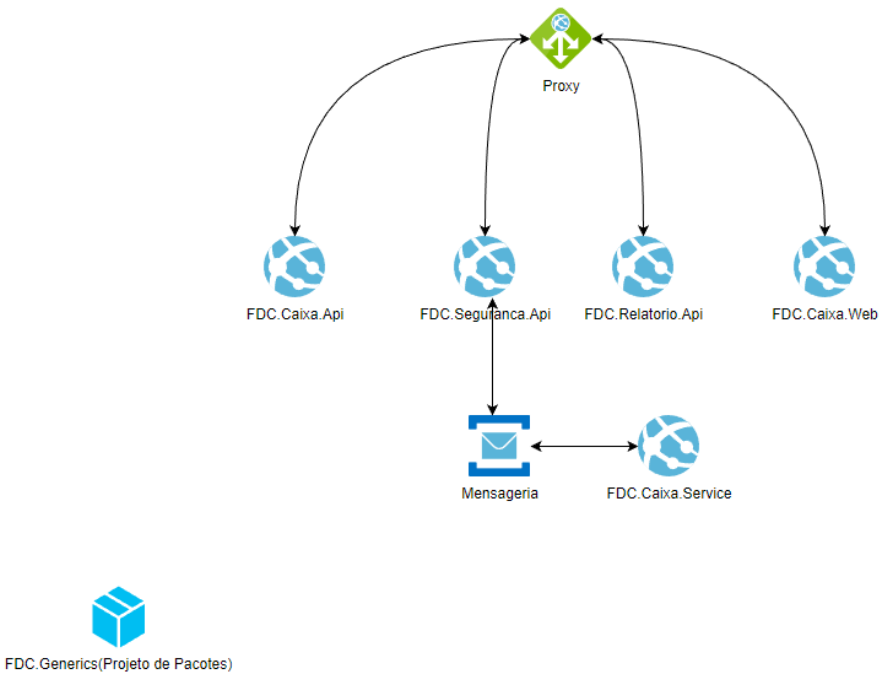
Não foi realizado a hospedagem nessa versão, para uma segunda pensei no Azure Cloud utilizando web apps para aproveitar seu recurso de autoscalling, ou uma solução com docker com kubernetes. Deixei também na raiz do projeto o arquivo docker-compose.yml para subir a aplicação em containers.

Juntamente subi nesse mesmo repositório os mdf de um banco com casos de teste.

Separei em vários micro serviços por serem mais flexíveis, escaláveis e a possibilidade de usar diversas tecnologias, além da facil manutenção pelo time de desenvolvimento.



Proxy NGINX

Coloquei um proxy reverso utilizando NGINX para somente esse servidor ter ips públicos e todo o trafego passar por ele, poderia ter utilizado outros gateway exemplos: azure gateway, frontdoor.

Projetos

FDC.Caixa.** - projetos voltados para o desenvolvimento da regra de negocio para o fluxo de caixa.

FDC.Generics - projeto de pacotes.

FDC.Relatorio - projeto utilizado para emissão de relatórios.

FDC.Seguranca - projeto para realizar autenticação e autorização.

FDC.Caixa.Service – projeto consumidor das mensagens enviadas pelo produtor FDC.Seguranca.

FDC.Caixa.Web (não criado) - projeto que contem a interface para o usuário.

FDC.Caixa.Mensageria(não criado) – projeto em que ficaria guardado todos os logs de mensagens do sistema

FDC.Tests(não finalizado) – projeto para testes de integração, serviço e entidades.

Observações

O projeto **FDC.Caixa.Infra.Query** não foi finalizado porém o objetivo dele seria ter todas consultas utilizarem um micro-orm(exemplo dapper) assim deixando o Entity-Framework apenas para modificações, inserções e exclusões para termos um controle maior do que está sendo executado via banco.

Nessa primeira versão o projeto **FDC.Relatorio** está gerando apenas relatórios simples em XLSX, pensei para uma segunda versão este projeto ser apenas uma faixada para se comunicar com um solução para geração de relatórios, exemplo: **Jasper Reports**

No momento os projetos estão dentro da mesma solução, pensei para uma segunda versão cada um ter seu próprio repositório e solução para não sobrecarregar a atual com uma grande quantidade de arquivos e acabar atrasando o carregamento do projeto.

O projeto **FDC.Generics** pensei em transforma-lo em pacotes e subir ele em um gerenciador, exemplo: **Azure Artifacts**

Não foi utilizado nenhuma solução para cache(redis.memcached), mais possivelmente iria utilizar para consultas de baixa modificação.

Para mensageria nessa versão foi utilizada uma comunicação direta entre os projetos **FDC.Seguranca** e **FDC.Caixa** mas gostaria de ter utilizado o projeto **FDC.Caixa.Mensageria(gateway)** entre eles para guardar logs das mensagens que executaram com sucesso, data, hora e possíveis informações que facilitariam analise e correções.

O banco foi criado em um container, mais recomendo a criação do banco usando um provider de alguma nuvem onde este fara o controle de backups, processamento, memoria e redundância.

Para executar a aplicação via Docker, execute o comando docker-compose up -d na raiz do projeto.

Urls

Autenticação

http://localhost/segurancaserver/swagger/index.html

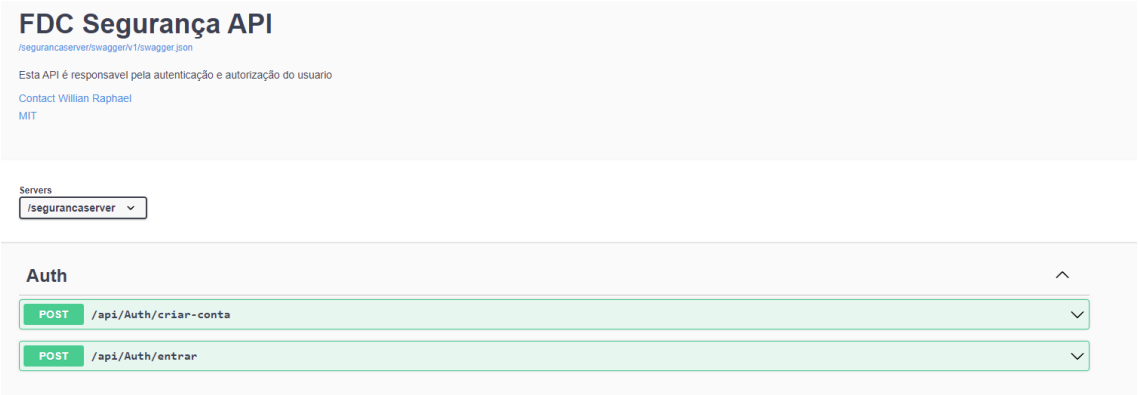
Fluxo de Caixa (Abrir, fechar, Obter e Imprimir)

http://localhost/caixaserver/swagger/index.html

Relatórios

http://localhost/relatorioserver/swagger/index.html

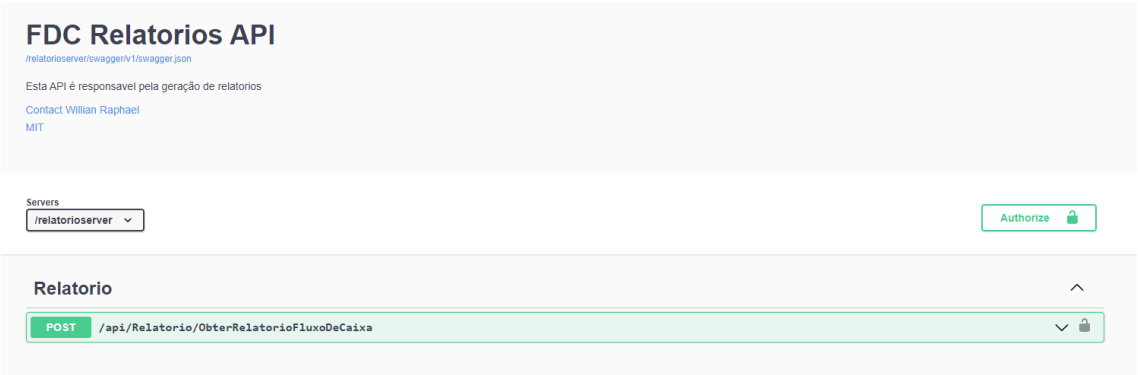
Swagger Segurança



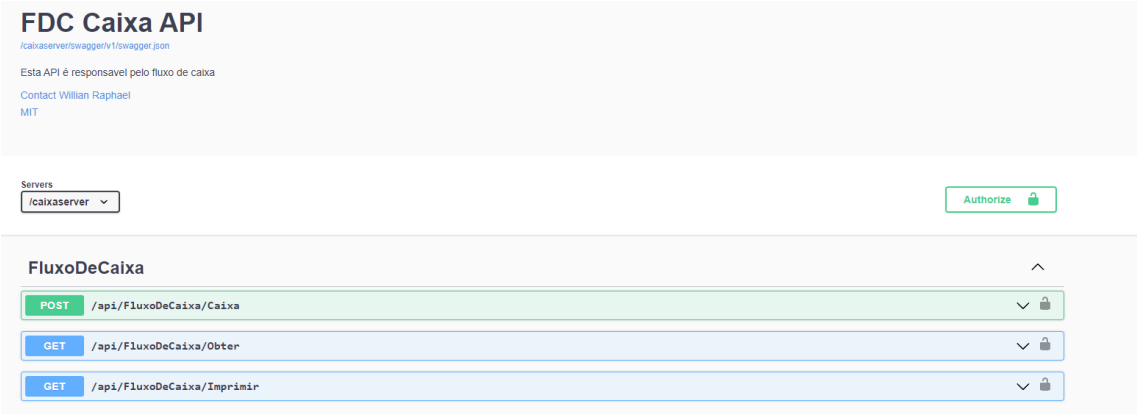
Criar-conta – método para se cadastrar na aplicação.

Entrar – método para realizar login na aplicação.

Swagger Relatório



ObterRelatorioFluxoDeCaixa – Método para gerar relatório do fluxo de caixa, necessário estar autenticado.



FluxoDeCaixa/Caixa – Método que criar um novo fluxo de caixa e tbm o fecha o caixa, necessário estar autenticado.

Parâmetros:

{

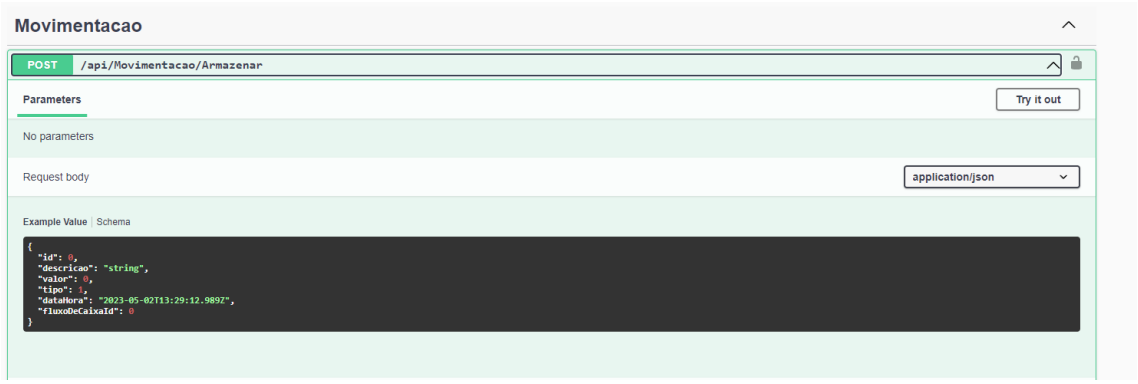
"id": 0, (Zero para criação e maior que zero para edição)

"situacao": 1 (`Aberto = 1 e Fechado = 2`)

}

FluxoDeCaixa/Obter – Método para obter o fluxo de caixa junto de suas movimentações. Recebe o id de um fluxo existente, necessário estar autenticado.

FluxoDeCaixa/Imprimir – Método responsável para imprimir o fluxo de caixa. Recebe o id de um fluxo existente, necessário estar autenticado.



Método responsável para cadastrar ou editar as movimentações, necessário estar autenticado.

{

"id": 0, (Zero para criação e maior que zero para edição)

"descricao": "string", (Descrição sobre o lançamento)

"valor": 0, (Valor do lançamento)

"tipo": 1, (`Debito = 1 e Credito = 2`)

"fluxoDeCaixaId": 0 (Id do fluxo de caixa que será lançada a movimentação)

}