UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

BRENO CAMPOS GOMES

ARQUITETURA SERVERLESS DISTRIBUÍDA PARA ANALISE DO PREÇO DO DÓLAR

SOBRAL

2024

BRENO CAMPOS GOMES

ARQUITETURA SERVERLESS DISTRIBUÍDA PARA ANALISE DO PREÇO DO DÓLAR

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Júnior

SOBRAL

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G1a GOMES, BRENO.
ARQUITETURA SERVERLESS DISTRIBUÍDA PARA ANALISE DO PREÇO DO DÓLAR / BRENO GOMES. – 2024.
26 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2024.
Orientação: Prof. Dr. Iális Cavalcante de Paula Júnior.

1. Serviços AWS. I. Título.

CDD 621.39

BRENO CAMPOS GOMES

ARQUITETURA SERVERLESS DISTRIBUÍDA PARA ANALISE DO PREÇO DO DÓLAR

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula Júnior (Orientador)
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

Este trabalho apresenta o desenvolvimento de uma arquitetura *serverless* distribuída para a análise em tempo real do preço do dólar, utilizando os serviços da AWS. A arquitetura proposta utiliza o AWS Lambda para automação e escalabilidade, o Amazon EventBridge para agendamento das coletas de dados, e o DynamoDB para armazenamento. A API FreeCurrConv foi utilizada como fonte de dados cambiais, e notificações automáticas foram configuradas via Amazon SES. Experimentos realizados demonstraram que o sistema é capaz de escalar automaticamente, mantendo tempos de resposta consistentes, mesmo sob alta carga de requisições. Entretanto, a limitação da API em termos de número de requisições por minuto destacou a necessidade de utilizar alternativas mais robustas para aplicações de alta demanda. O trabalho conclui que a arquitetura *serverless* é uma solução eficiente, escalável e economicamente viável para monitoramento cambial em tempo real, com espaço para futuras melhorias e expansão.

Palavras-chave: Arquitetura *serveless*. Serviços AWS. Preço Dólar.

ABSTRACT

This work presents the development of a distributed serverless architecture for real-time analysis of the US dollar exchange rate, using AWS services. The proposed architecture leverages AWS Lambda for automation and scalability, Amazon EventBridge for data collection scheduling, and DynamoDB for storage. The FreeCurrConv API was used as the source for currency data, and automatic notifications were configured via Amazon SES. Experiments demonstrated that the system can automatically scale, maintaining consistent response times even under high request loads. However, the API's limitation on the number of requests per minute highlighted the need for more robust alternatives in high-demand applications. The study concludes that the serverless architecture is an efficient, scalable, and cost-effective solution for real-time currency monitoring, with room for future improvements and expansion.

Keywords: Serverless architecture. AWS services. Dollar price.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A análise do preço do dólar é de fundamental importância em um mundo cada vez mais interligado economicamente. O dólar americano é considerado uma moeda de referência global e, como tal, suas flutuações têm um impacto direto e abrangente. Desde investidores individuais até grandes corporações e instituições financeiras, todos dependem do entendimento e acompanhamento das tendências cambiais do dólar para tomar decisões informadas. Além disso, os responsáveis pela formulação de políticas econômicas de um país monitoram de perto o comportamento do dólar, pois ele tem implicações significativas na inflação, no crescimento econômico e na estabilidade financeira (EYLL, 2008).

A análise constante do preço do dólar é uma tarefa contínua e que pode ser automatizada para diversos fins. Nesse contexto, a tecnologia pode desempenhar um papel cada vez mais relevante na automação e aprimoramento da análise do preço do dólar. A arquitetura *serverless* emerge como uma abordagem notável para otimizar esse tipo de análise, pois permite coletar e processar dados em tempo real, atendendo à demanda variável. A arquitetura *serverless* é um modelo de computação em nuvem onde o provedor de serviços gerencia a infraestrutura e automaticamente aloca os recursos necessários para executar o código dos desenvolvedores. Nesse modelo, os desenvolvedores podem se concentrar exclusivamente na lógica de suas aplicações sem se preocupar com a manutenção de servidores, escalabilidade ou balanceamento de carga (FONTANA; ZANELATTO, 2019).

Uma das principais vantagens da arquitetura *serverless* é a capacidade de fornecer respostas em tempo real, algo crucial para aqueles que dependem de análises cambiais para tomar decisões imediatas (NGUYEN *et al.*, 2019). A análise do preço do dólar em tempo real permite que investidores ajam com rapidez, empresas façam ajustes em suas estratégias de comércio internacional e órgãos governamentais implementem políticas adequadas para lidar com as oscilações cambiais.

Além disso, a abordagem *serverless* contribui para reduzir custos operacionais, já que os recursos de computação são alocados somente quando necessário, evitando gastos desnecessários. Ademais, a distribuição de tarefas em diferentes funções *serverless* torna o sistema mais resiliente e capaz de lidar com possíveis falhas, garantindo a confiabilidade da análise (MAHMOUDI; KHAZAEI, 2020).

O Amazon Web Services (AWS) Lambda é frequentemente reconhecido como o primeiro serviço a ser chamado de *serverless*. Nele, o desenvolvedor cria uma função em uma

das linguagens de programação suportadas, que é então disponibilizada em uma URL. Essa função é acionada conforme critérios definidos previamente, como uma solicitação HTTP ou um evento desencadeado por uma fila de mensagens (VIEIRA *et al.*, 2020).

O objetivo geral deste trabalho é desenvolver e avaliar uma arquitetura *serverless* distribuída para a análise em tempo real do preço do dólar, com foco em eficiência, escalabilidade e automação. Para atingir o objetivo geral, os objetivos específicos deste trabalho são:

- Projetar um sistema ágil de coleta e processamento de informações cambiais.
- Implementar uma infraestrutura escalável baseada na arquitetura *serverless*.
- Integrar mecanismos de tendências no comportamento do dólar.
- Avaliar a eficácia da arquitetura em termos de agilidade, escalabilidade e adaptabilidade a cenários cambiais variáveis.

2 FUNDAMENTAÇÃO TEÓRICA

A computação em nuvem tem revolucionado a forma como as aplicações são desenvolvidas e implantadas. Dentro desse paradigma, a arquitetura serverless emergiu como uma abordagem poderosa para construir sistemas escaláveis e eficientes sem a necessidade de gerenciar a infraestrutura subjacente. Este capítulo explora em profundidade a arquitetura serverless e os principais serviços da Amazon Web Services (AWS) que suportam esse modelo, detalhando suas características, vantagens e casos de uso (FRANTZ *et al.*, 2014).

2.1 Arquitetura *Serverless*

A arquitetura *serverless*, ou "sem servidor", representa uma mudança significativa em relação aos modelos tradicionais de computação, onde os desenvolvedores são responsáveis pelo provisionamento e gerenciamento de servidores físicos ou virtuais. Neste modelo, a infraestrutura é abstraída e gerenciada pelo provedor de serviços em nuvem, permitindo que os desenvolvedores se concentrem na lógica de suas aplicações (FILHO, 2024).

2.1.1 Conceito de Serverless

No contexto da arquitetura *serverless*, o termo "sem servidor"é algo enganoso, pois os servidores ainda existem, mas são completamente gerenciados pelo provedor de serviços. Isso significa que o desenvolvedor não precisa se preocupar com a configuração, manutenção ou escalabilidade dos servidores. Em vez disso, ele define apenas a lógica da aplicação em forma de funções que são executadas em resposta a eventos específicos.

Essas funções, muitas vezes chamadas de Functions as a Service (FaaS), são pequenas unidades de código que realizam tarefas específicas. Elas são ativadas em resposta a eventos como requisições HTTP, mudanças em bancos de dados, mensagens em filas ou eventos de cronômetro, e são cobradas apenas pelo tempo de execução e recursos utilizados, tornando o modelo altamente eficiente em termos de custo (FERNANDES *et al.*, 2021).

2.1.2 Vantagens da Arquitetura Serverless

A arquitetura *serverless* oferece várias vantagens em relação às abordagens tradicionais:

- Escalabilidade Automática: As funções *serverless* escalam automaticamente em resposta à demanda, eliminando a necessidade de configurar manualmente a escalabilidade vertical ou horizontal.
- Custo Efetivo: Como os desenvolvedores pagam apenas pelo tempo de execução e pelos recursos consumidos, os custos podem ser significativamente reduzidos, especialmente em cargas de trabalho intermitentes ou de baixa utilização.
- Redução da Complexidade Operacional: Com o gerenciamento de servidores sendo responsabilidade do provedor, a complexidade operacional é drasticamente reduzida, permitindo que as equipes de desenvolvimento se concentrem mais na lógica de negócio e menos na infraestrutura.
- Desenvolvimento Rápido e Flexível: A arquitetura *serverless* permite que os desenvolvedores criem, testem e implantem funções de maneira rápida e iterativa, facilitando a inovação e o tempo de entrada no mercado.

2.1.3 Desafios da Arquitetura Serverless

Apesar das vantagens, a arquitetura *serverless* também apresenta desafios:

- Latência de Inicialização: Funções *serverless* podem sofrer de latência de inicialização, especialmente quando estão "frias"(não foram recentemente ativadas). Isso pode ser um problema em aplicações que requerem respostas em tempo real.
- Complexidade de Depuração e Monitoramento: A natureza distribuída e baseada em eventos das aplicações *serverless* pode tornar a depuração e o monitoramento mais complexos. Ferramentas adicionais são frequentemente necessárias para rastrear a execução das funções e detectar problemas.
- Limitações de Configuração: Funções *serverless* têm limites específicos de tempo de execução, tamanho de pacote e memória, o que pode restringir o uso em certas aplicações ou exigir a divisão de tarefas em funções menores.

2.1.4 Segurança em Arquiteturas Serverless

Embora a arquitetura *serverless* simplifique muitos aspectos do desenvolvimento de software, a segurança continua sendo uma preocupação crítica. A responsabilidade pela segurança é compartilhada entre o provedor de nuvem e o cliente, exigindo atenção cuidadosa em vários níveis.

Um dos principais aspectos de segurança em ambientes *serverless* é o controle de acesso. Na AWS, isso é gerenciado principalmente através do IAM (Identity and Access Management), que define quem pode acessar o quê. Funções Lambda, por exemplo, devem ter permissões IAM apropriadas para acessar outros serviços da AWS, como DynamoDB ou S3 (FERNANDES *et al.*, 2021).

A auditoria e o monitoramento são componentes essenciais para manter a segurança e a conformidade em sistemas *serverless*. Serviços como AWS CloudTrail e CloudWatch Logs permitem rastrear a atividade de contas e monitorar o comportamento das funções Lambda, fornecendo uma trilha de auditoria e ajudando a identificar e responder a incidentes de segurança.

2.1.5 Melhores Práticas para Desenvolvimento Serverless

Para maximizar os benefícios da arquitetura *serverless*, é importante seguir as melhores práticas de desenvolvimento, que incluem desde a organização do código até a gestão eficiente dos recursos.

2.1.5.1 Modularização e Reutilização de Código

Em ambientes *serverless*, modularizar o código em funções pequenas e reutilizáveis é uma prática recomendada. Isso facilita a manutenção e a escalabilidade da aplicação, permitindo que componentes sejam atualizados ou substituídos sem impactar todo o sistema.

2.1.5.2 Gerenciamento de Logs e Monitoramento

A gestão eficaz de logs e o monitoramento contínuo são cruciais para manter a saúde e o desempenho das aplicações *serverless*. Usar serviços como AWS CloudWatch Logs para coletar, monitorar e analisar logs ajuda a detectar problemas antes que eles afetem a aplicação.

2.1.5.3 Uso Eficiente dos Recursos

Embora a arquitetura *serverless* seja altamente eficiente em termos de custo, é importante otimizar o uso dos recursos para evitar desperdícios. Configurar limites de tempo adequados para as funções Lambda, escolher classes de armazenamento de arquivos, como o S3 (*Simple Storage Service*), apropriadas e otimizar o uso de capacidades provisionadas no DynamoDB, serviço de banco de dados NoSQL que veremos melhor na frente, são práticas que

ajudam a manter os custos sob controle.

2.2 Serviços AWS

A Amazon Web Services (AWS) oferece um conjunto robusto de serviços que suportam a arquitetura *serverless*. Esses serviços cobrem desde a execução de funções até o armazenamento de dados, a integração de APIs e o gerenciamento de eventos. A seguir, são discutidos alguns dos serviços mais importantes para a construção de aplicações *serverless* na AWS (MOTA; OLIVEIRA, 2024).

2.2.1 AWS Lambda

O AWS Lambda é o serviço central da AWS para a execução de funções *serverless*. Com o Lambda, os desenvolvedores podem executar código em várias linguagens de programação suportadas (como Python, Node.js, Java, e Go) em resposta a eventos disparados por outros serviços da AWS ou fontes externas.

O AWS Lambda funciona carregando e executando funções em resposta a eventos. Cada função Lambda é composta por um código-fonte, configuração de *runtime* e permissões de execução. Quando um evento dispara uma função, o AWS Lambda gerencia automaticamente todos os recursos computacionais necessários para executar o código e fornece logs detalhados sobre a execução.

Os casos de uso comuns para AWS Lambda incluem processamento de dados em tempo real, automação de tarefas administrativas, *backends* para aplicações móveis ou web, e integração de sistemas através de APIs. Por exemplo, uma função Lambda pode ser configurada para processar arquivos enviados para um *bucket* S3, realizando operações como redimensionamento de imagens ou análise de dados.

2.2.2 Amazon DynamoDB

O Amazon DynamoDB é um serviço de banco de dados NoSQL gerenciado que fornece armazenamento rápido e previsível para aplicações *serverless*. Ele é particularmente adequado para aplicações que exigem alta disponibilidade e desempenho consistente em larga escala.

O DynamoDB utiliza um modelo de dados baseado em tabelas, onde cada item

pode ter uma estrutura flexível de atributos. Esse modelo é altamente adequado para aplicações que exigem uma estrutura de dados não rígida, permitindo que os desenvolvedores adaptem o armazenamento de dados às necessidades específicas da aplicação.

O DynamoDB é projetado para oferecer latência baixa e escalabilidade automática. Ele pode lidar com grandes volumes de dados e tráfego, ajustando-se dinamicamente para atender à demanda. A capacidade de leitura e escrita pode ser provisionada ou ajustada automaticamente, dependendo das necessidades da aplicação.

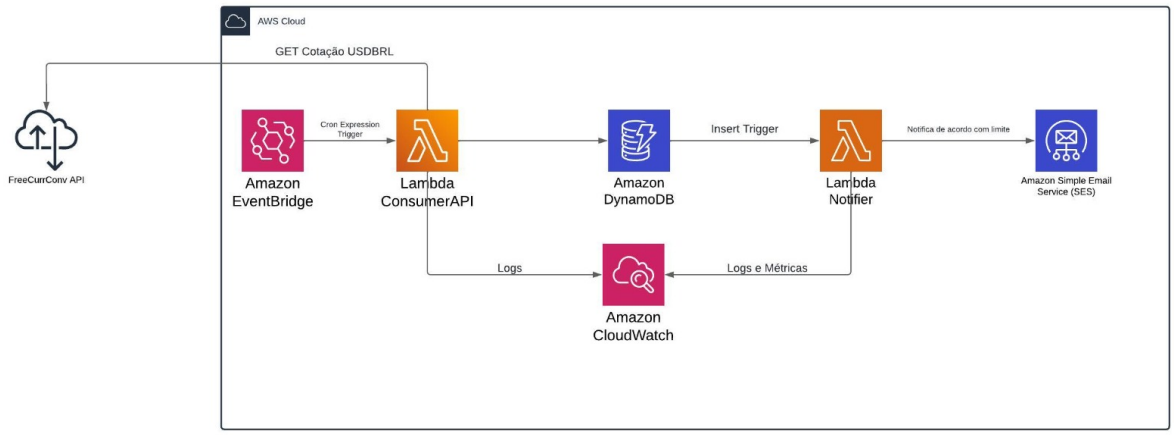
3 METODOLOGIA

Neste capítulo, detalha-se o processo metodológico adotado para o desenvolvimento de uma arquitetura *serverless* distribuída para análise do preço do dólar. A metodologia empregada consiste em uma série de etapas interconectadas, implementadas de forma sequencial e interdependente para garantir a eficácia, escalabilidade e precisão da solução proposta.

3.1 Arquitetura Proposta

A arquitetura desenvolvida para a análise do preço do dólar utiliza o Amazon EventBridge, configurado com uma Cron Expression Trigger, sendo assim a única maneira de invocação do código, dado que não há outros *cron jobs* no servidor. Esta função Lambda acessa a API FreeCurrConv (Free Currency Converter API), que oferece serviços web gratuitos para conversão de moedas, com atualizações a cada 60 minutos. Os dados obtidos são então enviados para o Amazon DynamoDB, e os logs são armazenados no Amazon CloudWatch. Caso os últimos dados apresentem uma alteração significativa, o DynamoDB gera um trigger para um Lambda notifier, que aciona o Amazon Simple Email Service (SES) e armazena logs e métricas no Amazon CloudWatch. A arquitetura é ilustrada na Figura 1.

Figura 1 – Arquitetura Proposta



Fonte: Próprio autor.

A função cron utilizada na configuração do Amazon EventBridge é essencial para a automação do processo. Em sistemas de computação, a cron expression é uma maneira de agendar tarefas para serem executadas em momentos específicos, de forma repetida. No caso desta arquitetura, a expressão cron(0 13-20 ? * MON-FRI *) significa que a função Lambda

será acionada a cada hora, entre as 13h e 20h UTC, de segunda a sexta-feira. Esse intervalo de horário refere-se ao intervalo entre 10h e 17h no horário de Brasília, que coincide com horário de funcionamento da bolsa. Essa configuração assegura que as cotações do dólar sejam coletadas regularmente durante o período de maior atividade nos mercados financeiros.

3.2 Seleção da API de Acesso aos Dados Cambiais

O primeiro passo no desenvolvimento da arquitetura envolve a seleção da API de acesso aos dados cambiais. A API FreeCurrConv é escolhida como a fonte principal de dados sobre o preço do dólar. A escolha baseia-se em critérios como custo, tempo de atualização dos dados, que ocorre a cada 60 minutos, e compatibilidade com as necessidades específicas da aplicação.

A seleção criteriosa da API assegura que os dados utilizados na análise sejam precisos e atualizados. Também se considera a facilidade de integração da API com os serviços AWS, bem como sua confiabilidade e disponibilidade, garantindo que ela possa sustentar as operações contínuas do sistema.

3.3 Configuração do Amazon EventBridge

Após a escolha da API, configura-se o Amazon EventBridge, que monitora a API FreeCurrConv e coleta informações cambiais em intervalos regulares definidos pela Cron Expression Trigger. A configuração do EventBridge é crucial para manter um fluxo constante de dados para a aplicação, permitindo que as análises sejam realizadas com base em informações atualizadas.

Neste processo, definem-se regras que determinam a frequência com que os dados são coletados e os eventos que acionam o processamento subsequente. Isso inclui a definição de intervalos de tempo adequados para a coleta de dados, considerando as flutuações típicas no preço do dólar e as necessidades de análise do sistema.

3.4 Processamento de Dados com AWS Lambda

O processamento dos dados coletados é realizado através de funções Lambda, que são acionadas automaticamente pelos eventos gerados no EventBridge. Essas funções Lambda acessam a API, processam os dados recebidos e os enviam para o Amazon DynamoDB para

armazenamento.

A configuração das funções Lambda envolve a definição de *triggers* baseados nos eventos do EventBridge, a escrita do código que implementa as análises desejadas e a otimização do ambiente de execução para garantir desempenho eficiente. As funções são projetadas para serem escaláveis, permitindo que o sistema se ajuste automaticamente ao volume de dados sendo processado.

3.5 Armazenamento dos Resultados no Amazon DynamoDB

Os resultados das análises realizadas pelas funções Lambda são armazenados no Amazon DynamoDB, um banco de dados NoSQL gerenciado que oferece alta disponibilidade e desempenho escalável. O DynamoDB é escolhido por sua capacidade de lidar com grandes volumes de dados e por sua integração perfeita com os outros serviços AWS utilizados na arquitetura.

Nesta etapa, projeta-se a estrutura das tabelas DynamoDB de maneira a otimizar o armazenamento e a recuperação dos dados. Isso inclui a definição de chaves de partição e de classificação que permitem consultas eficientes, bem como a configuração de índices secundários que suportam diferentes padrões de acesso aos dados.

3.6 Detecção de Eventos e Notificações

Além do armazenamento de dados, a arquitetura inclui mecanismos para a detecção de eventos específicos, como tendências de alta ou baixa consecutivas no preço do dólar. O Amazon CloudWatch monitora continuamente os logs armazenados, e ao detectar uma alteração significativa, gera um trigger que aciona um Lambda notifier. Este, por sua vez, utiliza o Amazon Simple Email Service (SES) para enviar notificações por e-mail para um conjunto de contas previamente estabelecidas.

De forma específica, consideramos uma alteração significativa quando o valor atual é 20 % maior ou menor que a média das últimas cinco cotações. Caso este cenário aconteça o Lambda notifier é ativado para que o mesmo possa gerar uma requisição de envio de e-mail para as contas cadastradas.

Este processo de detecção e notificação é configurado utilizando *triggers* baseados em mudanças de estado no DynamoDB e nos *logs* do CloudWatch, permitindo que as notifi-

cações sejam enviadas em tempo real. As funções Lambda são programadas para acessar e analisar os dados armazenados, aplicando regras predefinidas para determinar quando um evento significativo ocorre.

4 EXPERIMENTOS E RESULTADOS OBTIDOS

Com o objetivo de validar a arquitetura proposta e explorar experimentos que demonstrem, mesmo que em alguns estudos de caso, a eficiência, escalabilidade e automação da arquitetura.

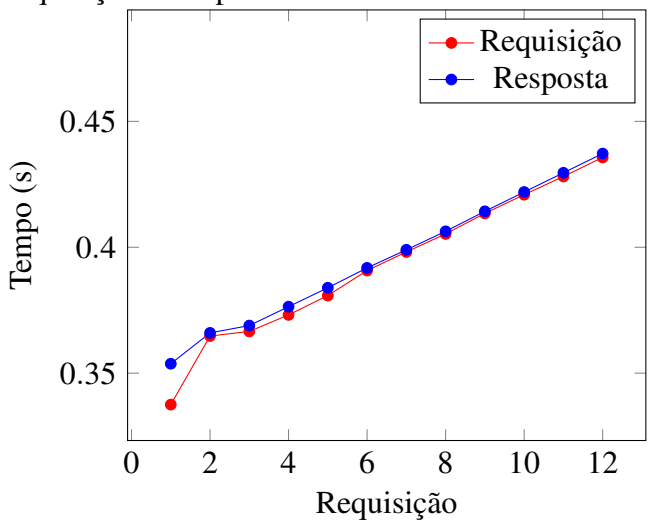
Um primeiro experimento realizado consistiu em alterar o último valor armazenado no Dynamo e inserir um valor alto, ou seja, gerando um *outline*. Com isso, a próxima requisição que gerou um valor comum estava abaixo da média dos valores anteriores por conta do *outlier*. Como esperado, o sistema conseguiu identificar e enviar a notificação aos e-mails cadastrados.

A metodologia deste primeiro experimento foi adotada devido ao fato que o mesmo força uma reação esperada do sistema, pois no cenário real poderíamos demorar meses ou até anos para que a condição de ativação da notificação fosse atendida.

A escalabilidade na arquitetura *serverless* da AWS é garantida automaticamente pelo próprio serviço AWS Lambda, que gerencia a alocação de recursos de forma dinâmica e eficiente. Tanto a escalabilidade horizontal quanto a vertical são implementadas de maneira automática, sem a necessidade de intervenção manual. A escalabilidade horizontal ocorre ao se aumentar o número de instâncias de funções Lambda conforme a demanda cresce, enquanto a escalabilidade vertical ajusta a capacidade computacional de cada instância com base nos requisitos de processamento. Dessa forma, a AWS assegura que o sistema se adapte a qualquer variação de carga, mantendo o desempenho estável e eficiente.

Um segundo experimento foi realizado para avaliar o tempo de resposta do sistema quando submetido a várias requisições consecutivas em um curto intervalo de tempo. A Figura 2 apresenta os tempos de requisição e resposta para uma série de doze requisições feitas de maneira sequencial. Esse experimento visa testar a capacidade de resposta do sistema em cenários de alta demanda, simulando situações reais, como mercados financeiros ativos, onde múltiplas solicitações podem ser feitas quase simultaneamente. Esse tipo de teste é fundamental para avaliar a resiliência e a capacidade de adaptação da arquitetura *serverless*.

Figura 2 – Tempo de Requisição e Resposta



Fonte: Próprio autor.

A análise dos resultados apresentados na Figura 2 revela que o tempo de resposta (linha azul) acompanha de maneira muito próxima o tempo de requisição (linha vermelha). As diferenças entre os tempos de resposta e requisição são mínimas, o que indica que o sistema é eficiente em processar as solicitações sem introduzir latências significativas, mesmo sob a pressão de múltiplas requisições consecutivas. Isso reflete a eficácia da escalabilidade automática proporcionada pela AWS Lambda, que permite ao sistema ajustar dinamicamente seus recursos para manter o desempenho em cenários de alta carga. A capacidade de resposta rápida é crítica para aplicações em tempo real, como a análise do preço do dólar, onde atrasos podem impactar diretamente as decisões dos usuários.

Este teste, no entanto, foi limitado a doze requisições devido às restrições impostas pela FreeCurrConv API, que impõe limites no número de consultas. Este fator pode representar um gargalo para o sistema em cenários onde a frequência de captura de dados é elevada, e caso a aplicação exija uma coleta mais intensiva de dados cambiais, será necessário considerar a substituição da API por outra que ofereça maior capacidade e flexibilidade.

Em resumo, os experimentos realizados demonstraram a robustez e a eficiência da arquitetura *serverless* proposta, validando sua capacidade de escalar automaticamente conforme a demanda aumenta, enquanto mantém um tempo de resposta consistente. O sistema provou ser confiável em situações de alta carga, com potencial para ser aprimorado no futuro, especialmente em relação à fonte de dados utilizada. A análise confirma que a solução é capaz de atender aos requisitos de monitoramento em tempo real e de automação definidos nos objetivos do trabalho.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a avaliação de uma arquitetura *serverless* distribuída para a análise em tempo real do preço do dólar, utilizando recursos da AWS. A proposta buscou explorar as vantagens da computação *serverless*, como escalabilidade automática, eficiência no uso de recursos e redução de custos operacionais, além de automatizar processos críticos de análise cambial e notificação de eventos.

Os experimentos realizados validaram os objetivos do trabalho, demonstrando que a arquitetura proposta é eficiente, escalável e adaptável a diferentes cenários de demanda. O primeiro experimento, ao simular um *outlier* nos dados cambiais, comprovou que o sistema é capaz de detectar e responder automaticamente a variações significativas nos valores de forma rápida, enviando notificações em tempo real. Esse comportamento é crucial para sistemas que dependem de monitoramento constante e reações automáticas, como aqueles usados no mercado financeiro.

A escalabilidade foi outro ponto crítico avaliado e demonstrou ser um dos maiores benefícios da arquitetura *serverless*. Através dos serviços AWS Lambda e DynamoDB, o sistema foi capaz de se ajustar automaticamente ao volume de requisições, mantendo o tempo de resposta consistente mesmo sob alta carga. Esse recurso garante que o sistema possa lidar com picos de demanda sem comprometer o desempenho ou a estabilidade, tornando-o ideal para aplicações que exigem alto nível de disponibilidade e processamento em tempo real.

Entretanto, as limitações da API FreeCurrConv, que impõe restrições no número de requisições por minuto, mostraram ser um potencial gargalo para o sistema. Caso seja necessário aumentar a frequência de coletas de dados, será preciso adotar uma API mais robusta. Apesar dessa limitação, a arquitetura proposta se mostrou robusta e adaptável, cumprindo os objetivos estabelecidos no início deste trabalho.

Em conclusão, a arquitetura *serverless* distribuída proposta para a análise do preço do dólar provou ser uma solução viável, eficiente e escalável para o monitoramento em tempo real de dados cambiais. A combinação de automação, escalabilidade e integração de serviços AWS permitiu que o sistema atendesse às necessidades de análise, respondendo rapidamente a eventos significativos, ao mesmo tempo que manteve custos operacionais sob controle. Esse modelo pode ser expandido e adaptado para outras aplicações que requerem processamento de dados em tempo real, destacando-se como uma alternativa moderna e eficiente na construção de sistemas distribuídos.

Para trabalhos futuros, sugere-se explorar a integração de APIs com maior capacidade de requisições, visando ampliar o escopo de coleta de dados cambiais em intervalos mais curtos, o que permitiria análises ainda mais precisas e em tempo real. Além disso, pode-se investigar a implementação de algoritmos de inteligência artificial para a previsão de tendências cambiais com base nos dados coletados, automatizando também a tomada de decisão. Por fim, a expansão da arquitetura para incluir outras moedas ou indicadores econômicos seria outra abordagem interessante para aumentar a aplicabilidade do sistema em mercados financeiros globais.

REFERÊNCIAS

EYLL, T. X. v. **O prêmio pelo risco cambial: uma análise para as principais moedas globais**. Tese (Doutorado), 2008.

FERNANDES, M. d. S. L.; CARVALHO, L. R. de; ARAUJO, A. P. F. de. Framework de autenticação para faas em nuvem pública. In: SBC. **Anais da IV Escola Regional de Alto Desempenho do Centro-Oeste**. [*S. l.*], 2021. p. 32–34.

FILHO, V. C. Desvendando as nuvens: uma análise comparativa de desempenho de aplicações serverless e containers em provedores de computação em nuvem. Universidade Estadual Paulista (Unesp), 2024.

FONTANA, N. B. B.; ZANELATTO, A. D. Arquitetura serverless baseada em eventos para aplicações web utilizando a aws. **Trabalho de Conclusão de Curso - Universidade do Sul de Santa Catarina**, 2019.

FRANTZ, R.; SAWICKI, S.; ROOS-FRANTZ, F.; CORCHUELO, R.; BASTO-FERNANDES, V.; HERNÁNDEZ, I. Desafios para a implantação de soluções de integração de aplicações empresariais em provedores de computação em nuvem. **XIX Jornada de Pesquisa**, UNIJUI, p. 1–11, 2014.

MAHMOUDI, N.; KHAZAEI, H. Performance modeling of serverless computing platforms. **IEEE Transactions on Cloud Computing**, IEEE, v. 10, n. 4, p. 2834–2847, 2020.

MOTA, G. O.; OLIVEIRA, C. C. de. Computação em nuvem e provisionamento na amazon web services. **Advances in Global Innovation & Technology**, v. 2, n. 2, p. 17–31, 2024.

NGUYEN, H. D.; ZHANG, C.; XIAO, Z.; CHIEN, A. A. Real-time serverless: Enabling application performance guarantees. In: **Proceedings of the 5th International Workshop on Serverless Computing**. [*S. l.*: *s. n.*], 2019. p. 1–6.

VIEIRA, A. G.; PANTUZA, G.; FREIRE, J. H.; DUARTE, L. F.; PACÍFICO, R. D.; VIEIRA, M. A.; VIEIRA, L. F.; NACIF, J. A. Computação serverless: Conceito, aplicações e desafios. **Sociedade Brasileira de Computação**, 2020.