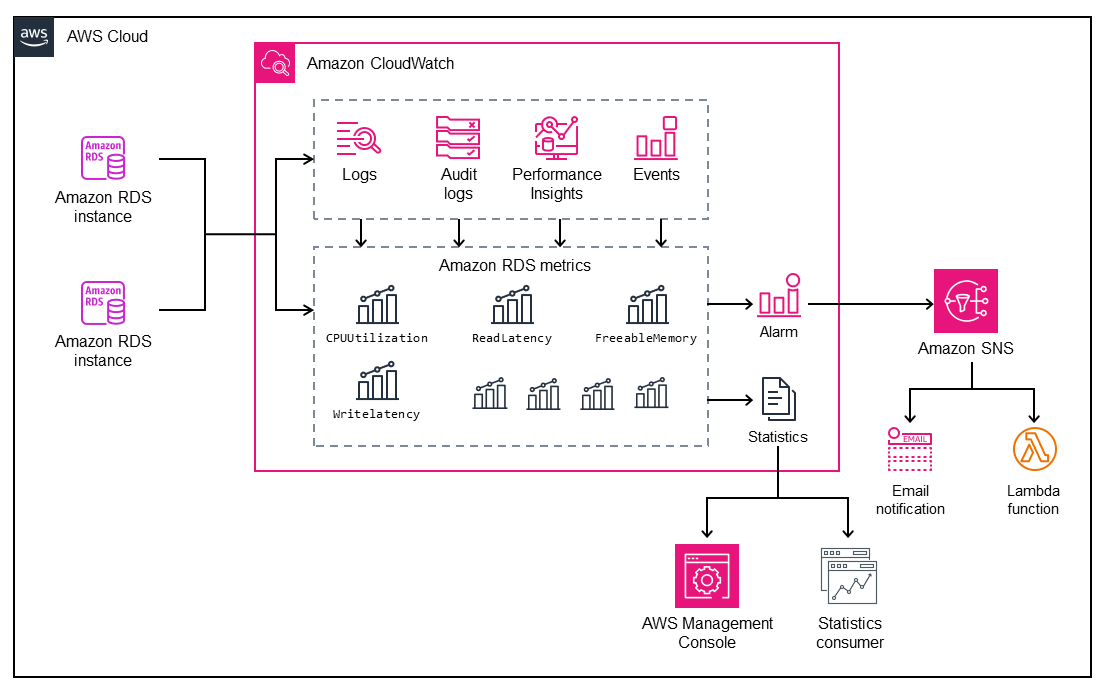
Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Podemos diminuir itens a lista conforme o budget. Itens como o Opensearch 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 [DBFLUXO1]**

**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.

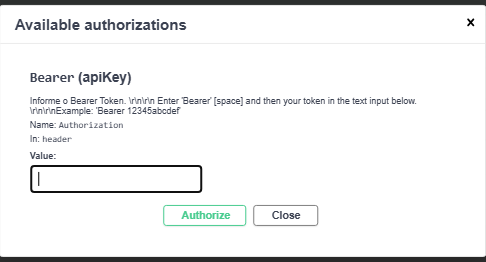
**Executando a API localmente (Swagger):** [Swagger UI](https://localhost:7041/swagger/index.html)

**Resultado esperado:**

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

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



## Chamando os métodos de Saldo Consolidado:

1. **Listar Saldos Consolidados por data:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

1. **Inserir um Saldo Consolidado para uma data específica:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

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

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

1. **Alterar um Saldo Consolidado por ID:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

## Chamando os métodos de lançamentos:

1. **Listar lançamentos:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

1. **Buscar lançamento por ID:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

1. **Alterar lançamento por ID:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Resultado esperado:**

Tabela

O conteúdo gerado por IA pode estar incorreto.

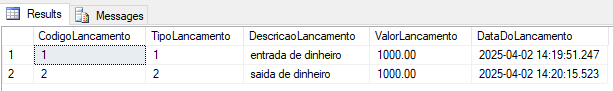
1. **Excluir um lançamento pelo ID:**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Resultado esperado:**

O registro com ID = 3 foi removido:



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

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Resultado desejado:**

Tabela

O conteúdo gerado por IA pode estar incorreto.

# Agradecimentos

**Obrigado!**

**Jesse Leandro Leoni**

**Arquiteto de Solução**

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