

Monitoramento e observabilidade de aplicações de instituições financeiras

Victor Luiz Domingues¹; Alexander Chávez López²

Backend

Informações Gerais

O 'backend' desse projeto foi feito com 4 (quatro) aplicações Web APIs, 3 (três) artefatos Class Library, 2 (dois) bancos de dados e 1 (um) 'cache', rodando em um cluster Kubernetes de 2 (dois) nós.

Informações de desenvolvimento

Sistema operacional: macOS Sequoia Version 15.4.1 (24E263)

Docker Desktop: Version 4.40.0 (187762)

Engine: 28.0.4

Compose: v2.34.0-desktop.1

Credential Helper: v0.9.3

Kubernetes: v1.32.2

IDE: JetBrains Rider 2024.3.7

Editor de texto: Visual Studio Code Version: 1.99.1 (Universal)

Acesso a dados: Azure Data Studio Version: 1.51.1

Cache: Redis 6.2

Banco de dados: PostgreSQL 10

Sistema de controle de versão: Git

Bases de dados

Foram usados dois mecanismos de armazenamento de dados levando em consideração dois aspectos, primeiramente a simplicidade, objetivo do projeto e não menos importante o teorema CAP. Considerando que a arquitetura foi projetada para trabalhar com microsserviços, em uma arquitetura distribuída o teorema CAP é fundamental para ajudar a definir a distribuição e armazenamento de dados pelas óticas de Consistência, Disponibilidade e tolerância a partição de modo que nunca é possível ter os três pilares em uma única solução. Portanto Redis e Postgres foram escolhidos, respectivamente Redis por ter mais consistência e tolerância a partição e Postgres por ter mais consistência e disponibilidade.

Redis aqui é utilizado para armazenamento de dados como saldo e sessão de cliente que podem ter um tempo de cache até ter a última versão mais quente do dado.

Postgres aqui é utilizado para armazenar dados transacionais como informação do cliente, movimentações e saldo consolidado.

Estrutura Redis

Chaves	Descrição
sessao-{idUsuario}	Dados de sessão
perfil-{idUsuario}	Dados de perfil do usuário
saldo-{idUsuario}	Dados de saldo

Tabela 1. Composição de chaves nas instancias Redis.

Estrutura Postgres

Base cadastros

01_CREATE_DB_CADASTRO.sql

```
CREATE DATABASE cadastro_db;
```

02_CREATE_CONTEXTO_CADASTRO.sql

```
DROP TABLE IF EXISTS usuarios;
```

```
CREATE TABLE usuarios (  
  id uuid PRIMARY KEY NOT NULL,  
  nome VARCHAR(300) NOT NULL,  
  cpf VARCHAR (11) NOT NULL,  
  email VARCHAR(300) UNIQUE NOT NULL,  
  senha VARCHAR(256) NOT NULL,  
  situacao SMALLINT NOT NULL,  
  criado_em TIMESTAMP NOT NULL,  
  atualizado_em TIMESTAMP,  
  deletado_em TIMESTAMP  
);
```

Base movimentações

01_CREATE_DB_MOVIMENTACOES.sql

```
CREATE DATABASE movimentacoes_db;
```

02_CREATE_CONTEXTO_MOVIMENTACOES.sql

```
DROP TABLE IF EXISTS movimentacoes;
```

```
DROP TABLE IF EXISTS saldos;
```

```
CREATE TABLE movimentacoes (  
  id uuid PRIMARY KEY NOT NULL,  
  id_usuario uuid NOT NULL,  
  valor NUMERIC(10,2) NOT NULL,  
  valor_efetivar_saldo NUMERIC(10,2) NOT NULL,  
  tipo SMALLINT NOT NULL,  
  forma SMALLINT NOT NULL,  
  destinatario TEXT,  
  banco VARCHAR(10) NULL,  
  agencia VARCHAR(10) NULL,  
  conta VARCHAR (10) NULL,  
  dac CHAR(1) NULL,  
  situacao SMALLINT NOT NULL,  
  criado_em TIMESTAMP NOT NULL,  
  atualizado_em TIMESTAMP,  
  deletado_em TIMESTAMP  
);
```

```
CREATE TABLE saldos (  
  id_usuario uuid PRIMARY KEY NOT NULL,  
  valor NUMERIC(10,2) NOT NULL,  
  situacao SMALLINT NOT NULL,  
  criado_em TIMESTAMP NOT NULL,  
  atualizado_em TIMESTAMP,  
  deletado_em TIMESTAMP  
);
```

Web API's

Plataforma: .NET 8

<https://dotnet.microsoft.com/pt-br/learn>

<https://dotnet.microsoft.com/pt-br/download/dotnet/8.0>

Tipo de API: API Mínima

<https://learn.microsoft.com/pt-br/aspnet/core/tutorials/min-web-api?view=aspnetcore-9.0&tabs=visual-studio#overview>

Imagens base:

- mcr.microsoft.com/dotnet/sdk:8.0-alpine
- mcr.microsoft.com/dotnet/aspnet:8.0-alpine
- postgres-10-alpine
- redis-6.2-alpine

Estrutura dos projetos

Estrutura 'backend' da arquitetura hexagonal com .NET.

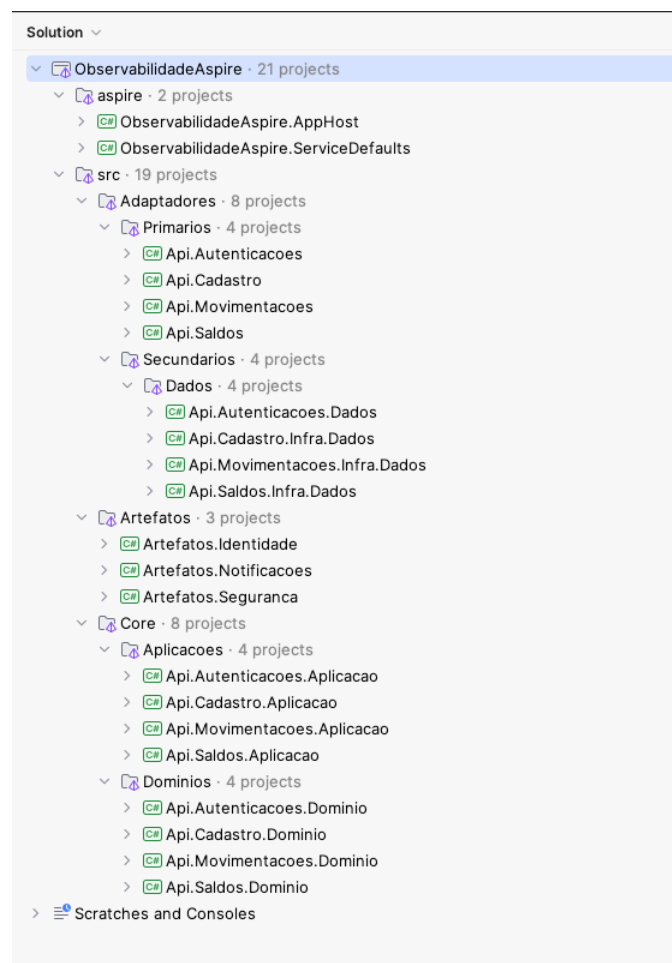
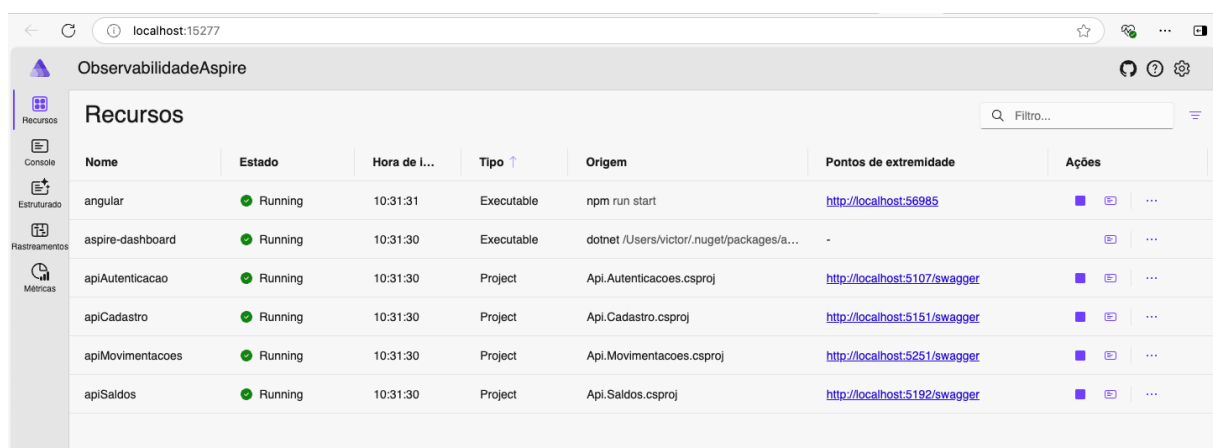


Figura 1. Modelagem do projeto backend em arquitetura hexagonal. Fonte: Elaborado pelo autor.

.NET Aspire

O .NET Aspire foi fundamental para o desenvolvimento local do 'backend', com ele facilmente subir um ambiente local com banco de dados e cache integrado antes de passar para o kubernetes a infraestrutura. Além disso ele já instrumenta para as aplicações a configuração e exposição de métricas OpenTelemetry, implementa trace e log, no entanto o seu benefício está apenas no desenvolvimento local, há ressalvas para usalo como stack de observabilidade em produção.

Com o .NET Aspire é possível orquestrar a subida de todos os microsserviços e interface de usuario em uma unica solution, você pode subir e depurar os diferentes projetos de forma isolada (em instancia separadas) mas ao mesmo tempo, como visto nas figuras a seguir.



The screenshot shows the .NET Aspire dashboard in a web browser. The title bar indicates the address is localhost:15277. The dashboard has a sidebar with navigation options: Recursos, Console, Estruturado, Rastreamentos, and Métricas. The main area is titled 'Recursos' and contains a table with the following data:

Nome	Estado	Hora de I...	Tipo	Origem	Pontos de extremidade	Ações
angular	Running	10:31:31	Executable	npm run start	http://localhost:56985	[Stop] [Refresh] [More]
aspire-dashboard	Running	10:31:30	Executable	dotnet /Users/victor/.nuget/packages/a...	-	[Stop] [Refresh] [More]
apiAutenticacao	Running	10:31:30	Project	Api.Autenticacoes.csproj	http://localhost:5107/swagger	[Stop] [Refresh] [More]
apiCadastro	Running	10:31:30	Project	Api.Cadastro.csproj	http://localhost:5151/swagger	[Stop] [Refresh] [More]
apiMovimentacoes	Running	10:31:30	Project	Api.Movimentacoes.csproj	http://localhost:5251/swagger	[Stop] [Refresh] [More]
apiSaldo	Running	10:31:30	Project	Api.Saldos.csproj	http://localhost:5192/swagger	[Stop] [Refresh] [More]

Figura 2. Dashboard .NET Aspire. Fonte: Elaborado pelo autor.

API Cadastros

API (Application Programming Interface) responsável por realizar o cadastro dos novos clientes, a seguir uma visão detalhada da API Cadastros, recursos, arquitetura e testes.

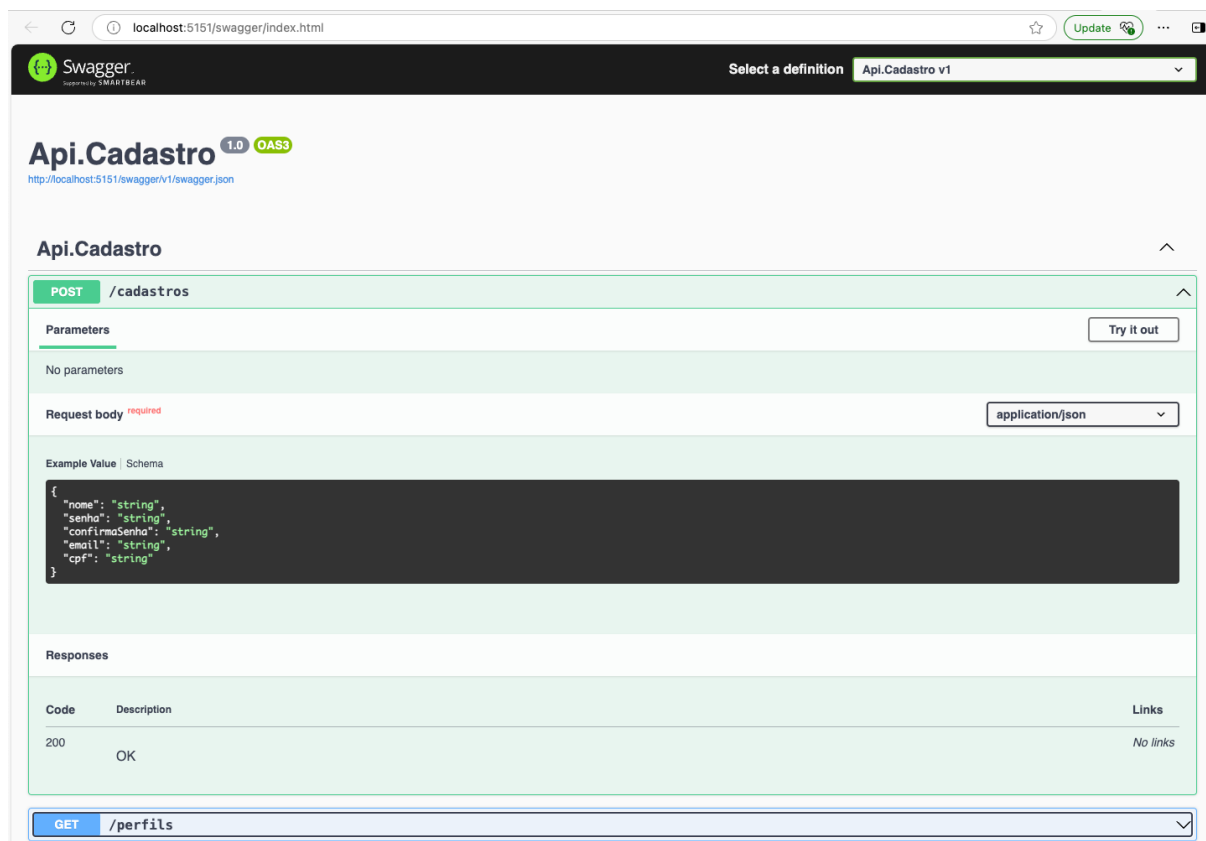


Figura 3. Contrato Swagger Open API – API Cadastros. Fonte: Elaborado pelo autor.

API Cadastros – Solução

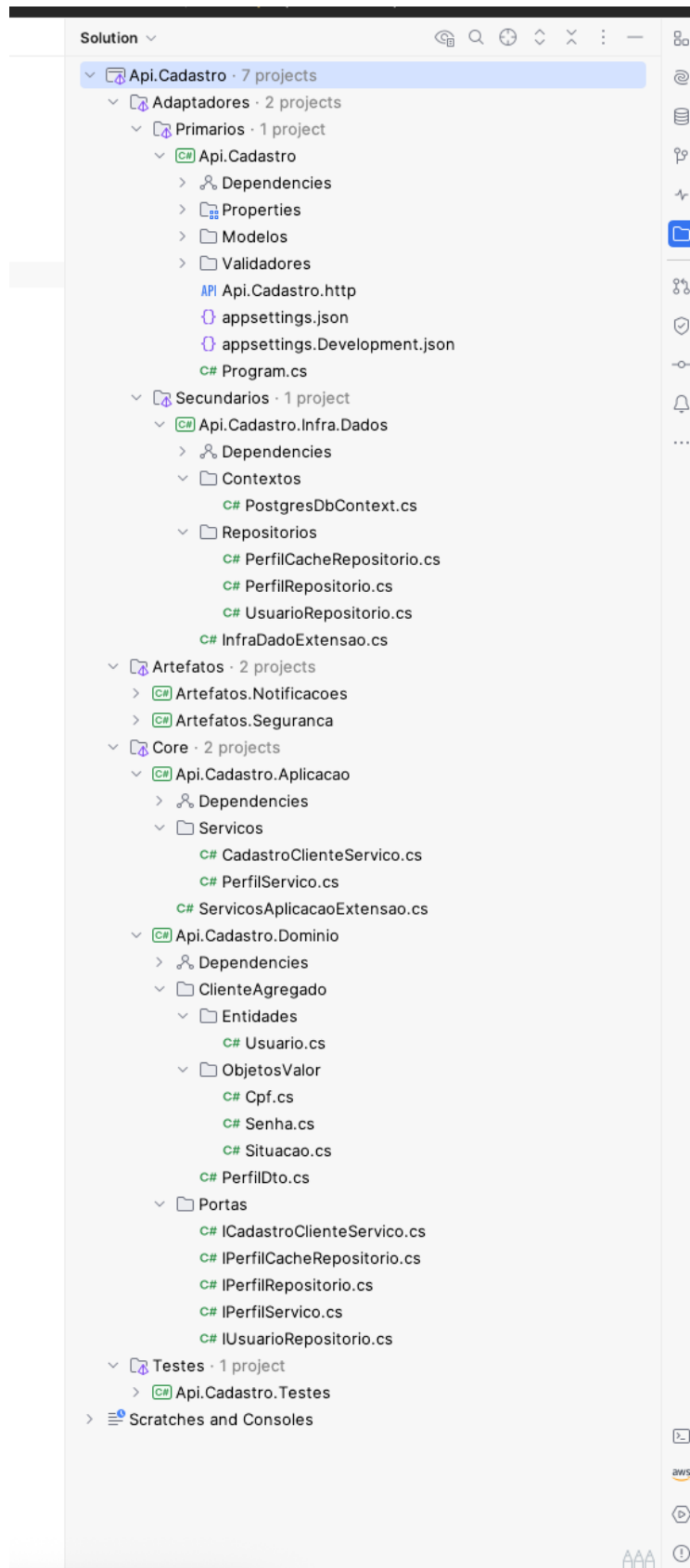


Figura 4. Solução da API Cadastros. Fonte: Elaborado pelo autor.

API Cadastros - Testes

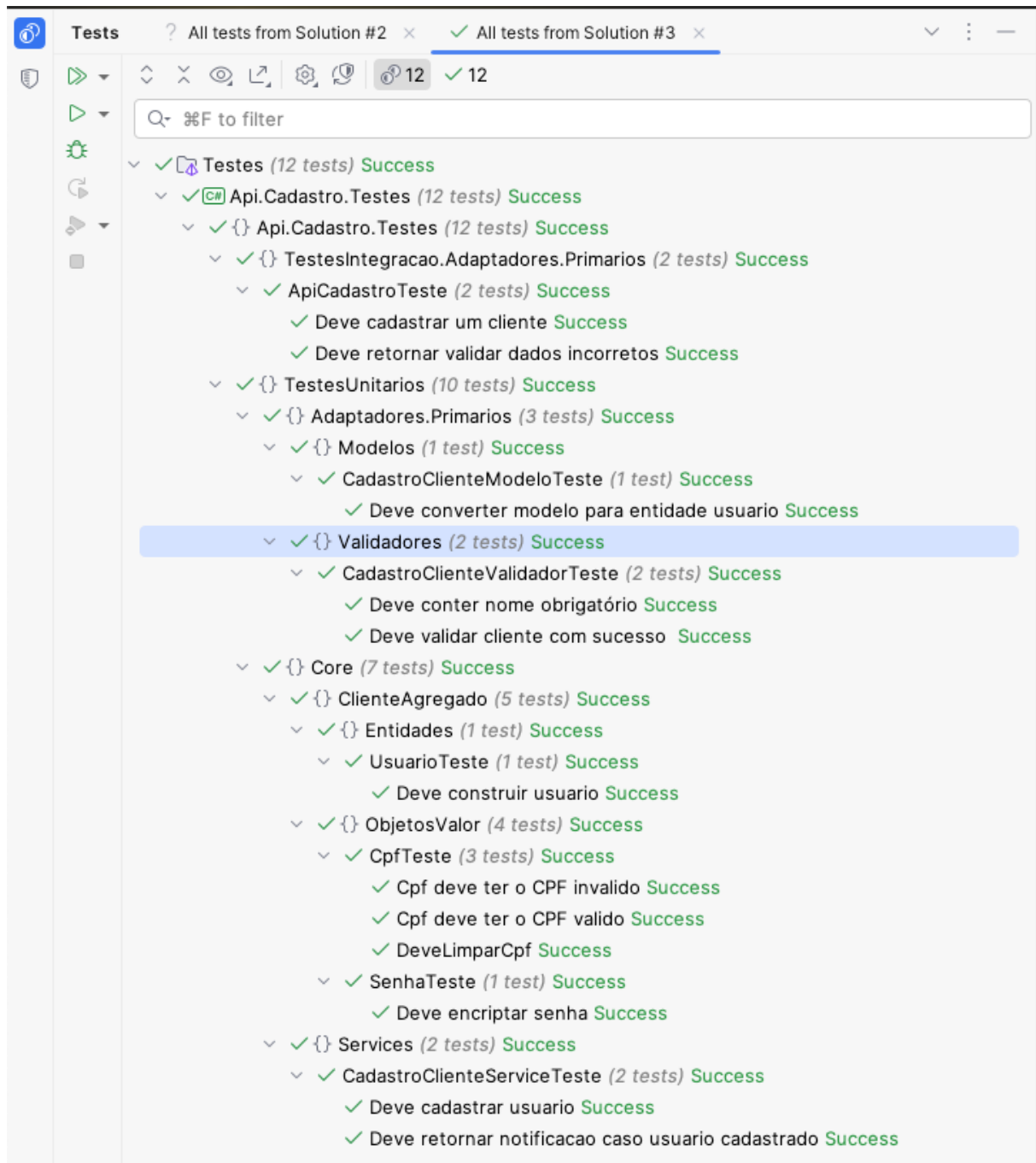


Figura 5. Testes unitários e inegração da API Cadastros. Fonte: Elaborado pelo autor.

API Cadastros – Cobertura de testes

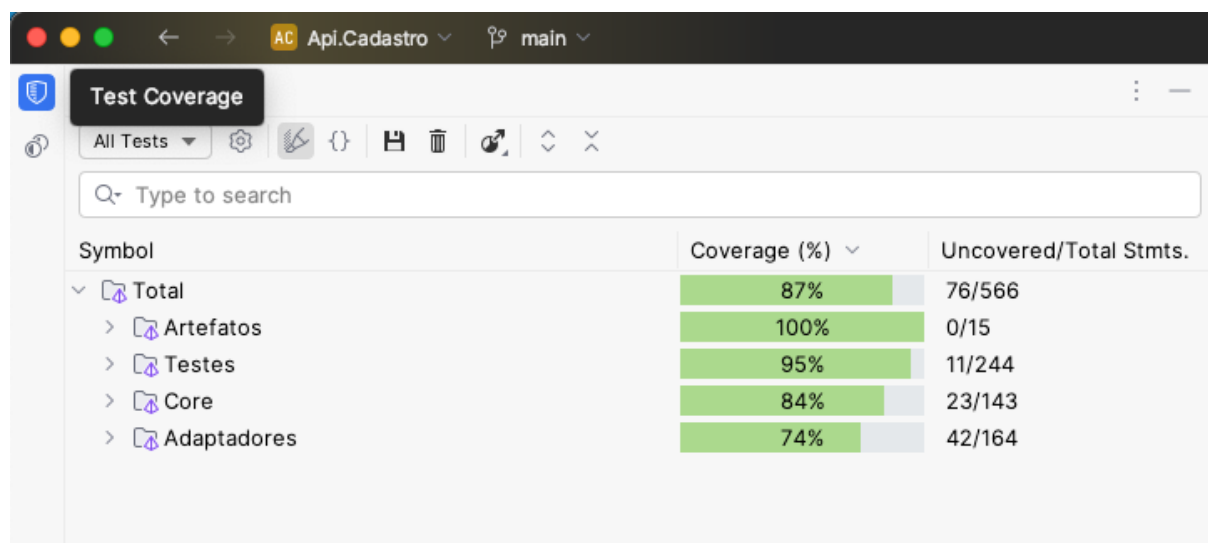


Figura 6. Percentual de cobertura da API Cadastros. Fonte: Elaborado pelo autor.

As demais API's seguem o mesmo padrão de projeto, padrão de comunicação REST (Representational State Transfer), e testes.

API Autenticações

API responsável por fazer a autenticação e autorização de clientes, gera um 'token' de acesso baseado em credenciais do cliente e valida os 'tokens' de acesso de cada API via introspecção, a seguir uma visão resumida da API Autenticações, recursos disponíveis.

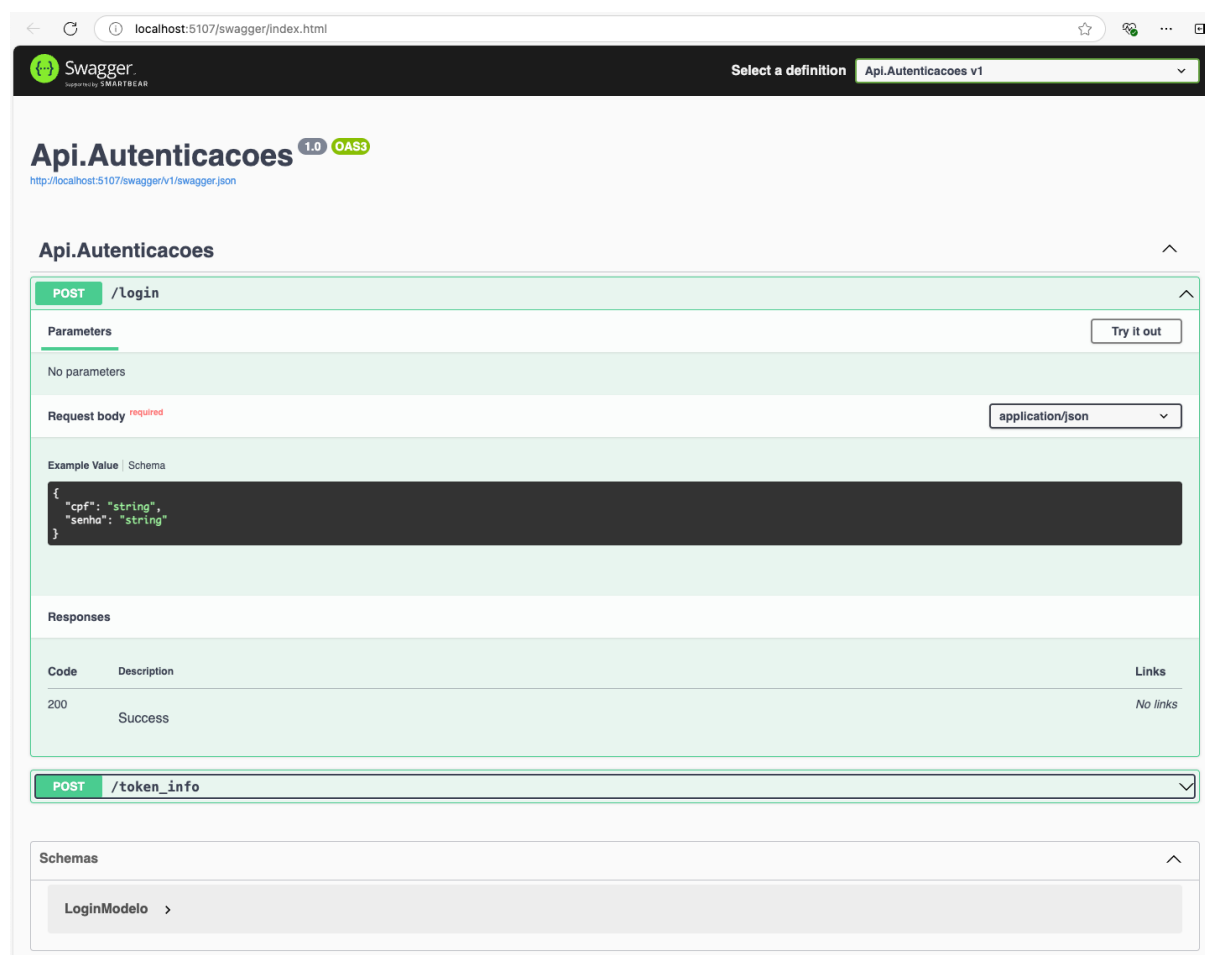


Figura 7. Contrato Swagger Open API – API Autenticações. Fonte: Elaborado pelo autor.

API Saldos

API responsável por retornar o saldo do cliente, a seguir uma visão resumida da API Saldos, recursos disponíveis.

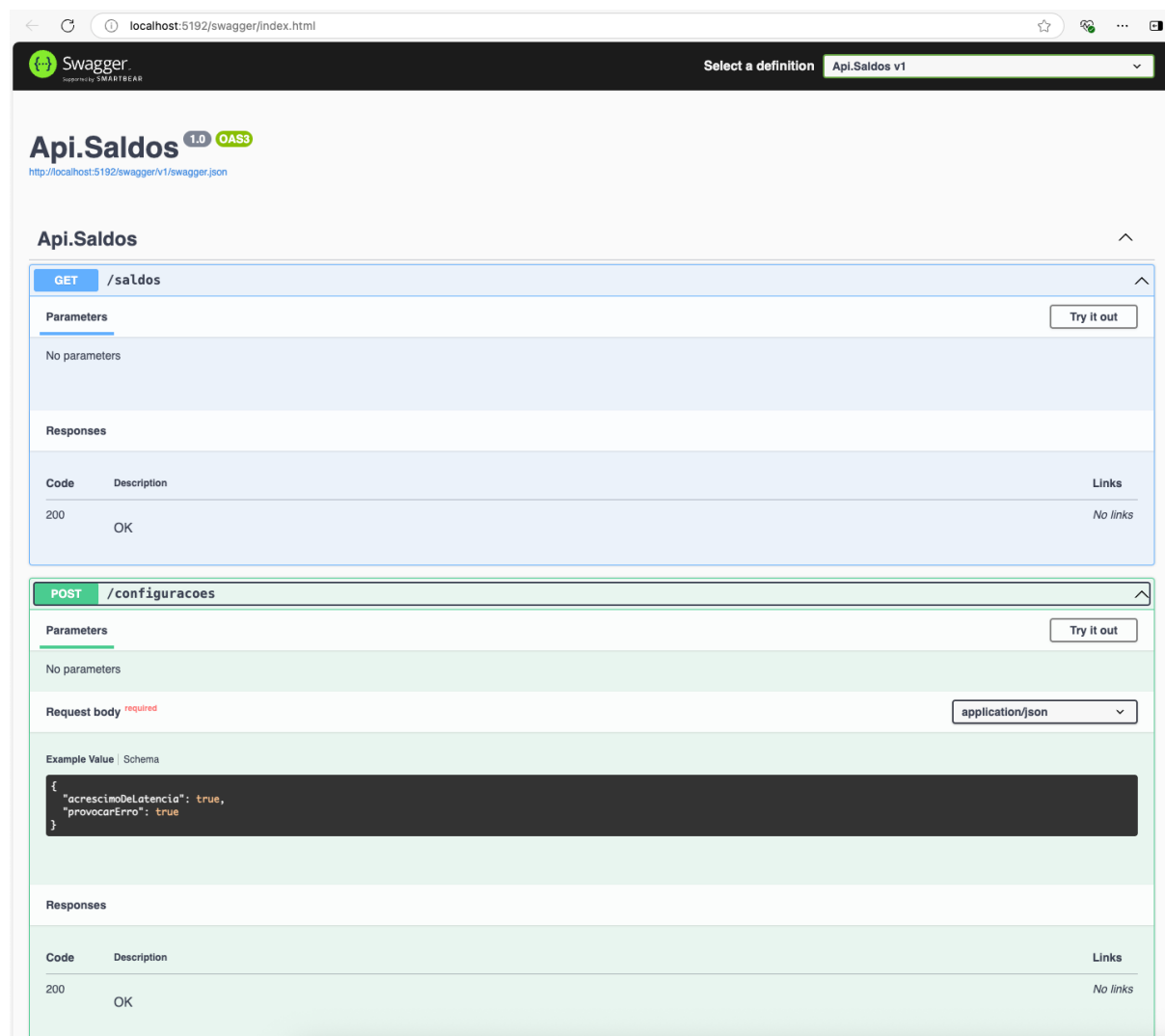


Figura 8. Contrato Swagger Open API – API Saldos. Fonte: Elaborado pelo autor.

API Movimentações

API responsável por retornar as movimentações do cliente, a seguir uma visão resumida da API Movimentações, recursos disponíveis.

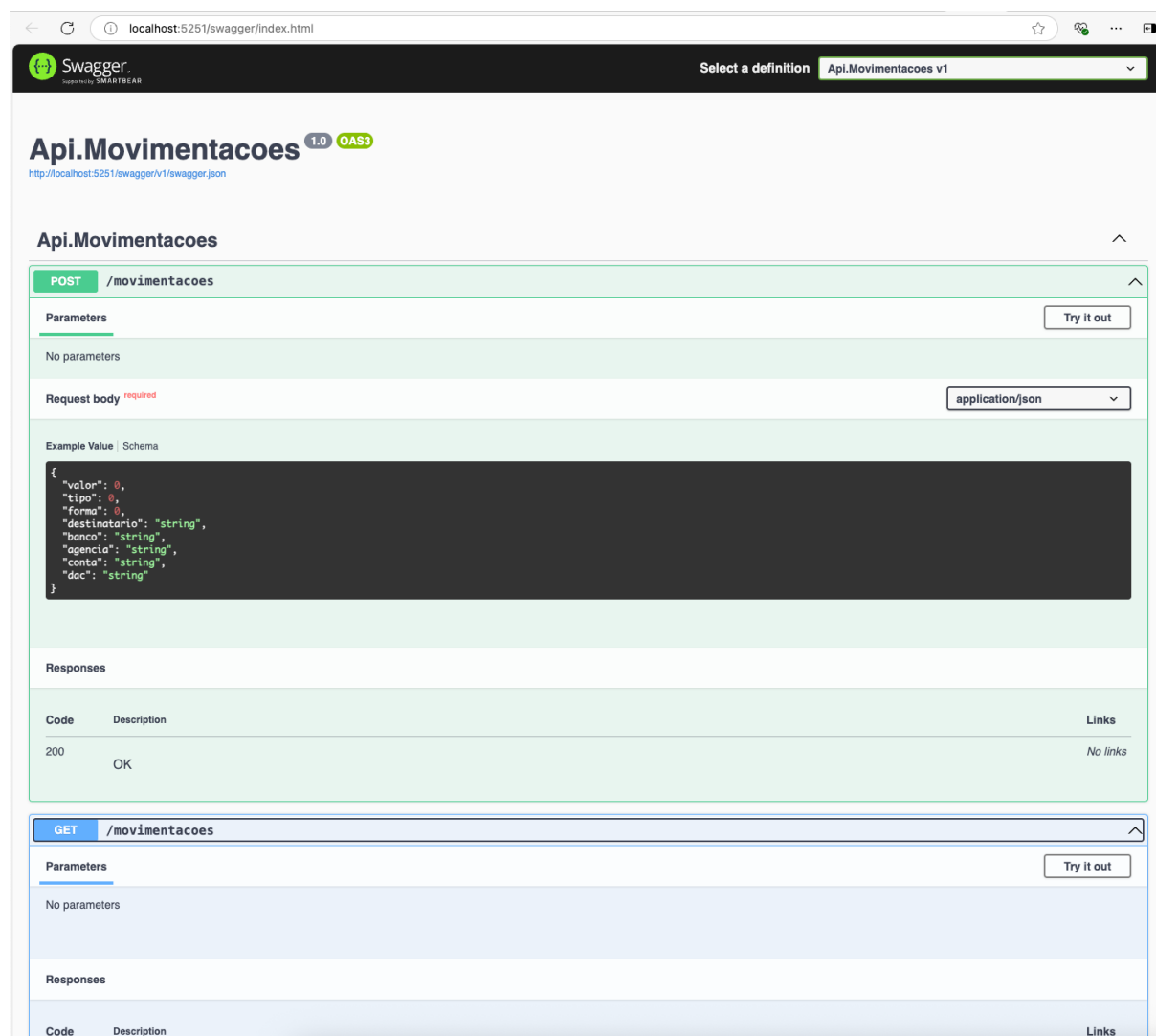


Figura 9. Contrato Swagger Open API – API Movimentações. Fonte: Elaborado pelo autor.

Autenticação e Autorização

Com exceção das API's de cadastros e autenticações todas as outras possuem recursos privados que precisa de autorização para poder ser acessadas.

O modelo de autenticação e autorização provido é baseado no 'Oauth2', dado o objetivo deste trabalho não foi implementado em sua totalidade.

O cliente faz a autenticação e é gerado um novo token JWT (Jason Web Token), após autenticado qualquer API da area restrita (logada) irá validar o token de acesso do usuário passando pelo processo de autorização, validado em cada recurso.

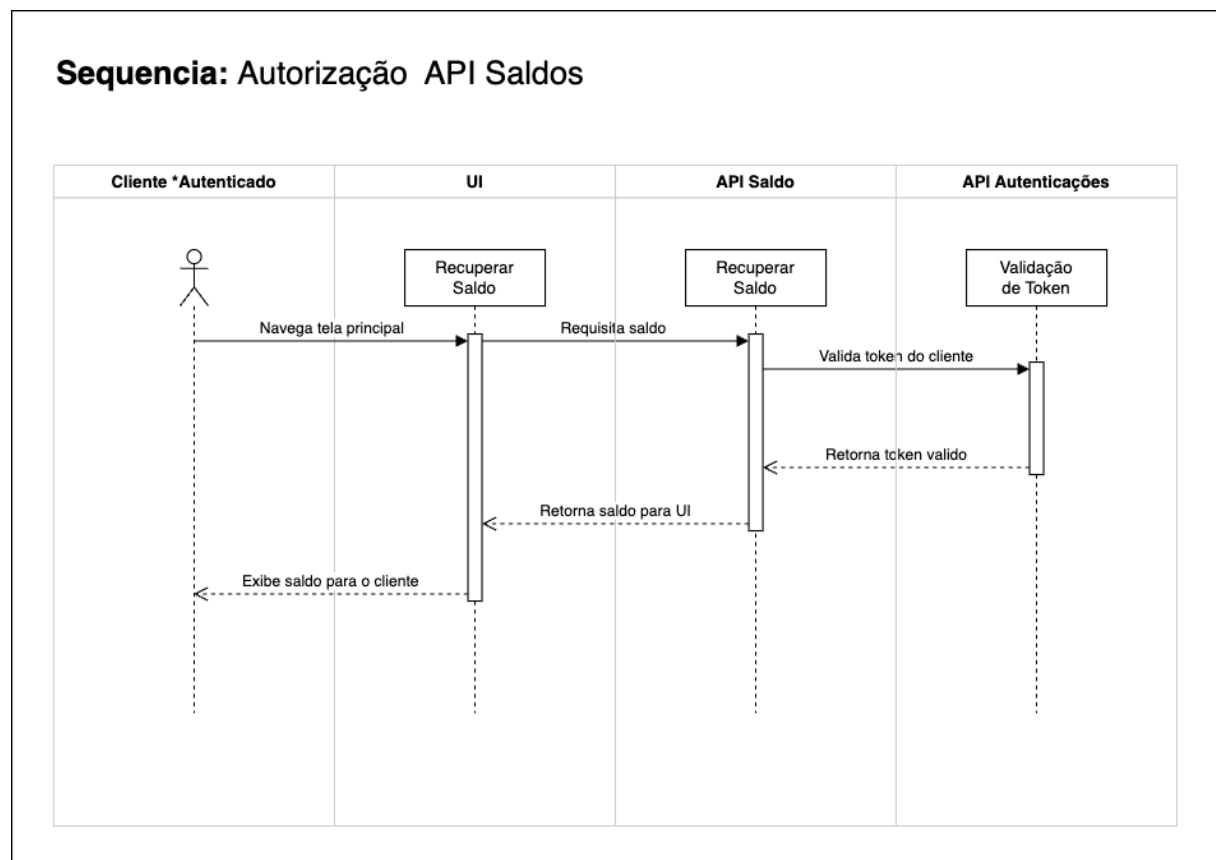


Figura 10. Diagrama de sequência para fluxo de autorização, na perspectiva da API Saldos. Fonte: Elaborado pelo autor.

Observabilidade

Para a observabilidade desse projeto foi necessária a instrumentação de algumas capacidades no backend que são fundamental para o monitoramento e observabilidade de aplicações.

Métricas

As métricas padrão podem ser instrumentadas pelo .NET Aspire usando o OpenTelemetry.

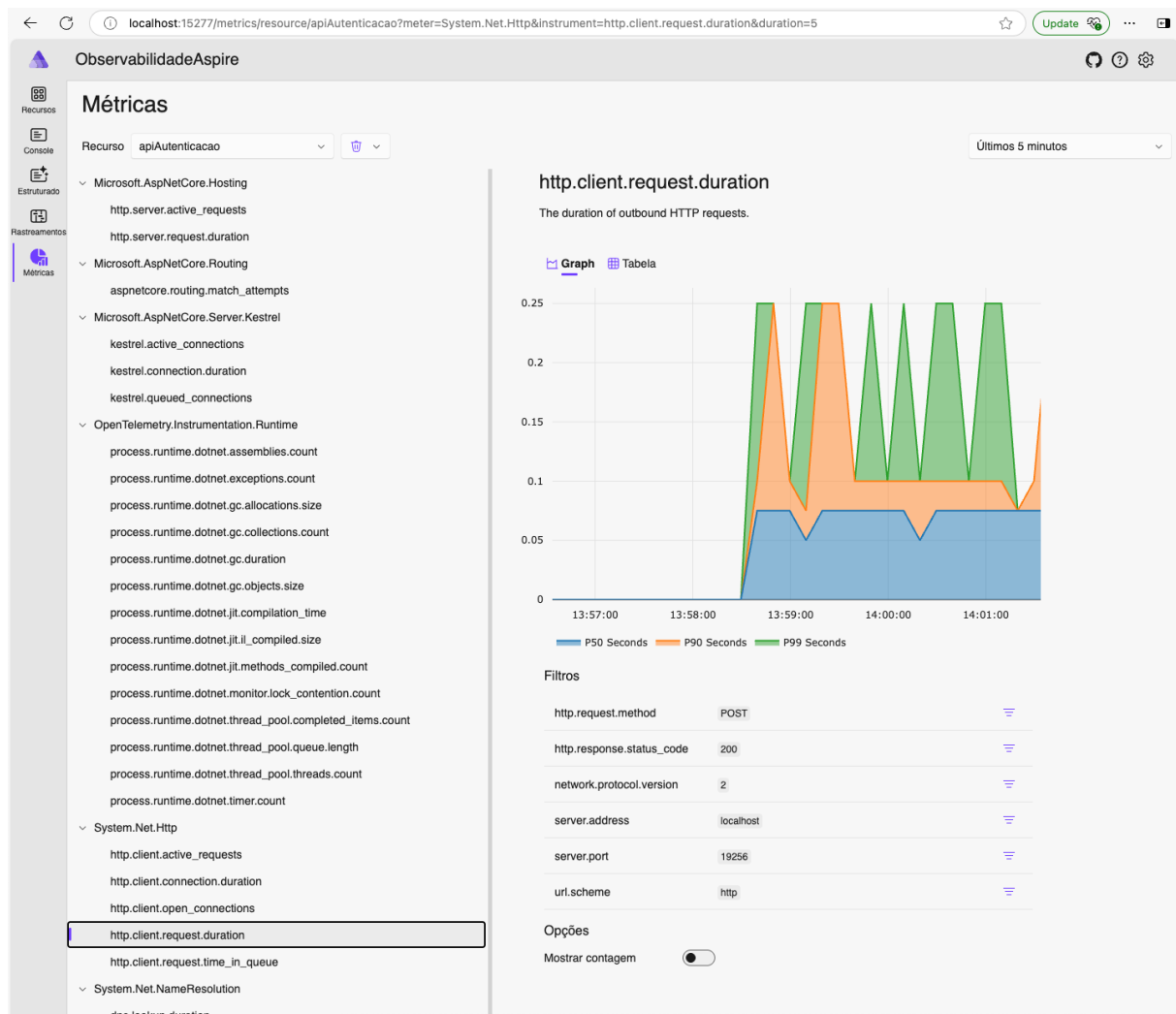


Figura 11. Métricas exportadas das API's. Fonte: Elaborado pelo autor.

Health Check

Para este projeto foi adotado os padrões de health check do .NET Aspire <https://learn.microsoft.com/en-us/dotnet/aspire/fundamentals/health-checks>.

Apesar de não ser implementado em sua totalidade nesse projeto é uma configuração fundamental para monitorar a saúde da sua aplicação bem como a saúde das suas integrações, se um serviço integrado seja ele uma API ou banco de dados estiver indisponível certamente a sua aplicação também ficará se não tiver nenhum mecanismo de resiliência ou tolerância a falhas.

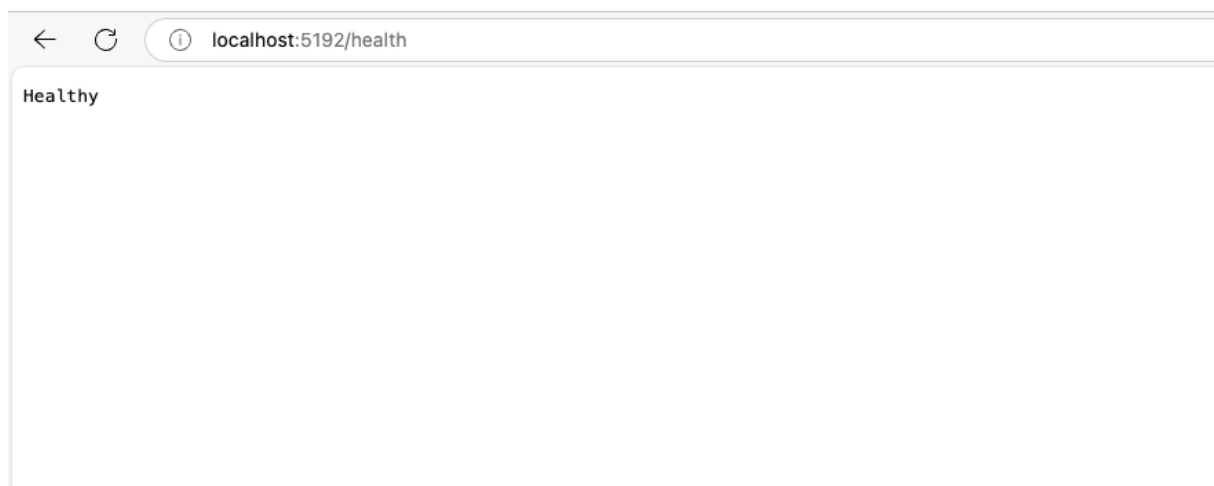


Figura 12. situação do health check da API Saldos. Fonte: Elaborado pelo autor.

Logs

Os logs foram formatados em JSON e despejados com a skin console do Serilog. Essa configuração é importante para que seja feito um log estruturado, facilita para a busca por propriedades na maioria das ferramentas de ingestão de logs, incluindo o próprio Datadog, Splunk, ou CloudWatch.

```
{
  "@t": "2025-04-19T18:06:34.4689392Z",
  "@mt": "UnprocessableEntity: Cadastro Invalido Ocorreu um erro ao cadastrar o cliente e-mail ou cpf já cadastrado. ",
  "@l": "Warning",
  "@tr": "b23774c1bd3a2435af4bff75daf8193e",
  "@sp": "e65ca6e6d3d6e11f",
  "SourceContext": "Api.Cadastro",
  "RequestId": "0HNBVB1LMDTK7:00000001",
  "RequestPath": "/cadastro",
  "ConnectionId": "0HNBVB1LMDTK7",
  "CorrelationId": "ec9ab1ac-e1fe-4ae8-abb1-9d2019ac3e1d",
  "Application": "api-cadastro",
  "Environment": "Production"
}
```

Figura 13. Saída via console da API Cadastro. Fonte: Elaborado pelo autor.

Implementação Identificador Unico de Correção

Foi implementado um middleware em cada API para que o header Correlation-Id seja injetado nos no contexto de log pré-configurado pelo Serilog.

O Middleware é um recurso nativo do .NET para intermediar a requisição e resposta HTTP de API, ou seja, quando uma request chega no servidor imediatamente antes de acessar o processamento do recurso propriamente dito o middleware trabalha enriquecendo o log.

<https://learn.microsoft.com/pt-br/aspnet/core/fundamentals/middleware/?view=aspnetcore-8.0>

```
0 references
18 public async Task InvokeAsync(HttpContext context)
19 {
20     context.Request.Headers.TryGetValue("Correlation-Id", out var correlationIds);
21     var correlationId = correlationIds.FirstOrDefault() ?? Guid.NewGuid().ToString();
22     using (LogContext.PushProperty("CorrelationId", correlationId))
23     {
24         await _next(context);
25     }
26 }
```

Figura 14. Método do Middleware CorrelationId. Fonte: Elaborado pelo autor.

Pacotes

Projeto: Api.Movimentacoes.Testes

Nome: coverlet.collector | Versão: 6.0.4
Nome: coverlet.msbuild | Versão: 6.0.4
Nome: Microsoft.AspNetCore.Mvc.Testing | Versão: 8.0.13
Nome: Microsoft.NET.Test.Sdk | Versão: 17.8.0
Nome: Moq | Versão: 4.20.72
Nome: Shouldly | Versão: 4.3.0
Nome: xunit | Versão: 2.5.3
Nome: xunit.runner.visualstudio | Versão: 2.5.3
Nome: Microsoft.EntityFrameworkCore.InMemory | Versão: 8.0.13

Projeto: Api.Movimentacoes.Infra.Dados

Nome: EFCore.NamingConventions | Versão: 8.0.3
Nome: Microsoft.EntityFrameworkCore | Versão: 8.0.13
Nome: Microsoft.EntityFrameworkCore.Abstractions | Versão: 8.0.13
Nome: Microsoft.EntityFrameworkCore.Relational | Versão: 8.0.13
Nome: Npgsql.EntityFrameworkCore.PostgreSQL | Versão: 8.0.11
Nome: Microsoft.EntityFrameworkCore.Tools | Versão: 7.0.3

Projeto: Api.Movimentacoes.Aplicacao

Nome: Microsoft.Extensions.DependencyInjection.Abstractions | Versão: 8.0.2

Projeto: Api.Movimentacoes

Nome: FluentValidation.AspNetCore | Versão: 11.3.0
Nome: Microsoft.AspNetCore.OpenApi | Versão: 8.0.3
Nome: Serilog.AspNetCore | Versão: 9.0.0
Nome: Serilog.Settings.Configuration | Versão: 9.0.1-dev-02317
Nome: Serilog.Sinks.Console | Versão: 6.0.1-dev-00953
Nome: Swashbuckle.AspNetCore | Versão: 6.4.0
Nome: IdentityModel.AspNetCore.OAuth2Introspection | Versão: 6.2.0

Projeto: Api.Cadastro.Aplicacao

Nome: Microsoft.Extensions.DependencyInjection.Abstractions | Versão: 8.0.2

Projeto: Api.Cadastro.Testes

Nome: coverlet.collector | Versão: 6.0.4
Nome: coverlet.msbuild | Versão: 6.0.4
Nome: Microsoft.AspNetCore.Mvc.Testing | Versão: 8.0.13
Nome: Microsoft.NET.Test.Sdk | Versão: 17.8.0
Nome: Moq | Versão: 4.20.72
Nome: Shouldly | Versão: 4.3.0
Nome: xunit | Versão: 2.5.3
Nome: xunit.runner.visualstudio | Versão: 2.5.3

Projeto: Api.Cadastro.Infra.Dados

Nome: StackExchange.Redis | Versão: 2.8.31
Nome: EFCore.NamingConventions | Versão: 8.0.3
Nome: Microsoft.EntityFrameworkCore | Versão: 8.0.13
Nome: Microsoft.EntityFrameworkCore.Abstractions | Versão: 8.0.13
Nome: Microsoft.EntityFrameworkCore.InMemory | Versão: 8.0.13
Nome: Microsoft.EntityFrameworkCore.Relational | Versão: 8.0.13
Nome: Npgsql.EntityFrameworkCore.PostgreSQL | Versão: 8.0.11
Nome: Microsoft.EntityFrameworkCore.Tools | Versão: 7.0.3

Projeto: Api.Cadastro

Nome: FluentValidation.AspNetCore | Versão: 11.3.0
Nome: IdentityModel.AspNetCore.OAuth2Introspection | Versão: 6.2.0
Nome: Microsoft.AspNetCore.OpenApi | Versão: 8.0.3
Nome: Serilog.AspNetCore | Versão: 9.0.0
Nome: Serilog.Settings.Configuration | Versão: 9.0.1-dev-02317
Nome: Serilog.Sinks.Console | Versão: 6.0.1-dev-00953
Nome: Swashbuckle.AspNetCore | Versão: 6.4.0

Projeto: ObservabilidadeAspire.ServiceDefaults

Nome: Microsoft.Extensions.Http.Resilience | Versão: 9.2.0
Nome: Microsoft.Extensions.ServiceDiscovery | Versão: 9.1.0
Nome: OpenTelemetry.Exporter.OpenTelemetryProtocol | Versão: 1.9.0
Nome: OpenTelemetry.Extensions.Hosting | Versão: 1.9.0
Nome: OpenTelemetry.Instrumentation.AspNetCore | Versão: 1.9.0
Nome: OpenTelemetry.Instrumentation.Http | Versão: 1.9.0
Nome: OpenTelemetry.Instrumentation.Runtime | Versão: 1.9.0

Projeto: ObservabilidadeAspire.AppHost

Nome: Aspire.Hosting.AppHost | Versão: 9.1.0
Nome: Aspire.Hosting.NodeJs | Versão: 9.1.0
Nome: Aspire.Hosting.PostgreSQL | Versão: 9.1.0
Nome: Aspire.Hosting.Redis | Versão: 9.1.0
Nome: Aspire.Npgsql | Versão: 9.1.0

Projeto: Api.Autenticacoes.Testes

Nome: coverlet.collector | Versão: 6.0.0
Nome: Microsoft.NET.Test.Sdk | Versão: 17.8.0
Nome: xunit | Versão: 2.5.3
Nome: xunit.runner.visualstudio | Versão: 2.5.3
Nome: Microsoft.AspNetCore.Mvc.Testing | Versão: 8.0.13
Nome: Microsoft.NET.Test.Sdk | Versão: 17.8.0
Nome: Moq | Versão: 4.20.72
Nome: Shouldly | Versão: 4.3.0
Nome: xunit | Versão: 2.5.3

Nome: xunit.runner.visualstudio | Versão: 2.5.3

Nome: Microsoft.EntityFrameworkCore.InMemory | Versão: 8.0.13

Projeto: Api.Autenticacoes.Aplicacao

Nome: Microsoft.Extensions.DependencyInjection.Abstractions | Versão: 8.0.2

Nome: Microsoft.AspNetCore.Authentication.JwtBearer | Versão: 8.0.13

Projeto: Api.Autenticacoes.Dados

Nome: StackExchange.Redis | Versão: 2.8.31

Nome: EFCore.NamingConventions | Versão: 8.0.3

Nome: Microsoft.EntityFrameworkCore | Versão: 8.0.13

Nome: Microsoft.EntityFrameworkCore.Abstractions | Versão: 8.0.13

Nome: Microsoft.EntityFrameworkCore.Relational | Versão: 8.0.13

Nome: Npgsql.EntityFrameworkCore.PostgreSQL | Versão: 8.0.11

Projeto: Api.Autenticacoes

Nome: Aspire.Npgsql.EntityFrameworkCore.PostgreSQL | Versão: 9.1.0

Nome: Aspire.StackExchange.Redis.DistributedCaching | Versão: 9.1.0

Nome: Microsoft.AspNetCore.OpenApi | Versão: 8.0.3

Nome: Serilog.AspNetCore | Versão: 9.0.0

Nome: Serilog.Settings.Configuration | Versão: 9.0.1-dev-02317

Nome: Serilog.Sinks.Console | Versão: 6.0.1-dev-00953

Nome: Swashbuckle.AspNetCore | Versão: 6.4.0

Projeto: Api.Saldos.Aplicacao

Nome: Microsoft.Extensions.DependencyInjection.Abstractions | Versão: 8.0.2

Projeto: Api.Saldos.Infra.Dados

Nome: Dapper | Versão: 2.1.66

Nome: Microsoft.Extensions.DependencyInjection.Abstractions | Versão: 8.0.2

Nome: Npgsql | Versão: 9.0.3

Nome: StackExchange.Redis | Versão: 2.8.31

Projeto: Api.Saldos

Nome: IdentityModel.AspNetCore.OAuth2Introspection | Versão: 6.2.0

Nome: Microsoft.AspNetCore.OpenApi | Versão: 8.0.3

Nome: Serilog.AspNetCore | Versão: 9.0.0

Nome: Serilog.Settings.Configuration | Versão: 9.0.1-dev-02317

Nome: Serilog.Sinks.Console | Versão: 6.0.1-dev-00953

Nome: Swashbuckle.AspNetCore | Versão: 6.4.0

Nome: Microsoft.AspNetCore.Authentication.JwtBearer | Versão: 8.0.13

Projeto: Api.Saldos.Testes

Nome: coverlet.collector | Versão: 6.0.4

Nome: coverlet.msbuild | Versão: 6.0.4
Nome: Microsoft.NET.Test.Sdk | Versão: 17.8.0
Nome: Moq | Versão: 4.20.72
Nome: Shouldly | Versão: 4.3.0
Nome: xunit | Versão: 2.5.3
Nome: xunit.runner.visualstudio | Versão: 2.5.3
Nome: Microsoft.AspNetCore.Mvc.Testing | Versão: 8.0.13

Projeto: Artefatos.Notificacoes

Nome: Microsoft.Extensions.DependencyInjection.Abstractions | Versão: 8.0.2

Projeto: Artefatos.Identidade

Nome: Microsoft.AspNetCore.Authentication.JwtBearer | Versão: 8.0.13
Nome: Serilog | Versão: 4.2.0