UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CÂMPUS DE ILHA SOLTEIRA

RAFAEL PURCINI STORONE

DE MONÓLITOS A MICROSSERVIÇOS: UM ESTUDO DE CASO DA MATURAÇÃO DE UMA EMPRESA E SUA TRANSIÇÃO ARQUITETURAL

Ilha Solteira
2023

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CÂMPUS DE ILHA SOLTEIRA

RAFAEL PURCINI STORONE

DE MONÓLITOS A MICROSSERVIÇOS: UM ESTUDO DE CASO DA MATURAÇÃO DE UMA EMPRESA E SUA TRANSIÇÃO ARQUITETURAL

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira – Unesp como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof Dr. Carlos Antônio Alves

Ilha Solteira
2023

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S885d

Storone, Rafael Purcini.
De monólitos a microsserviços: um estudo de caso da maturação de uma empresa e sua transição arquitetural / Rafael Purcini Storone. -- Ilha Solteira: [s.n.], 2023
49 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Carlos Antônio Alves

Inclui bibliografia

1. Desenvolvimento de sistemas. 2. Programação. 3. Computação em nuvem. 4. Tecnologia.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Bibliotecas e Documentação
CRB/8 - 9999

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos quinze dias do mês de junho do ano de dois mil e vinte e três, o discente ***Rafael Purcini Storone,*** matriculado sob o nº 162053991, tendo como banca examinadora o seu orientador, *o Prof. Dr. Carlos Antonio Alves, o Prof. Dr. Rodrigo Cardim e o Doutorando Lucas do Carmo Yamaguti,* apresentou o Trabalho de Graduação intitulado **"De Monólitos à microsserviçoes: Um Estudo de Caso da Maturação de Uma Empresa e Sua Transição Arquitetural",** obtendo a nota 9,5 (nove e meio) e conceito APROVADO.

Prof. Dr. Carlos Antonio Alves
- Orientador-

Rafael Purcini Storone
- Discente -

Prof. Dr. *Rodrigo Cardim*
- Membro da Banca -

Doutorando Lucas do Carmo Yamaguti
- Membro da Banca -

DEDICATÓRIA

Dedico este trabalho a todas as pessoas que, de alguma forma, contribuíram para meu crescimento pessoal e profissional, em especial meus pais, minha irmã, minha namorada e meus colegas de trabalho.

RESUMO

O objetivo deste trabalho é comparar duas arquiteturas comuns na engenharia de sistemas, o monólito e os microsserviços, sob uma perspectiva de custos e para um fluxo de informações financeiras de entrada e saída (recebimento, faturamento, agregação de dados e pagamento ou cobrança) utilizando a computação em nuvem (a cada dia mais comum nas companhias) ao invés de servidores locais, o que possibilita uma simulação de custos de componentes a serem usados antes de efetivamente colocar em prática. A motivação para este trabalho é realizar uma análise mais profunda sobre a tomada de decisão no ambiente corporativo que, em boa parte das vezes, leva em consideração os custos entre as arquiteturas monolítica e de microsserviços para contribuir nessa escolha entre as duas e fornecer informações confiáveis. Para isso, utilizou-se as calculadoras disponibilizadas pela *Amazon Web Services,* escolhendo as máquinas com base no processamento desejado, assim como o armazenamento de dados. Foram definidos critérios básicos para o projeto, o que influenciou na escolha de tais componentes, possibilitando a simulação e o cálculo dos custos e a comparação entre os resultados finais das duas arquiteturas que são analisadas neste trabalho. Além deste, outros fatores devem ser levados em conta na hora de escolher entre as duas, como a maturidade dos times, resiliência e escalabilidade dos serviços.

Palavras-chave: desenvolvimento de sistemas; programação; computação em nuvem; tecnologia

ABSTRACT

This paper aims to compare two very common architectures when it comes to software engineering, the monoliths and the microsservices, looking specifically at the costs in a financial flow (receiving, billing, aggregation and payment or charging) using cloud computing (every day more common in companies) instead of local servers. It allows a simulation of costs of components that will be used before effectively using them. The motivation of this paper is a more deep analysis about the decision making in companies that most of the time consider the costs between these two architectures and, to help this choice, provide reliable information. For it, Amazon Web Services calculators were used, choosing machines based on the required processing parameters and also data storage. Some criteria have been defined for this project, helping the choice of components and allowing the simulation and the costs analysis of the two architectures presented in this paper. But other criteria must be considered when choosing between those two architectures like team maturity, resilience and scalability of the services.

Keywords: software development; programming; cloud computing; technology

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

API	Interface de Programação de Aplicação
CNPJ	Cadastro Nacional de Pessoa Jurídica
DDD	*Design* Orientado ao Domínio
E2E	*End-to-end*
IOPS	Entradas e Saídas por Segundo

SUMÁRIO

1. INTRODUÇÃO

A engenharia de sistemas vem evoluindo rapidamente ao longo dos anos devido à evolução tecnológica que está acontecendo no mundo, como o aumento na velocidade dos processadores e o acesso à computação em nuvem. Isso tornou possível a evolução dos processos empresariais com o processamento de um grande volume de dados e um crescimento exponencial das empresas do setor de tecnologia, além do acesso a melhores equipamentos de saúde e o avanço na idealização de carros autônomos. O aumento no acesso à *internet* tem conectado pessoas de todas as partes do mundo, revolucionando a comunicação, a disseminação de informação, as culturas, o comércio eletrônico, as relações de trabalho e muitos outros pontos da sociedade.

A tecnologia da informação está por trás dessas evoluções que vemos no mundo e refere-se aos recursos, técnicas e dispositivos utilizados para adquirir, processar e armazenar informações de maneira digital e é isso que permite a automação dos processos e o avanço nas capacidades de processamento, armazenamento e análise de dados.. Quando analisa-se sua estrutura, pode-se separá-la em quatro partes principais: os sistemas de informação, a arquitetura, a infraestrutura e a gestão, sendo a arquitetura dividida em arquitetura da aplicação e arquitetura da infraestrutura (VERAS, 2012).

Os sistemas da informação dão vida às regras de negócio e as informações são geradas e transitam por eles, garantindo que o fluxo ocorra e que as entradas sejam processadas gerando as suas respectivas saídas. A infraestrutura é o alicerce que vai conseguir manter de pé esses sistemas, sendo capaz de prover os recursos necessários para que o processamento e armazenamento de informação aconteçam. As arquiteturas da aplicação e da infraestrutura são a idealização de como cada etapa se relaciona quando colocada em prática. A parte da aplicação é como o sistema será desenhado para que atenda às necessidades do negócio, ou seja, como suas estruturas internas se relacionam. Já a arquitetura da infraestrutura está relacionada a como os recursos serão mantidos e providos a esses sistemas para que o funcionamento aconteça da maneira ideal. A gestão é a governança por trás de todo esse processo, garantindo que as entregas aconteçam dentro do prazo

e com a qualidade exigida pelo projeto, além de agregar o valor esperado a todas as partes envolvidas.

A evolução tecnológica tem influência direta na maneira como essas arquiteturas são pensadas e desenhadas. Com o avanço na capacidade de processamento e no aumento do número de dados processados pelas empresas, a arquitetura escolhida para o desenvolvimento dos seus serviços precisou evoluir e outros desafios começaram a aparecer, como a manutenção e a confiabilidade de um sistema a longo prazo. Também aumentam os desafios relacionados ao gerenciamento de equipes no ambiente de tecnologia, pois deve-se manter a velocidade nas entregas lidando com tecnologias mais complexas e quantidade de informações cada vez maiores.

Para lidar com isso, algumas estratégias podem ser definidas, como a escrita de um código limpo com definição clara do nome de variáveis, respeitando princípios de boas práticas na construção de classes para que o acoplamento não apareça como um problema no sistema e que a mudança no presente não comprometa a saúde do sistema no longo prazo, trazendo lentidão ao processo de melhoria no futuro (MARTIN, 2009). Pode-se, também, basear-se em metodologias ágeis para o gerenciamento de equipes, o que possibilita que os times continuem entregando melhorias e projetos em prazos exigidas pelo mercado, mas com qualidade e alta entrega de valor a todas as partes envolvidas.

A escolha da arquitetura deve ser pautada em vários fatores como custo e o impacto financeiro no faturamento da empresa, maturidade dos processos da companhia, maturidade técnica das equipes, necessidade de escalar ou não partes independentes do fluxo, a capacidade do fluxo se manter no longo prazo e manter sua funcionalidade mesmo com a evolução dos processos da empresa e diversos outros pontos que vão ser analisados neste trabalho, mostrando as principais vantagens e desvantagens das mais comuns.

Além disso, este trabalho tem como objetivo simular os custos de duas arquiteturas muito comuns no mercado utilizando a calculadora dos serviços em nuvem da *Amazon*, a *Amazon Web Services.* Os custos analisados serão os específicos entre as arquiteturas, como custo de armazenamento e processamento dos dados e também da comunicação assíncrona entre os serviços. Não faz parte

deste trabalho a análise de custos referentes ao complemento de ambas arquiteturas, como chamadas externas e transferência de dados.

2. REVISÃO DE LITERATURA

Monólitos e microsserviços são duas arquiteturas muito comuns na engenharia de sistemas e a escolha entre elas depende de diversos fatores. Monólitos são estruturalmente mais simples, enquanto os microsserviços demandam esquemas mais complexos dada a necessidade de comunicação entre os serviços. O objetivo deste capítulo é aprofundar a teoria dessas arquiteturas e entender como elas funcionam.

2.1. MONÓLITOS

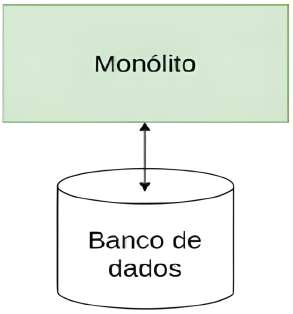
Os monólitos são sistemas com uma arquitetura desenhada para que as regras de negócio estejam contidas em um mesmo sistema e, com isso, haja apenas uma unidade de *deploy* (implantação, colocar no ar, subir a aplicação). Ou seja, quando há a necessidade de subir uma alteração, deve-se implantar a aplicação inteira pois todas as regras estão concentradas nela (NEWMAN, 2022).

Nesta seção serão detalhados os três principais tipos de monólitos, suas divisões estruturais e as vantagens e desvantagens de cada um.

2.1.1. MONÓLITOS DE PROCESSO ÚNICO

Os monólitos de processo único concentram as regras de negócio em apenas um sistema e também possuem um único banco de dados como visto na Figura 1.

Figura 1 - Monólito de processo único.



Fonte: próprio autor

Essa arquitetura é a mais simples, possibilitando a visualização completa do fluxo em um mesmo projeto. Porém, isso pode exigir uma alta capacidade de processamento e armazenamento dependendo da quantidade de informações que esse sistema precisa lidar, pois todo o processamento será realizado por uma única unidade de *deploy*. Além disso, essa concentração das regras de negócio em apenas um sistema faz com que todas as pessoas envolvidas precisem alterar o mesmo projeto caso haja a necessidade de alterações e isso pode dificultar o processo de teste e de *deploy* do projeto, pois muitas pessoas estarão concorrendo nesse processo. Essa problemática é comum em todos os monólitos, mas torna-se um pouco mais difícil de ser contornada nos casos de monólitos de processo único, pois, como é detalhado nas Seções 2.1.2 e 2.1.3, os monólitos modulares e monólitos modulares com segregação de banco de dados possuem a divisão interna do projeto em módulos (facilitando a organização e a alteração do código já existente), o que não ocorre nos monólitos de processo único.

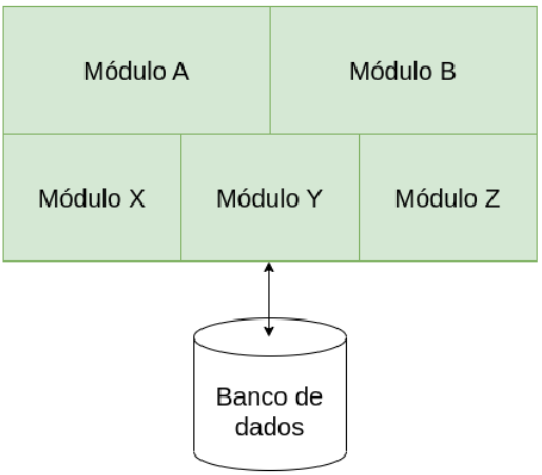
Uma boa organização no código facilita os testes do mesmo, o que garante uma evolução saudável do sistema, que tende a durar por mais tempo (MARTIN, 2009).

2.1.2. MONÓLITOS MODULARES

Os monólitos modulares partem da aplicação do DDD (*design* orientado ao domínio) ao contexto monolítico. O DDD prega que os contextos dos times devem ser abstraídos para o contexto da aplicação. Sendo muito usado como base de microsserviços, o DDD também pode ser aplicado para modularizar um monólito. Ou seja, ao invés de existir uma entidade como em um monólito de processo único, pode-se ter vários módulos que correspondem, cada um, a um contexto diferente da aplicação (EVANS, 2016).

Pode-se observar o exemplo na Figura 2.

Figura 2 - Monólito modular.



Fonte: próprio autor

Essa arquitetura é mais bem dividida em comparação aos monólitos de processo único, o que facilita a manutenção do projeto, pois os arquivos tendem a ficar separados em domínios. Isso permite que haja uma maior facilidade de compreender a estrutura do projeto por conta dessa melhor organização e, assim, uma alteração no código tende a ter um impacto mais controlado nessa comparação.

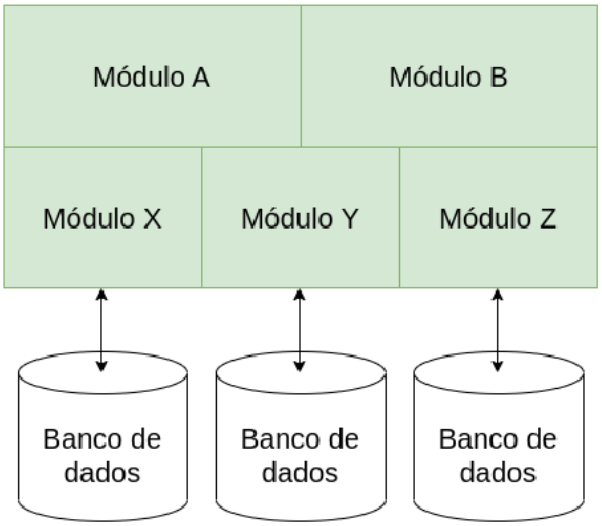
2.1.3. MONÓLITOS MODULARES COM SEGREGAÇÃO DE BANCO DE DADOS

Os monólitos modulares com segregação de banco de dados apresentam o conceito de DDD (*design* orientado ao domínio) aplicado também aos bancos de dados da aplicação, isolando-os para cada módulo específico. Desta forma, o sistema possui contextos divididos em módulos, mas ainda é um monólito porque esses módulos estão contidos na mesma aplicação. Assim, ainda usufrui dos benefícios de um monólito (um único *deploy* para subir alterações, simplicidade na observabilidade, times mais simples e comunicação interna no próprio sistema) (NEWMAN, 2022).

Essa segregação de banco de dados apresenta uma vantagem quando uma grande quantidade de dados está sendo processada ou quando muitas consultas estão sendo realizadas. Como os bancos de dados estão separados, não há uma sobrecarga no mesmo quando esse processamento está ocorrendo, pois contextos diferentes estão separados e as consultas podem acontecer sem concorrência, aumentando a eficiência do banco.

Além disso, construindo um monólito neste modelo, a evolução para os microsserviços, quando isso for necessário por decisões de negócios, é facilitada. Nesse modelo devemos nos atentar para a consistência de dados, visto que várias tabelas são construídas para um domínio parecido e uma mudança em um precisaria refletir em todos os registros, por exemplo a alteração do nome de um cliente. Observa-se a representação deste sistema na Figura 3.

Figura 3 - Monólito modular com segregação de banco de dados.



Fonte: próprio autor

Com isso, os monólitos apresentam alguns benefícios para utilização. O fato das regras de negócio estarem concentradas em um único sistema faz com que a organização dos times seja mais fácil (visto que não é necessário separar os times em contextos de sistemas como no caso dos microsserviços) e a complexidade do projeto seja menor, pois não há a necessidade da criação de sistemas

complementares e, consequentemente, não há a necessidade de comunicação entre eles (NEWMAN, 2022).

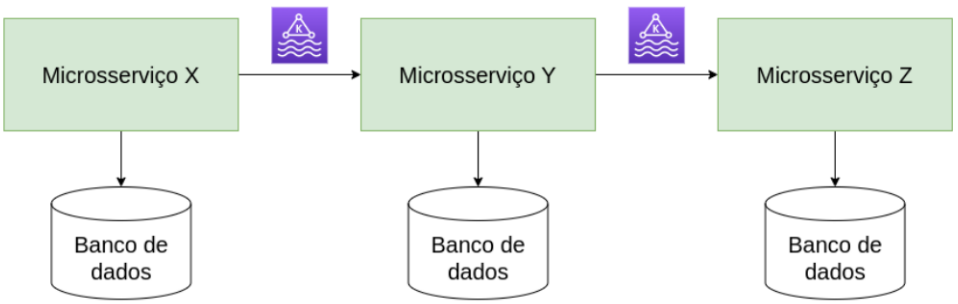
2.2. MICROSSERVIÇOS

Os microsserviços são sistemas independentes, ou seja, com *deploy* independente um do outro, dividindo as regras de negócio em aplicações menores. Essa divisão é feita com base nos domínios do negócio com base em contextos delimitados, diferentemente dos monólitos.

Essa mudança arquitetural traz algumas vantagens, como a possibilidade de escalar partes distintas do processo e também a independência na hora de subir uma alteração para produção, pois, como os contextos estão delimitados, a mudança pode ocorrer apenas naquele contexto. Em contrapartida, essa alteração pode trazer um impacto aos serviços que se comunicam com o que foi alterado, já que uma das práticas da arquitetura de microsserviços é a comunicação entre sistemas, seja de forma síncrona ou assíncrona; e isso faz com que testes E2E (que testam toda a cadeia) sejam mais complexos do que em uma arquitetura monolítica (FOWLER, 2017).

Pode-se observar um exemplo genérico de arquitetura baseada em microsserviços com comunicação assíncrona na Figura 4. A comunicação é representada pelos blocos roxos, que sinalizam o serviço de transmissão de mensagens assíncronas entre os microsserviços.

Figura 4 - Microsserviços com comunicação assíncrona.



Fonte: próprio autor

Dessa maneira, os times são construídos com bases nesses contextos que foram delimitados para a criação desses microsserviços que, por se tratar de

aplicações separadas, não compartilham o mesmo banco de dados. Com isso, o funcionamento de cada serviço é independente dos outros, o que contribui para a resiliência de um sistema. Porém, isso traz uma complexidade a mais ao processo, visto que é necessário pensar em maneiras de como garantir a consistência de dados entre as bases, pois uma alteração que seja feita em apenas talvez não seja propagada às outras (NEWMAN, 2022).

2.3. COMPUTAÇÃO EM NUVEM

A *internet* vem tendo muitas melhorias desde sua invenção, como a padronização de protocolos de comunicação e a redução de custos para interligação, o que fizeram a rede ser um dos principais meios de comunicação do planeta muito rapidamente. Com isso, ao invés do processamento e armazenamento de dados acontecer localmente em um servidor, a ideia de que algo fosse processado e armazenado usando a própria rede e, consequentemente, fora do ambiente corporativo, começou a ganhar forças e, assim, surgiu o conceito de computação em nuvem. Posteriormente, esse conceito foi evoluindo para o que vemos hoje, com o processamento e armazenamento ocorrendo em *data centers* (locais físicos com máquinas capazes de processar e armazenar as informações das empresas).

Uma das principais vantagens da computação em nuvem é a otimização dos recursos com o que é chamado de elasticidade. Como as demandas futuras não podem ser perfeitamente previstas, a necessidade de processamento e armazenamento também não pode ser calculada com precisão. Ora precisa-se de muito processamento e ora precisa-se de pouco a depender da demanda e do comportamento dos sistemas. Com a evolução para a computação em nuvem, é possível escalar e desescalar sistemas conforme a demanda e garantir a melhor eficiência dos recursos, algo que não seria possível com o processamento fixo local. Além disso, novos serviços podem ser feitos sem a necessidade de comprar novas máquinas físicas e locais, pode-se contratá-las dos *data centers* utilizando a nuvem (VERAS, 2012).

Assim, surgiram empresas que oferecem a computação em nuvem como um serviço, construindo *data centers* e disponibilizando-os via nuvem para clientes

contratarem e utilizarem. As mais famosas são a *Amazon Web Services*, a *Google Cloud Platform* e a Microsoft *Azure.* Essas empresas oferecem serviços semelhantes, mas com nomes diferentes e disputam o topo do mercado da computação em nuvem. Para este trabalho, utilizou-se a *Amazon Web Services* na região Norte da Virgínia.

3. DESENVOLVIMENTO

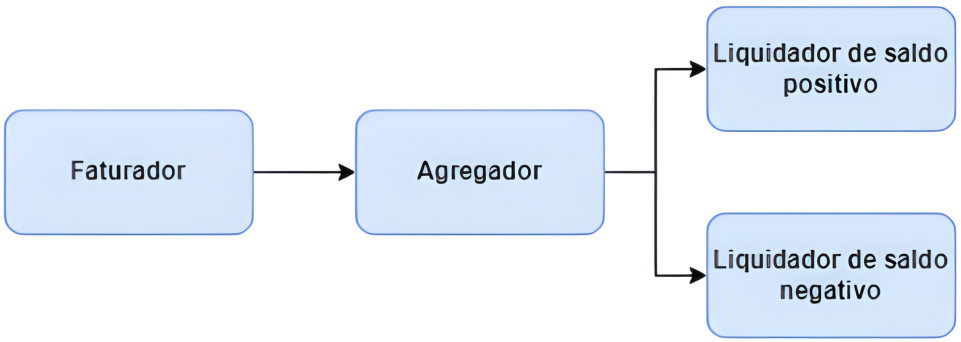
O objetivo deste capítulo é apresentar os requisitos relacionados a este trabalho, o fluxo de informações, as arquiteturas de cada projeto e também as simulações de custos de cada uma das arquiteturas. Os requisitos são fundamentais pois definem quais máquinas devem ser escolhidas para atender aquela demanda e, para isso, *Amazon Web Services* disponibiliza uma calculadora, onde as simulações em questão serão realizadas.

3.1. DEFINIÇÕES DO PROJETO

O monólito escolhido para as simulações foi o de processo único. Para as simulações dos cálculos de custos, considerou-se que ambas as arquiteturas, o monólito de processo único e os microsserviços, processam 50 milhões de entradas cada um por mês.

O fluxo de informações financeiras segue a Figura 5.

Figura 5 - Fluxo de informações do projeto.



Fonte: próprio autor

As informações financeiras chegam ao faturador. Nessa etapa, os cálculos de comissão e outras taxas serão realizados, como por exemplo a taxa de mensalidade para a contratação dos serviços. Após realizado esse processo, os dados são encaminhados ao agregador, que será responsável por agregar as informações com base em uma premissa. Pode ser, por exemplo, por cliente, por período de apuração ou por qualquer outro critério que faça sentido ao fluxo escolhido. Neste caso, as agregações serão realizadas por período. Essas agregações são a soma dos

valores que serão repassados ou cobrados dos clientes. Caso o montante final dessa agregação seja positivo, ou seja, a instituição deve ao cliente, o valor será encaminhado ao liquidador de saldo positivo, que será responsável por fazer o repasse ao cliente por meio de transferências bancárias. Caso o montante final das agregações seja negativo, ou seja, o cliente fica em débito com a instituição, o valor será repassado ao liquidador de saldo negativo, que terá a responsabilidade de realizar a cobrança do cliente por meio de boleto bancário.

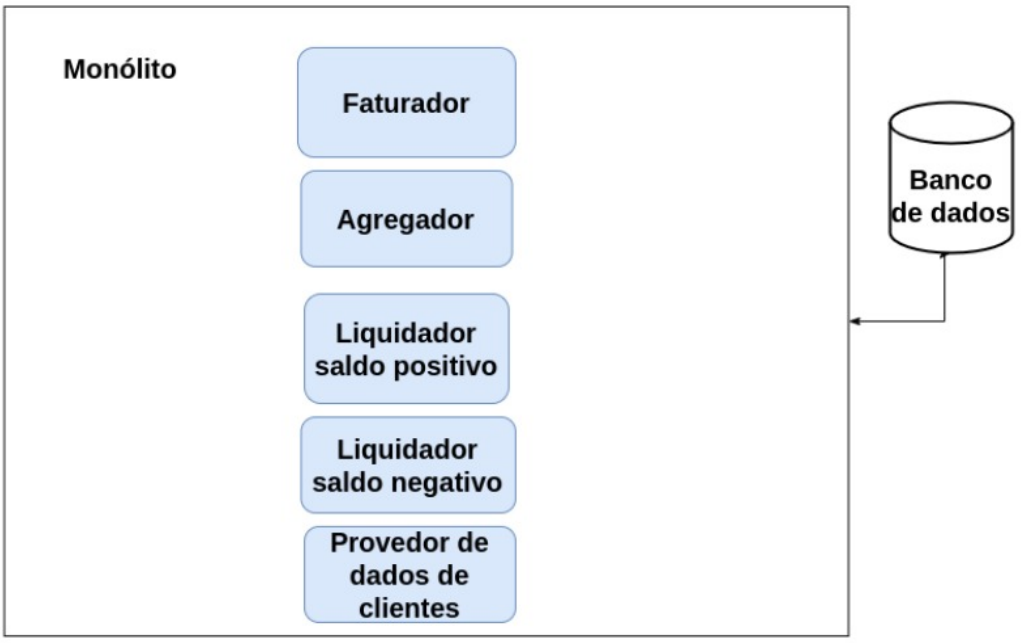
3.2. ARQUITETURA PROPOSTA

A arquitetura do projeto é a representação prática das regras de negócio apresentadas na Figura 5, ou seja, é a maneira como os sistemas serão organizados para realizar as operações esperadas por esse fluxo.

3.2.1. ARQUITETURA MONOLÍTICA

O sistema monolítico de processo único é detalhado na Figura 6 e realiza todas as operações do fluxo financeiro.

Figura 6 - Arquitetura monolítica.



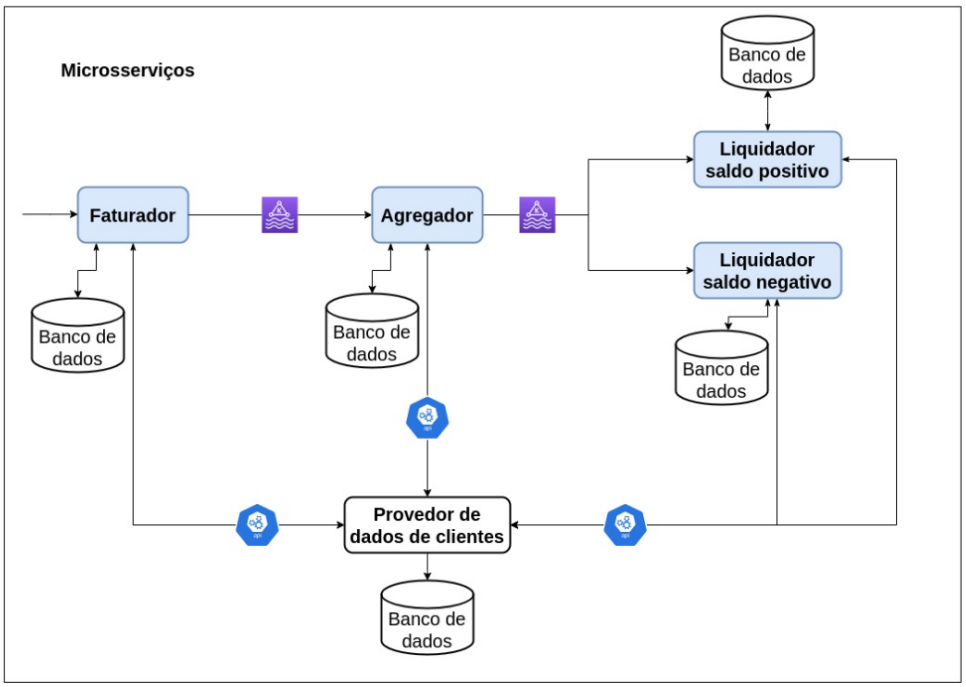
Fonte: próprio autor

Além do que é detalhado na Figura 5, esse sistema também possui um provedor de dados de clientes para que busque as informações corretas dos consumidores, como conta bancária, endereço e CNPJ. Isso tudo está contido na mesma aplicação e, consequentemente, compartilha o mesmo banco de dados, não havendo a necessidade de preocupar-se com a consistência de dados entre as bases, como no cenário de monólitos modulares com segregação de bancos de dados. Porém, deve-se atentar-se à capacidade desse banco de dados lidar com as requisições, pois todas as informações estão contidas nele. Isso é levado em conta na hora de dimensionar a sua capacidade e reflete no custo final para manutenção.

3.2.2. ARQUITETURA DE MICROSSERVIÇOS

A arquitetura da Figura 7 é a esquematização de microsserviços. Vários serviços que contém as regras de negócios são criados de forma distribuída e funcionam de maneira praticamente independente, realizando apenas consultas via APIs (Interface de Programação de Aplicação, do inglês *Application Programming Interface*) ou consumindo eventos via mensageria. Além disso, os bancos de dados são segregados.

Figura 7 - Arquitetura de microsserviços.



Fonte: próprio autor

Com essa organização, os sistemas representam contextos delimitados e, diferentemente dos monólitos, a tendência é que exista uma equipe para cada serviço ou algo próximo disso. Uma equipe poderia ser responsável pelos dois liquidadores, mas é pouco provável que um só time mantenha o fluxo todo. A comunicação assíncrona por meio de mensagens possibilita que quase todos os sistemas trabalhem de forma independente, com exceção do provedor de clientes, que recebe chamadas síncronas. Isso significa que, quando o provedor de dados de clientes estiver fora do ar, isso impactará diretamente o funcionamento dos outros serviços. Porém, quando o faturador estiver fora do ar, o agregador poderá continuar funcionando normalmente devido à comunicação assíncrona, pois o consumo de mensagens não é impactado.

As bases de dados segregadas apresentam o mesmo problema dos monólitos modulares com banco de dados segregados, pois é necessário atenção à consistência de informações entre as bases, visto que, por estarem separadas, uma atualização em um dado de uma das bases não refletirá, necessariamente, em uma outra base de dados. Isso pode causar inconsistências no fluxo de dados que transitam entre os sistemas.

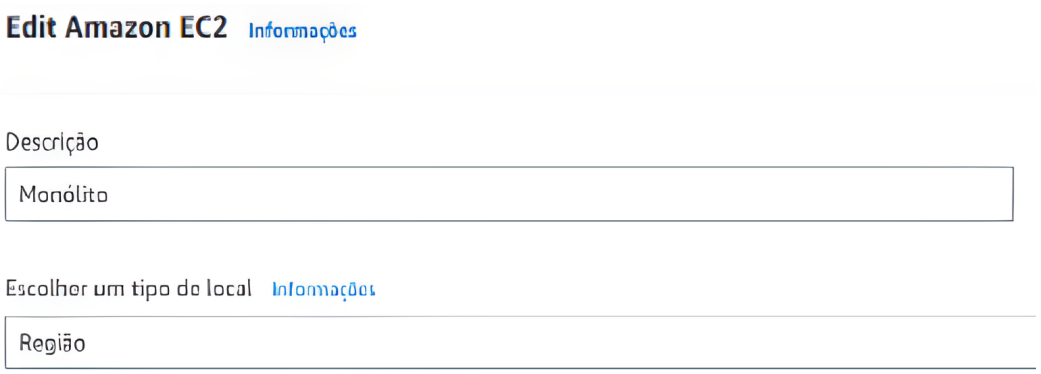
3.3. SIMULAÇÕES

Utilizou-se o site oficial da *Amazon Web Services* para simular os custos dos serviços desenvolvidos em cada arquitetura definida nas Figuras 6 e 7.

3.3.1. MONÓLITO

Para o monólito, considerou-se a capacidade de processamento que uma máquina precisaria para lidar com essa quantidade de informação especificada na Seção 3.1. Na Figura 8, observa-se as configurações iniciais para a máquina, como a descrição e a região escolhida.

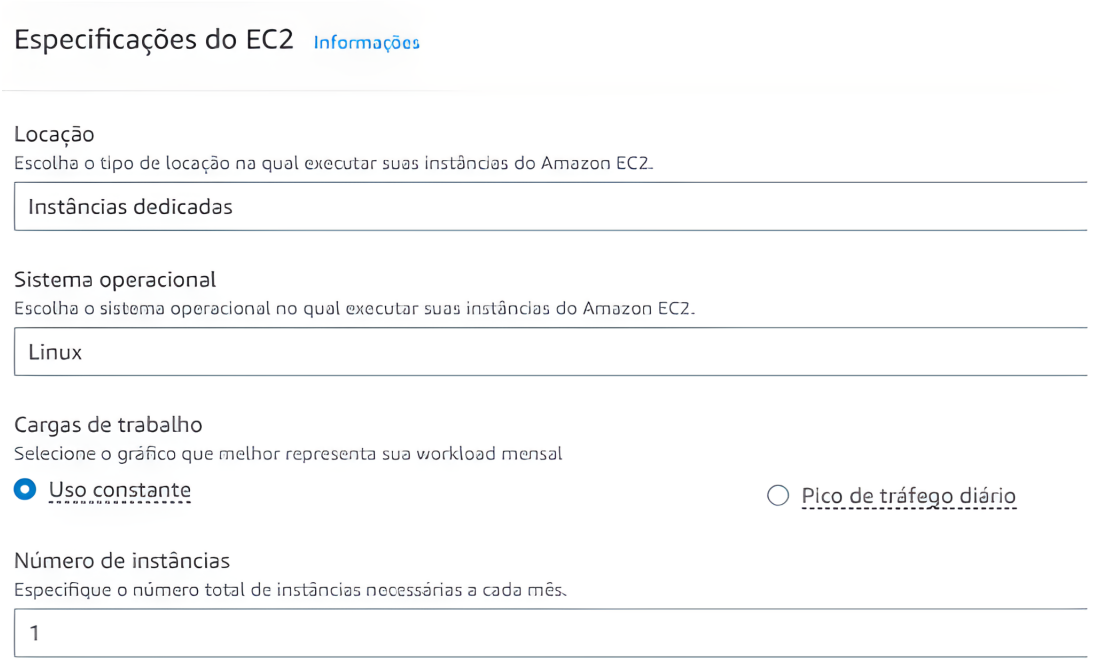
Figura 8 - Configurações básicas do monólito.



Fonte: (AMAZON WEB SERVICES, 2023)

Na Figura 9, evidencia-se as configurações de uso para o monólito, como o tipo de locação, carga de trabalho e o sistema operacional utilizado. Escolheu-se o Linux como sistema operacional devido a sua capacidade de customização, segurança e gratuidade.

Figura 9 - Configurações de uso do monólito.



Fonte: (AMAZON WEB SERVICES, 2023)

Visto que todos os processos estão agrupados em um mesmo sistema, isso exige uma alta capacidade de processamento, o que justificou a escolha da máquina

da Figura 10 para o monólito. Essa instância tem o custo mensal especificado pela Figura 13 (custo do EC2).

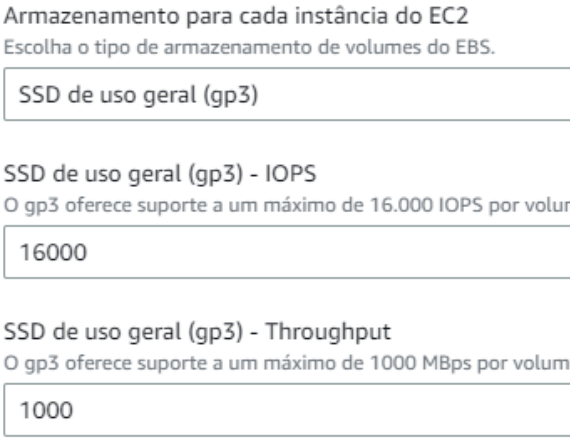
Figura 10 - Configurações de processamento do monólito.

	Instance name	Categoria da instância	vCPUs	Núcleos físicos	Memória
○	inf1.xlarge	Machine Learning ASIC Instances	4		8 GiB
◉	r6gd.xlarge	Memory optimized	4		32 GiB
○	r6a.xlarge	Memory optimized	4		32 GiB
○	c6in.xlarge	Compute optimized	4		8 GiB
○	m5n.xlarge	General purpose	4		16 GiB
○	m6id.xlarge	General purpose	4		16 GiB

Fonte: (AMAZON WEB SERVICES, 2023)

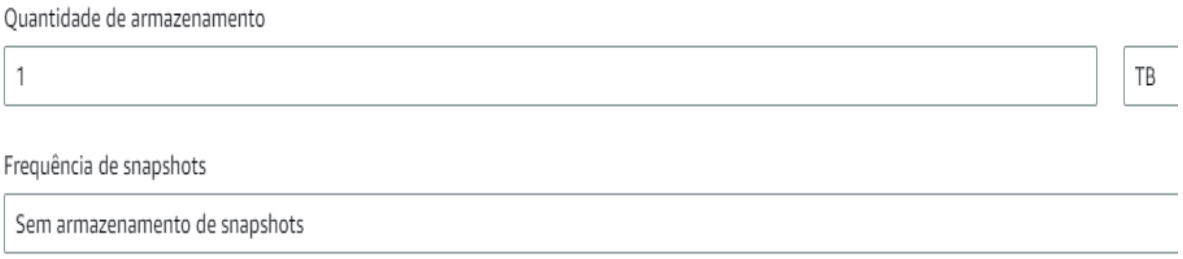
Além disso, considerou-se a capacidade de armazenamento que esse sistema precisaria para lidar com essa quantidade de informação especificada na Seção 3.1, detalhando-a nas Figuras 11 e 12. O custo mensal total é apresentado na Figura 13 (custo de *Elastic Block Store*).

Figura 11 - Configurações gerais de armazenamento do monólito.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 12 - Quantidade de armazenamento do monólito.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 13 - Custo mensal total na arquitetura monolítica.

Fonte: (AMAZON WEB SERVICES, 2023)

Throughput é a medida de *megabytes* por segundos que o disco pode fornecer e IOPS é a quantidade de requisições de entrada e saída por segundo que ele consegue receber. Essas medidas estão relacionadas e, para o caso dos monólitos, dado que o fluxo todo está concentrado em apenas um sistema, considerou-se os valores máximos para essas medidas, pois é esperado que o banco de dados seja muito requisitado e precise trabalhar próximo do limite muitas vezes.

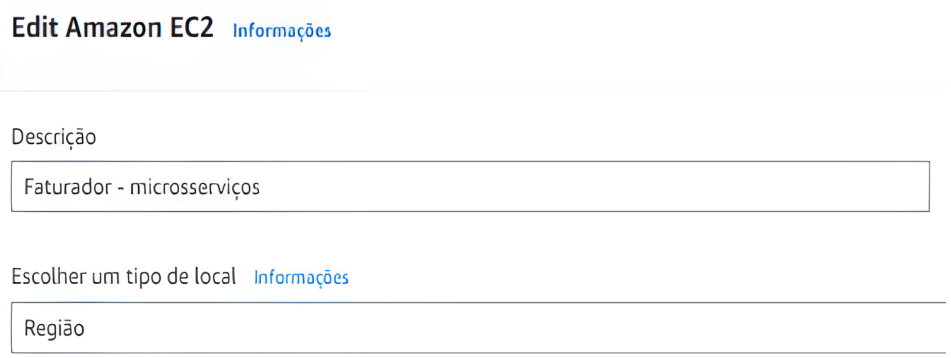
3.3.2. MICROSSERVIÇOS

Para os microsserviços, detalhou-se o custo de cada serviço conforme especificado na arquitetura da Figura 7, onde cada um representa um serviço distinto e com suas próprias máquinas.

3.3.2.1. FATURADOR

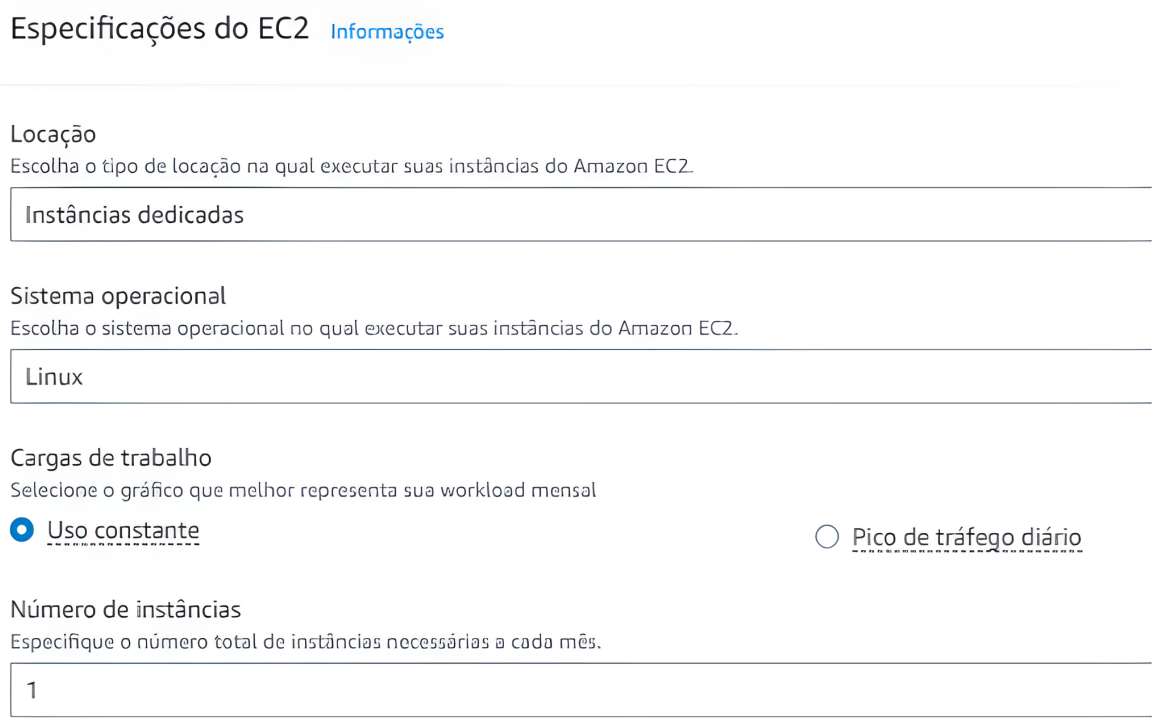
As Figuras 14 a 16 mostram os detalhes da configuração de processamento escolhida para o faturador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 19 (custo do EC2).

Figura 14 - Configurações básicas do faturador.



Fonte: (AMAZON WEB SERVICES, 2023

Figura 15 - Configurações de uso do faturador.



Fonte: (AMAZON WEB SERVICES, 2023)

Diferentemente do monólito, o faturador é responsável por apenas uma parte do processo referente ao fluxo de informações. Com isso, não há a necessidade de uma máquina tão potente quanto a do monólito, visto que a capacidade da máquina está diretamente ligada à quantidade de informações que ela processa. O sistema operacional escolhido também foi o Linux pelos mesmos motivos já descritos na Seção 3.3.1.

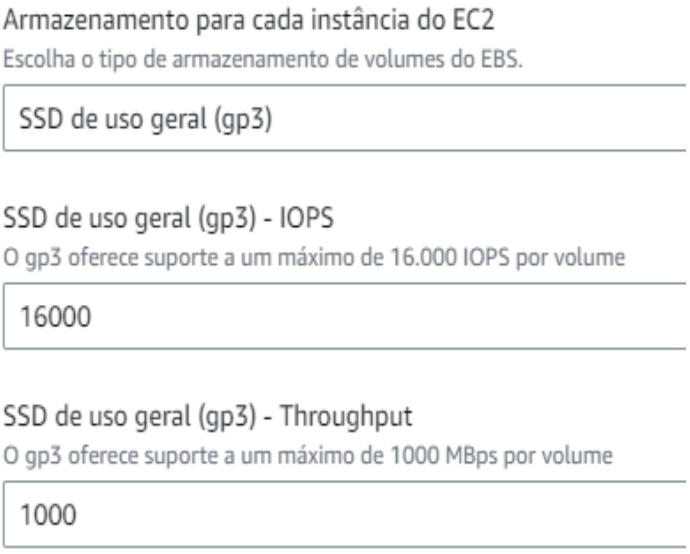
Figura 16 - Configurações de processamento do faturador.

	Instance name	Categoria da instância	vCPUs	Núcleos físicos	Memória
○	t3.nano	General purpose	2		0.5 GiB
○	t3.micro	General purpose	2		1 GiB
○	t3.small	General purpose	2		2 GiB
○	a1.medium	General purpose	1		2 GiB
○	c6g.medium	Compute optimized	1		2 GiB
○	c7g.medium	Compute optimized	1		2 GiB
○	c6gd.medium	Compute optimized	1		2 GiB
○	m6g.medium	General purpose	1		4 GiB
○	m7g.medium	General purpose	1		4 GiB
◉	t3.medium	General purpose	2		4 GiB

Fonte: (AMAZON WEB SERVICES, 2023)

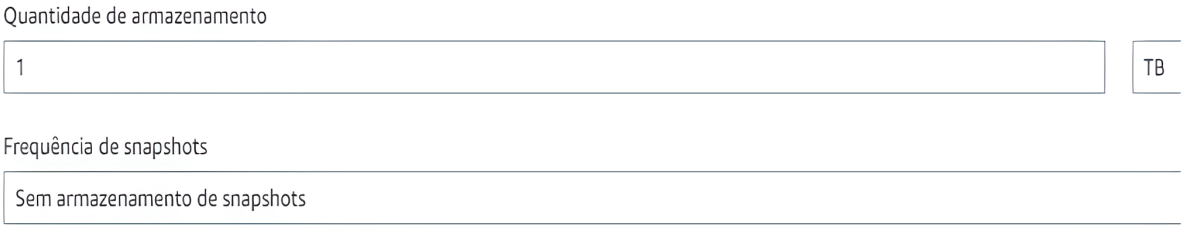
As Figuras 17 e 18 mostram os detalhes da configuração de armazenamento escolhida para o faturador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 19 (custo de *Elastic Block Store*).

Figura 17 - Configurações gerais de armazenamento do faturador.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 18 - Quantidade de armazenamento do faturador.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 19 - Custo mensal total para o faturador.

Fonte: (AMAZON WEB SERVICES, 2023)

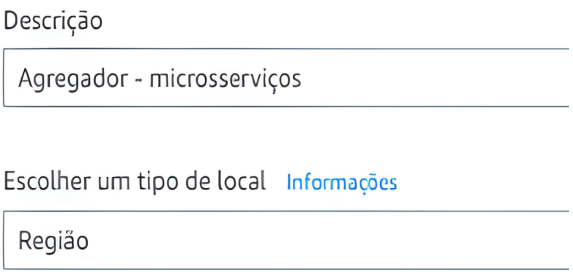
Para o armazenamento, considerou-se os mesmos valores escolhidos para o monólito. A diferença é que, no caso do faturador, essa escolha foi baseada apenas na garantia de que o banco de dados suportaria picos de requisições, sendo esses

menos prováveis e menos recorrentes do que no cenário do monólito, onde espera-se que tenha que lidar com esse problema mais frequentemente dada a concentração das regras de negócio em apenas um sistema.

3.3.2.2. AGREGADOR

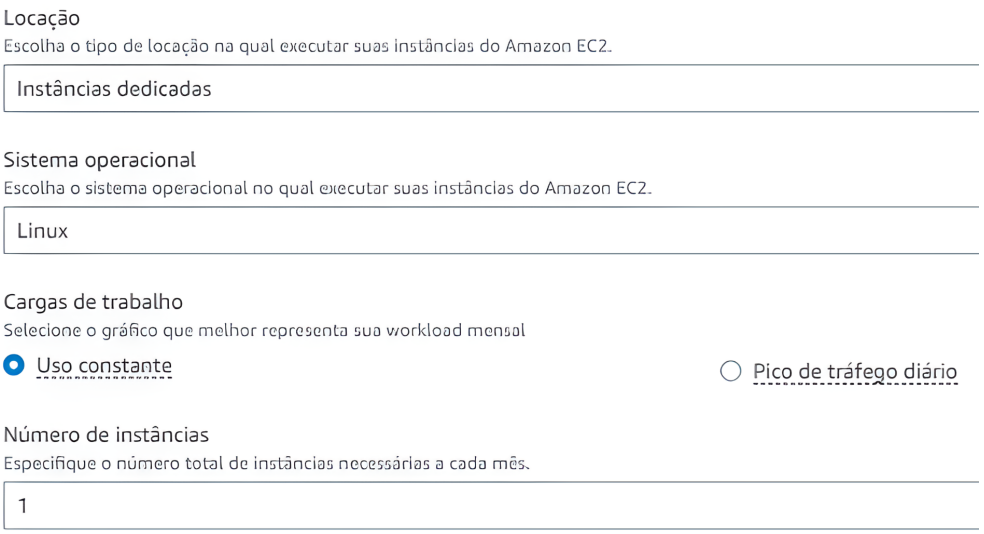
As Figuras 20 a 22 mostram os detalhes da configuração de processamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 25 (custo do EC2).

Figura 20 - Configurações básicas do agregador.



Fonte: (AMAZON WEB SERVICES, 2023)

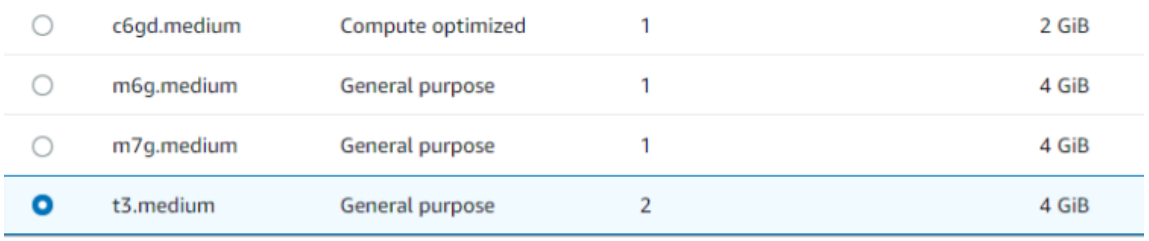
Figura 21 - Configurações de uso do agregador.



Fonte: (AMAZON WEB SERVICES, 2023)

Pode-se considerar que o faturador e o agregador consomem a mesma quantidade de processamento, pois passa por eles quantidades muito parecidas de informação. O faturador, como consta na Seção 3.1, processa 50 milhões de entradas e envia o resultado de todas elas ao agregador. Por isso, a máquina escolhida para o agregador foi a mesma escolhida para o faturador.

Figura 22 - Configurações de processamento do agregador.

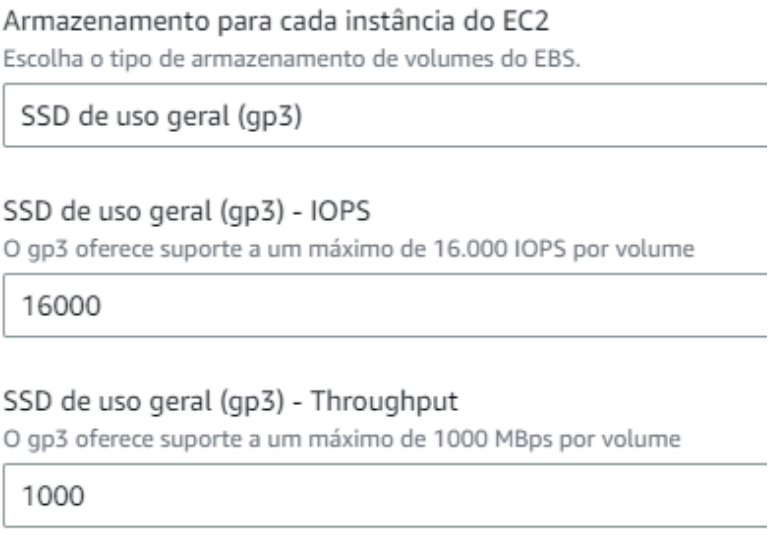


○	c6gd.medium	Compute optimized	1	2 GiB
○	m6g.medium	General purpose	1	4 GiB
○	m7g.medium	General purpose	1	4 GiB
◉	t3.medium	General purpose	2	4 GiB

Fonte: (AMAZON WEB SERVICES, 2023)

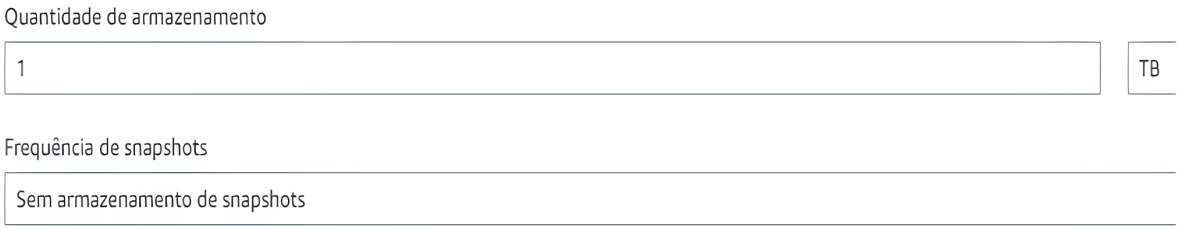
As Figuras 23 e 24 mostram os detalhes da configuração de armazenamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 25 (custo de *Elastic Block Store*).

Figura 23 - Configurações gerais de armazenamento do agregador.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 24 - Quantidade de armazenamento do agregador.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 25 - Custo mensal total para o agregador.

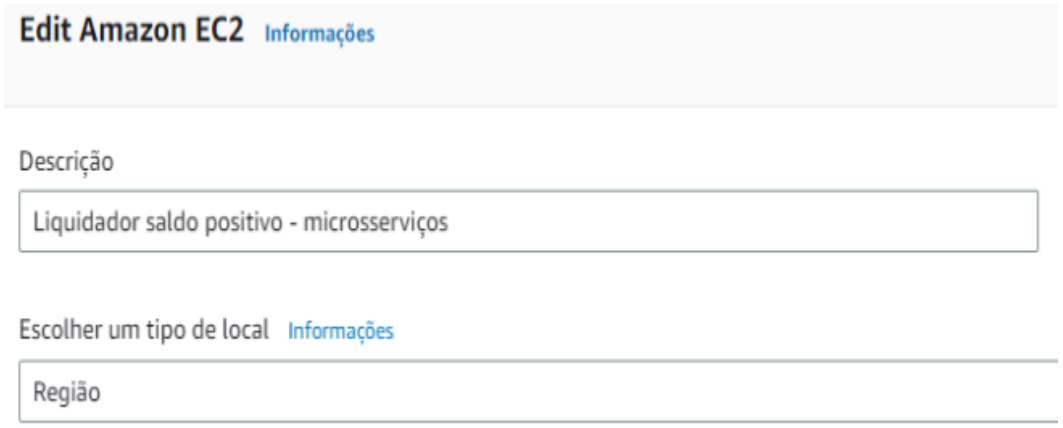
Fonte: (AMAZON WEB SERVICES, 2023)

Escolheu-se o throughput e IOPS máximo para o agregador novamente para que ele consiga lidar com picos de processamento, por exemplo quando houver muitas informações enviadas para ele pelo faturador, pois isso exigirá uma alta capacidade de escrita de seu disco. No caso do agregador, assim como no faturador, esses picos tendem a ser menos frequentes do que no monólito.

3.3.2.3. LIQUIDADOR DE SALDO POSITIVO

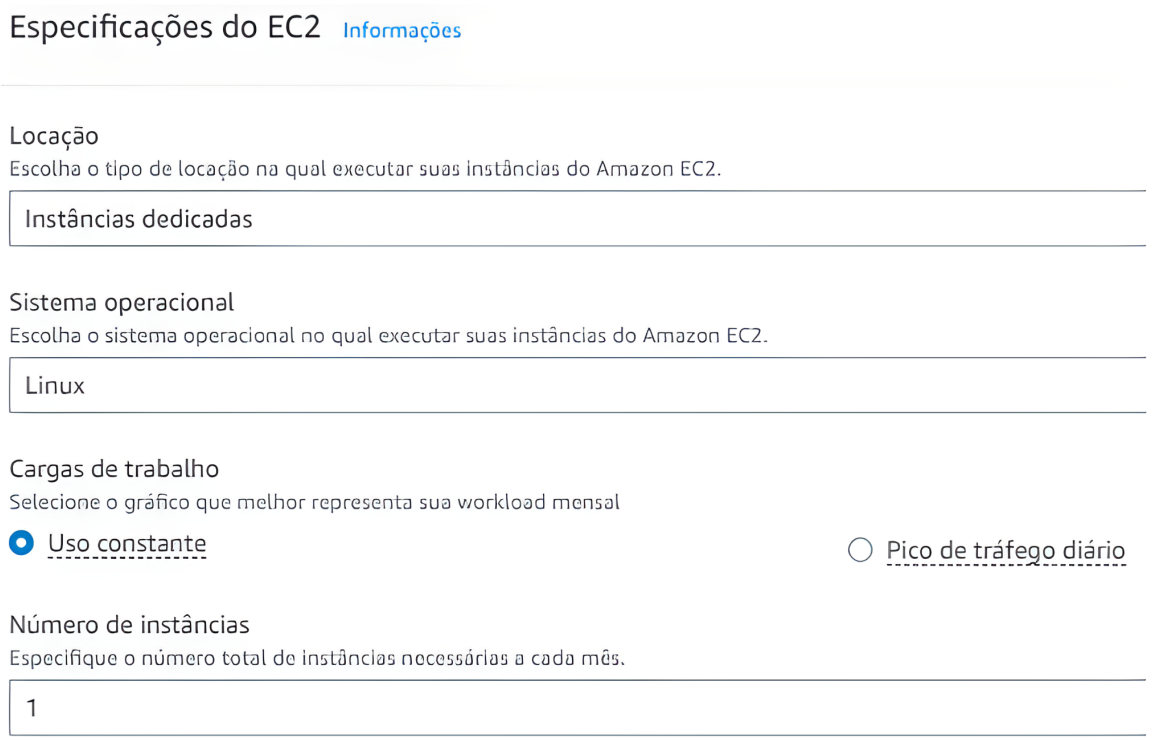
As Figuras 26 a 28 mostram os detalhes da configuração de processamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 31 (custo do EC2).

Figura 26 - Configurações básicas do liquidador de saldo positivo.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 27 - Configurações de uso do liquidador de saldo positivo.



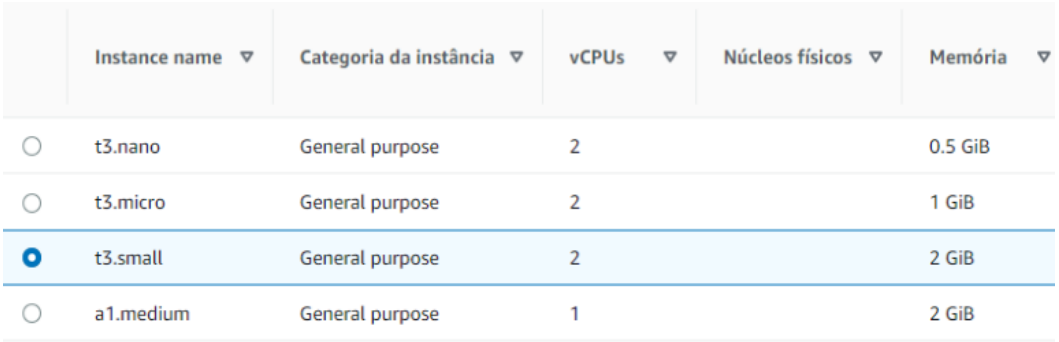
Fonte: (AMAZON WEB SERVICES, 2023)

Diferentemente do faturador e do agregador, o liquidador de saldo positivo não irá processar as 50 milhões de entradas do fluxo financeiro porque a saída do agregador é um valor final para cada cliente que pode ser positivo ou negativo e, consequentemente, tendo a possibilidade de ser enviado para um entre dois sistemas. De forma mais prática, se um cliente tiver 200 pedidos em um mês,

existirão 200 entradas no faturador para esse cliente e 200 no agregador. Porém, existirá apenas uma saída do agregador para esse mesmo parceiro, pois o papel do agregador é justamente o de juntar os saldos para ter um montante final. Além disso, esse valor final pode ser tanto positivo quanto negativo. Caso seja negativo, não será enviado ao liquidador de saldo positivo, mas sim ao liquidador de saldo negativo e vice-versa.

Esses fatores fazem com que a máquina escolhida para o liquidador de saldo positivo seja uma máquina menos potente do que a escolhida para o faturador e para o agregador.

Figura 28 - Configurações de processamento do liquidador de saldo positivo.

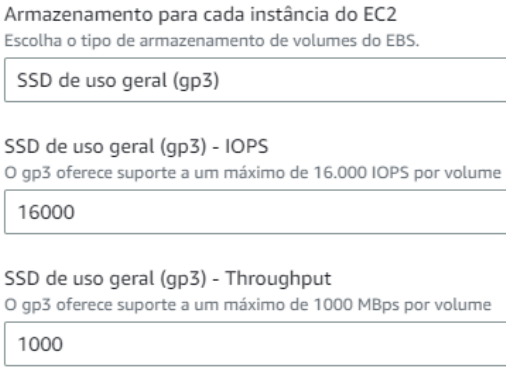


	Instance name	Categoria da instância	vCPUs	Núcleos físicos	Memória
○	t3.nano	General purpose	2		0.5 GiB
○	t3.micro	General purpose	2		1 GiB
◉	t3.small	General purpose	2		2 GiB
○	a1.medium	General purpose	1		2 GiB

Fonte: (AMAZON WEB SERVICES, 2023)

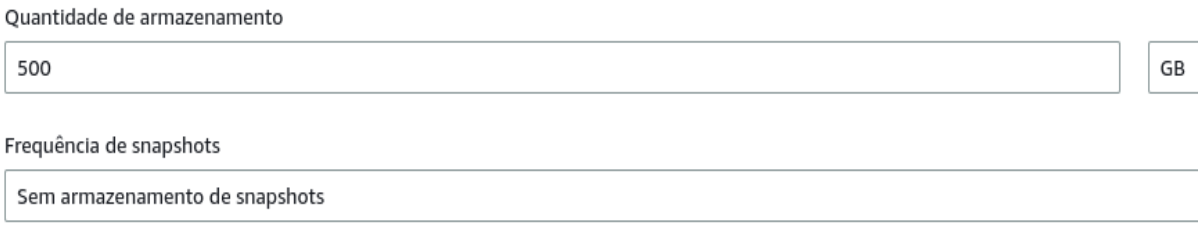
As Figuras 29 e 30 mostram os detalhes da configuração de armazenamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 31 (custo de *Elastic Block Store*).

Figura 29 - Configurações gerais de armazenamento do liquidador de saldo positivo.



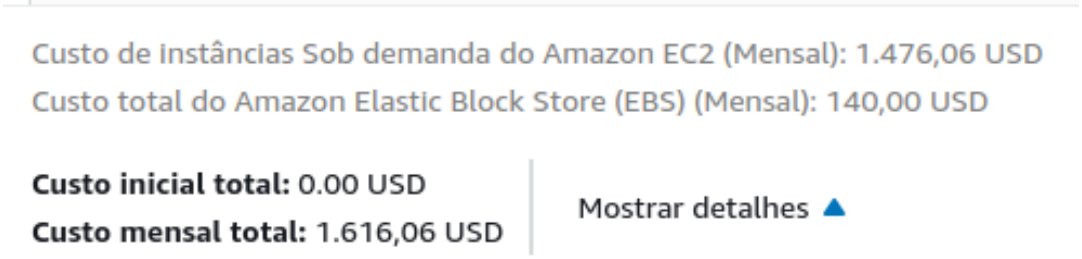
Fonte: (AMAZON WEB SERVICES, 2023)

Figura 30 - Quantidade de armazenamento do liquidador de saldo positivo.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 31 - Custo mensal total para o liquidador de saldo positivo.



Fonte: (AMAZON WEB SERVICES, 2023)

Para o armazenamento, a escolha para o liquidador de saldo positivo é diferente do faturador e do agregador para a capacidade. Escolheu-se a máxima para que o sistema consiga lidar com picos de armazenamento mesmo que a possibilidade de isso acontecer seja bem menor do que no caso dos monólitos. Porém, olhando para a quantidade de armazenamento, não há a necessidade de algo tão grande quanto o faturador e o agregador, pois a quantidade de dados que chega a esse serviço é menor do que a que a que chega aos seus antecessores, de acordo com o que foi especificado na Seção 3.3.2.3.

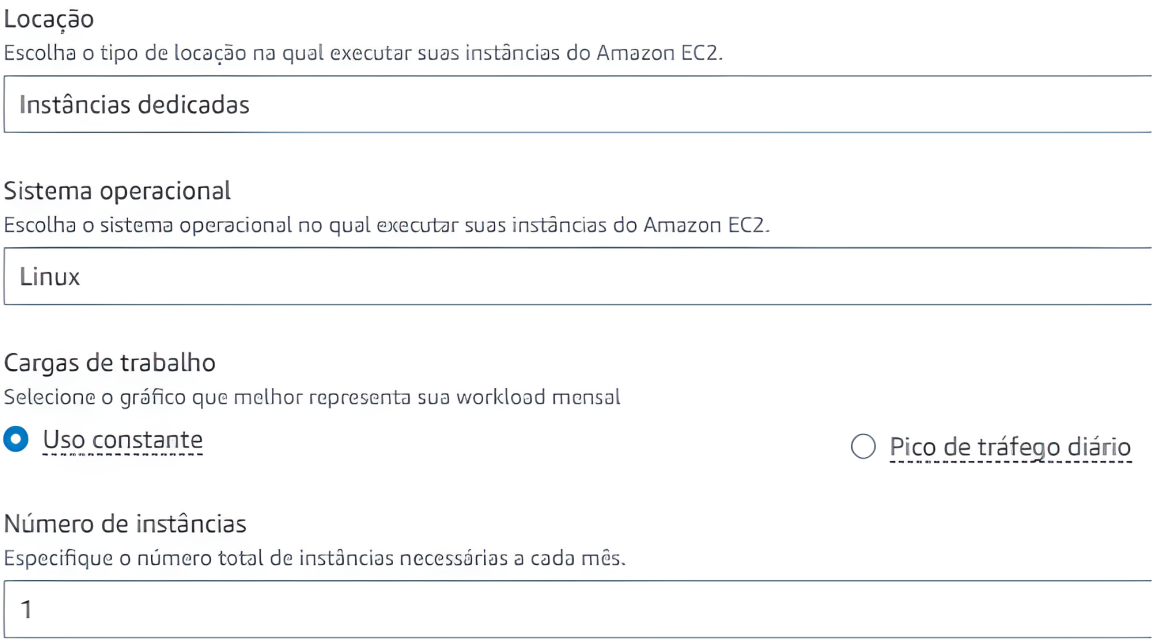
3.3.2.4. LIQUIDADOR DE SALDO NEGATIVO

As Figuras 32 a 34 mostram os detalhes da configuração de processamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 37 (custo do EC2).

Figura 32 - Configurações básicas do liquidador de saldo negativo.

Fonte: (AMAZON WEB SERVICES, 2023)

Figura 33 - Configurações de uso do liquidador de saldo negativo.



Fonte: (AMAZON WEB SERVICES, 2023)

O caso do liquidador de saldo negativo é similar ao liquidador de saldo positivo, pois, como detalhado na Seção 3.3.2.3, a quantidade de informação que chega nesses sistemas é menor do que a que chega em seus antecessores, o faturador e o agregador. Com isso, escolheu-se para o liquidador de saldo negativo a mesma máquina capaz de suportar o processamento do liquidador de saldo positivo.

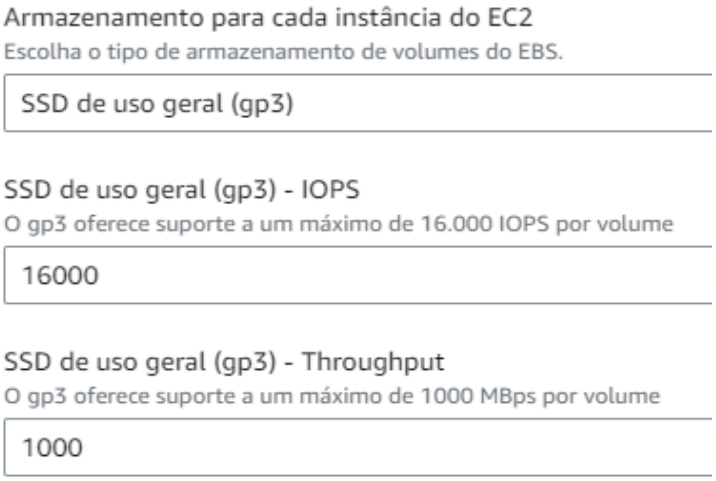
Figura 34 - Configurações de processamento do liquidador de saldo negativo.

	Instance name	Categoria da instância	vCPUs	Núcleos físicos	Memória
○	t3.nano	General purpose	2		0.5 GiB
○	t3.micro	General purpose	2		1 GiB
◉	t3.small	General purpose	2		2 GiB
○	a1.medium	General purpose	1		2 GiB

Fonte: (AMAZON WEB SERVICES, 2023)

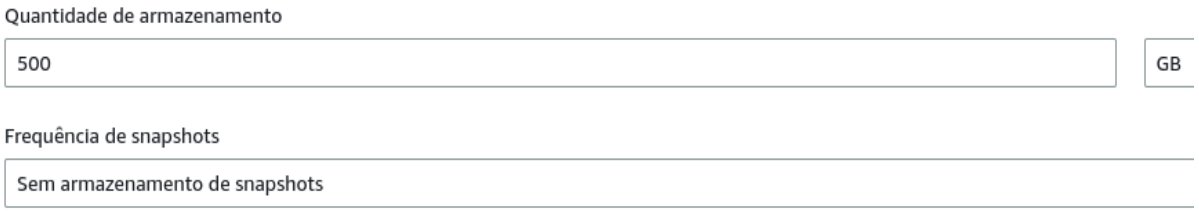
As Figuras 35 e 36 mostram os detalhes da configuração de armazenamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 37 (custo de *Elastic Block Store*).

Figura 35 - Configurações gerais de armazenamento do liquidador de saldo negativo.



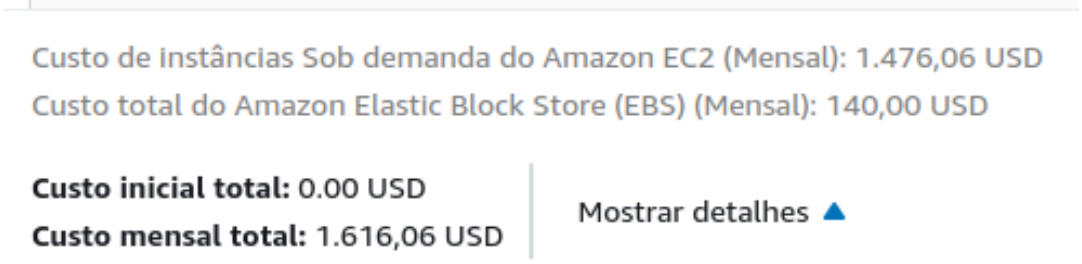
Fonte: (AMAZON WEB SERVICES, 2023)

Figura 36 - Quantidade de armazenamento do liquidador de saldo negativo.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 37 - Custo mensal total para o liquidador de saldo positivo.



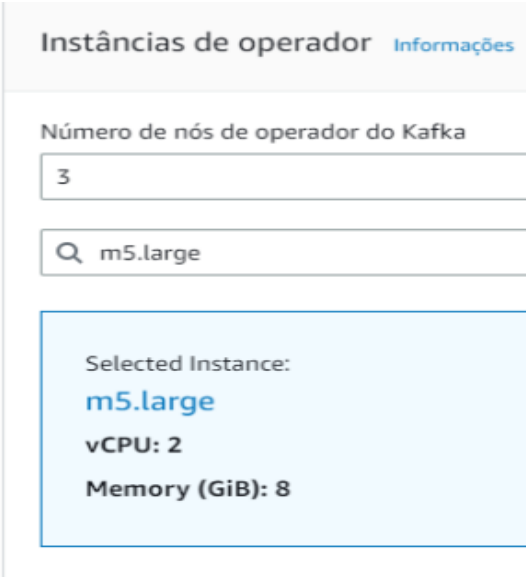
Fonte: (AMAZON WEB SERVICES, 2023)

O armazenamento do liquidador de saldo negativo segue os mesmos critérios dos utilizados no liquidador de saldo positivo por serem sistemas semelhantes. Com isso, selecionou-se o máximo de *throughput* e IOPS para lidar com picos de requisição, mas com uma quantidade de armazenamento menor que a escolhida para o faturador e para o agregador.

3.3.2.5. COMUNICAÇÃO ASSÍNCRONA

As Figuras 38 a 40 mostram os detalhes da configuração escolhida para a comunicação assíncrona na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 41.

Figura 38 - Configurações gerais para comunicação assíncrona de microsserviços.

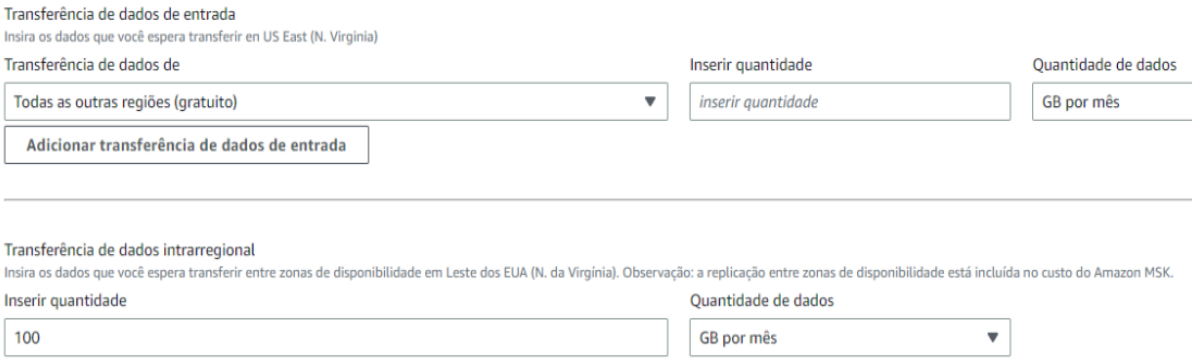


Fonte: (AMAZON WEB SERVICES, 2023)

Figura 39 - Configurações de armazenamento para comunicação assíncrona de microsserviços.

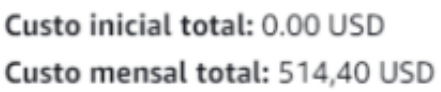
Fonte: (AMAZON WEB SERVICES, 2023)

Figura 40 - Configurações para transferência de dados de comunicação assíncrona de microsserviços.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 41 - Custo mensal total para comunicação assíncrona de microsserviços.



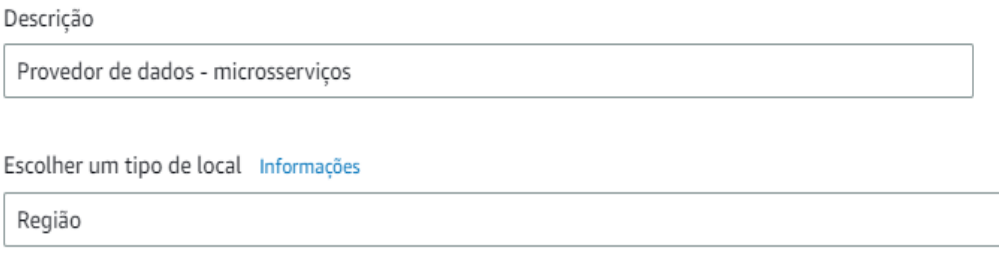
Fonte: (AMAZON WEB SERVICES, 2023)

Essas características foram escolhidas para suportarem o volume de informações que os microsserviços deverão processar. Cada serviço publicará uma mensagem em um tópico que terá o serviço seguinte como consumidor. Esse tópico precisa armazenar essas informações publicadas e pode-se configurar um tempo para que as mensagens sejam apagadas do tópico, diminuindo a necessidade de armazenamento.

3.3.2.6. PROVEDOR DE DADOS DE CLIENTES

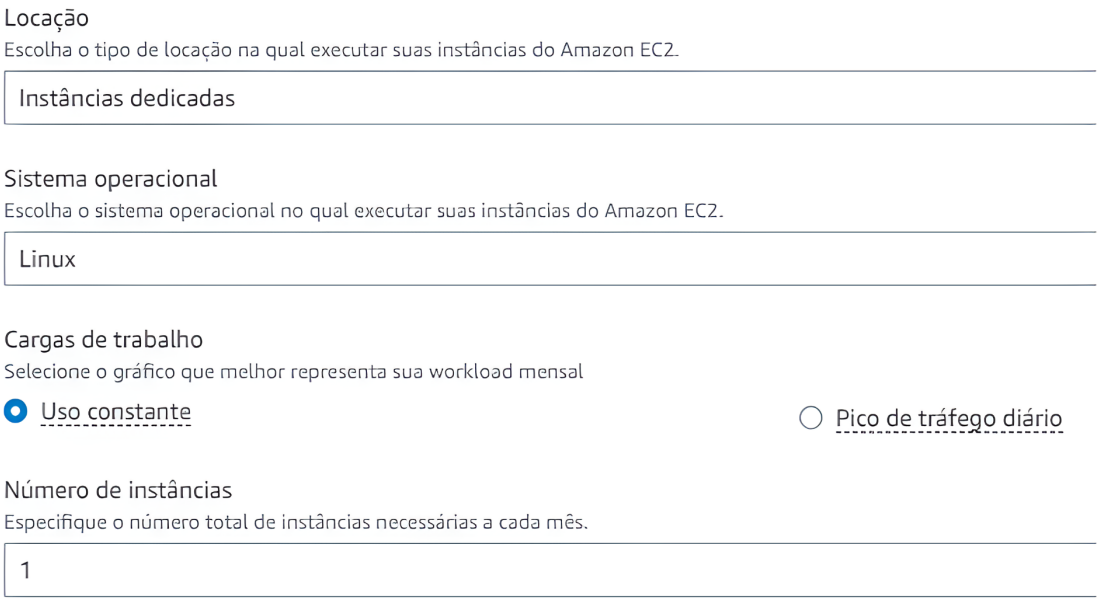
As Figuras 42 a 44 mostram os detalhes da configuração de processamento escolhida para o provedor de dados de clientes na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 47 (custos do EC2).

Figura 42 - Configurações gerais do provedor de dados de clientes.



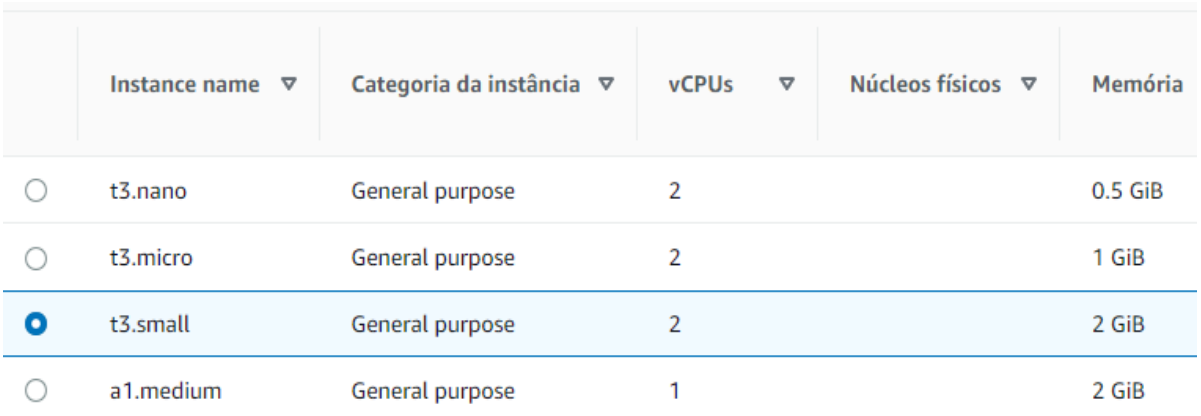
Fonte: (AMAZON WEB SERVICES, 2023)

Figura 43 - Configurações de uso do provedor de dados de clientes.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 44 - Configurações de processamento do provedor de dados de clientes.

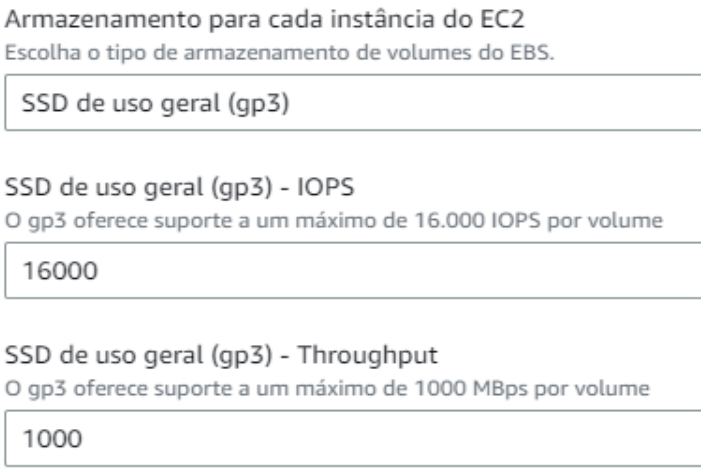


	Instance name	Categoria da instância	vCPUs	Núcleos físicos	Memória
○	t3.nano	General purpose	2		0.5 GiB
○	t3.micro	General purpose	2		1 GiB
◉	t3.small	General purpose	2		2 GiB
○	a1.medium	General purpose	1		2 GiB

Fonte: (AMAZON WEB SERVICES, 2023)

As Figuras 45 e 46 mostram os detalhes da configuração de armazenamento escolhida para o agregador na arquitetura de microsserviços com base nos requisitos detalhados na Seção 3.1. Os custos dessa configuração são detalhados na Figura 47 (custo de *Elastic Block Store*).

Figura 45 - Configurações gerais de armazenamento do provedor de dados de clientes.



Fonte: (AMAZON WEB SERVICES, 2023)

Figura 46 - Quantidade de armazenamento do provedor de dados de clientes.

Fonte: (AMAZON WEB SERVICES, 2023)

Figura 47 - Custo total mensal do provedor de dados de clientes.

Fonte: (AMAZON WEB SERVICES, 2023)

O provedor de dados de clientes recebe uma alta quantidade de requisições de outros sistemas para poder fornecer as informações necessárias para que o fluxo financeiro aconteça, como endereço, CNPJ e outros. Por isso, exige um alto *throughput* e IOPS e considerou-se o máximo para ambos. Para a capacidade de armazenamento, considerou-se a metade da capacidade do faturador e do

agregador, pois o provedor de dados de clientes apenas armazena as informações de clientes, o que representa uma quantidade consideravelmente menor do que o armazenado por esses dois sistemas.

4. RESULTADOS E DISCUSSÕES

Para os cálculos de custos de cada arquitetura, deve-se somar os valores de armazenamento e processamento dos mesmos. No caso do monólito, apenas um custo com processamento e um com armazenamento. Para os microsserviços, deve-se somar os custos de cada sistema (faturador, agregador, liquidadores e provedor de dados), além do custo de comunicação assíncrona entre eles.

4.1. MONÓLITO

O custo final para manter um sistema monolítico por um mês é o valor total mensal da Figura 13, ou seja, U$ 1.820,19.

4.2. MICROSSERVIÇOS

Para os microsserviços, devemos somar os valores correspondentes aos custos da Seção 3.1.2, como evidencia a Equação 1.

$$Custo\ Total\ =\ Custo\ da\ Máquina\ +\ Custo\ de\ Armazenamento \quad (1)$$

Na Equação 2 temos os custos relacionados ao faturador.

$$U\$\ 1.492,19\ +\ U\$\ 181,92\ =\ U\$\ 1.674,11 \quad (2)$$

Na Equação 3 temos os custos relacionados ao agregador.

$$U\$\ 1.492,19\ +\ U\$\ 181,92\ =\ U\$\ 1.674,11 \quad (3)$$

Na Equação 4 temos os custos relacionados ao liquidador de saldo positivo.

$$U\$\ 1.476,06\ +\ U\$\ 140,00\ =\ U\$\ 1.616,06 \quad (4)$$

Na Equação 5 temos os custos relacionados ao liquidador de saldo negativo.

$$U\$\ 1.476,06\ +\ U\$\ 140,00\ =\ U\$\ 1.616,06 \quad (5)$$

Na Equação 6 temos os custos relacionados ao provedor de dados de clientes.

$$U\$\ 1.476,06\ +\ U\$\ 140,00\ =\ U\$\ 1.616,06 \quad (6)$$

Assim, somamos os resultados das equações 2 a 6 com o custo da comunicação assíncrona entre os serviços que, de acordo com a Figura 42, é de $U\$\ 514,40$. Assim, tem-se a Equação 7 para o cálculo do custo total e, aplicando os valores simulados na Seção 3.3.2, tem-se a Equação 8.

$$Custo\ Total\ =\ Custo\ Total\ dos\ Microsserviços\ +\ Custo\ de\ Comunicação \quad (7)$$

$$U\$\ 1.674,11\ +\ U\$\ 1.674,11\ +\ U\$\ 1.616,06\ +\ U\$\ 1.616,06\ +\ U\$\ 1.616,06\ +\ U\$\ 514,40\ =\ U\$\ 8.710,80 \quad (8)$$

Isso significa que, para manter os microsserviços atuando, tem-se, aproximadamente, um custo de U$ 8.710,80.

Para comparar os custos de microsserviços e do monólito, tem-se a Equação 9 que, aplicando os valores calculados nesta Seção, tem-se a Equação 10.

$$Diferença\ (\%)\ =\ \frac{(Custo\ dos\ Microsserviços\ -\ Custo\ do\ Monólito)}{Custo\ do\ Monólito}\ *\ 100\% \quad (9)$$

$$(U\$\ 8710,80\ -\ U\$\ 1820,19\ )\div U\$\ 1820,19\ *\ 100\%\ =\ 378,56\% \quad (10)$$

Com base nela, podemos concluir que há um aumento de aproximadamente 378% nos custos quando adota-se a arquitetura de microsserviços em comparação

à arquitetura monolítica principalmente pela utilização de instâncias únicas para cada serviço.

Observando-se apenas o fator custo, os monólitos apresentam uma grande vantagem em relação aos microsserviços. Porém, no ambiente corporativo, outros fatores são considerados na tomada de decisão, como organização dos times, agilidade na entrega e capacidade técnica da equipe. A divisão de times com base em contextos definidos já se provou efetiva para a manutenção de um sistema no médio e longo prazo. Caso não existam contextos bem definidos entre times, isso significa que várias pessoas podem propor alguma alteração. Assim, mais alinhamentos são necessários para que o trabalho de uma pessoa não impacte no da outra. Com isso, aumenta-se a dificuldade em propor melhorias no sistema o que contribui para a diminuição nas melhorias do serviço e dificulta a manutenção no longo prazo (NEWMAN, 2022).

Os microsserviços são uma ótima alternativa para lidar com esse problema, pois a divisão em contextos bem definidos permite que cada time tenha autonomia no seu próprio sistema e não enfrente problemas externos durante esse processo, pois traz a noção de propriedade ao sistema (cada time é dono de um contexto específico), desde a criação até a implantação e manutenção do mesmo.

Isso é melhor no cenário de microsserviços pois é mais fácil que times pequenos lidem com serviços pequenos, o que aumenta a autonomia e a velocidade na entrega. Além disso, serviços em contextos bem definidos são mais fáceis de manter a comunicação entre si, visto que seus times possuem domínio sobre sua própria ferramenta (NEWMAN, 2022).

Logo, a escolha dos microsserviços é baseada majoritariamente em pessoas ao invés de ser pautada em custos e tecnologias, pois a grande vantagem apresentada pelos microsserviços é a organização que os times ganham com essa arquitetura.

Serviços menores em contextos bem definidos contribuem para a resiliência do fluxo. Ou seja, cada sistema consegue funcionar de maneira independente, visto que a comunicação entre eles é feita de forma assíncrona. Assim, um sistema consegue continuar funcionando mesmo que um serviço que ele dependa esteja fora do ar. Além disso, essa divisão permite a escalabilidade de serviços independentes. Ou seja, se um serviço estiver sendo demandado acima do normal, pode-se prover

mais processamento apenas para esse serviço ao invés de prover para todo o fluxo como seria no caso de monólitos, pois todas as regras estariam contidas em um único sistema.

5. CONCLUSÃO

A adoção da arquitetura de microsserviços representou um aumento considerável nos custos em comparação à arquitetura monolítica, sendo cerca de 378% mais cara principalmente por utilizar instâncias únicas para cada serviço, ou seja, cada sistema tem sua própria máquina rodando para atender às suas demandas. Logo, quanto maior for o número de microsserviços em comparação ao monólito, maior tende a ser essa porcentagem. Isso não significa que os microsserviços sejam piores que os monólitos, pois no ambiente corporativo existem diversos outros fatores que contribuem para a tomada de decisão na hora de escolher a melhor arquitetura para um fluxo de informações. Pontos como lentidão no processo de desenvolvimento, concorrência entre atualizações no software, maturidade da equipe, nível de conhecimento técnico da equipe e diversos outros também impactam a saúde financeira do negócio e dos colaboradores e, com isso, são levadas em conta na hora dessa escolha. Os microsserviços, por exemplo, trabalham de forma independente, garantindo resiliência ao fluxo. Isso significa que cada sistema trabalha mesmo que outro esteja fora do ar, o que diminui o risco de haver perda nas informações, já que a comunicação entre sistemas é feita de forma assíncrona. Por fim, este trabalho conclui que, sob a ótica única e exclusiva dos custos, os monólitos são consideravelmente mais econômicos.

REFERÊNCIAS BIBLIOGRÁFICAS

AMAZON WEB SERVICES. **Calculadora de preços da Amazon Web Services.** [*S. l.*], 2023. Disponível em: https://calculator.aws/#/estimate. Acesso em: 21 maio 2023.

EVANS, Eric. **Domain-Driven Design:** Atacando as complexidades no coração do software. [S. l.]: Alta Books, 2016.

FOWLER, Susan. **Microsserviços Prontos Para a Produção:** Construindo Sistemas Padronizados em uma Organização de Engenharia de Software. [S. l.]: Novatec, 2017.

MARTIN, Robert C. **Código limpo**: Habilidades práticas do Agile Software. [*S. l.*]: Alta Books, 2009.

NEWMAN, S. **Criando Microsserviços** - 2ª Edição. [*S. l.*]: Novatec Editora, 2022.

VERAS, Manoel. **Cloud Computing:** Nova Arquitetura da TI. [*S. l.*]: Brasport, 2012.