

# Computação em nuvem: Serverless computing

Lucas Pereira de Medeiros  
MCZ-A023 – Redes Convergentes  
Universidade Federal do ABC (UFABC)  
Santo André, Brasil  
lucas.medeiros@aluno.ufabc.edu.br

Vinicius Rodrigues Ribeiro Viana  
MCZ-A023 – Redes Convergentes  
Universidade Federal do ABC (UFABC)  
Santo André, Brasil  
rodrigues.v@aluno.ufabc.edu.br

Tiago Henrique Simionato Machado  
MCZ-A023 – Redes Convergente  
Universidade Federal do ABC (UFABC)  
Santo André, Brasil  
tiago.simionato@aluno.ufabc.edu.br

Murillo Vicentini de Alcantara  
MCZ-A023 – Redes Convergentes  
Universidade Federal do ABC (UFABC)  
Santo André, Brasil  
murillo.alcantara@aluno.ufabc.edu.br

**Resumo**— A computação em nuvem, embora não seja um tema muito recente, apresenta uma grande relevância na área de tecnologia, principalmente por sua influência em diversos setores, como o ambiental, o econômico, de trabalho, de entretenimento, entre outros. Entretanto, mesmo um tanto defasada, ela continua apresentando desenvolvimentos significativos que ampliam a sua aplicabilidade e, consequentemente, seus benefícios, como é o caso da computação sem servidor, onde o provedor de nuvem gerencia automaticamente a infraestrutura necessária para executar e escalar os aplicativos. Neste projeto serão abordados os conceitos de nuvem e computação sem servidor, aprofundando em suas características, funcionamento, benefícios, desafios a serem superados e sua importância, além de demonstrar na prática a aplicação desse conceito utilizando o serviço AWS Lambda da empresa Amazon e um algoritmo básico feito em Python ou Javascript. Ao final do projeto espera-se um relatório abordando os conceitos mencionados junto de uma aplicação funcional em uma nuvem.

**Palavras-chave**— *computação, nuvem, serverless, redes*

## I. INTRODUÇÃO

### A. Computação em Nuvem

A computação em Nuvem surgiu como um sistema ou modelo cujo objetivo é disponibilizar recursos através da internet, alterando a forma como empresas alheias se relacionam às empresas de TI, como as tecnologias de hardware e software são agrupadas, sua interação mútua e sua comercialização. Se trata da possibilidade de acessar arquivos e executar serviços diretamente da internet, ou seja, não é necessário que o usuário possua hardware para armazenar seus dados, já que estes estão armazenados em servidores oferecidos pelas empresas que oferecem o serviço da Nuvem.

Em outras palavras, podemos defini-la como um modelo de computação onde as capacidades relacionadas à tecnologia da informação são escaláveis e elásticas, sendo providas como um serviço para os usuários finais através da internet. São grandes repositórios de recursos virtualizados, como hardware, plataformas de desenvolvimento e software. Entre as diversas definições que podem ser encontradas, a computação em nuvem apresenta algumas características intrínsecas, como a virtualização de recursos, que possibilita a separação da infraestrutura dos recursos físicos, serviços sob demanda, que possibilita ao cliente obter maior ou menor quantidade de recursos de forma automática, independência de localização, que possibilita o acesso

aos recursos independente do ponto de acesso do usuário, elasticidade e escalabilidade, que possibilita disponibilizar ou remover recursos em tempo real e o aumento da capacidade de trabalho conforme a adição proporcional de recursos, medição dos serviços, que possibilita o custeamento apenas do que está sendo consumido, e repositório de recursos, que possibilita o armazenamento de dados.

Ela tem uma forte ligação com o conceito de virtualização e isso oferece-a diversas vantagens, como o baixo custo operacional e de consumo, a acessibilidade, a escalabilidade (maleabilidade quanto à quantidade de serviço desejado), a redução de riscos e despesas de manutenção e a redução de custos com hardware.

A nuvem pode ser dividida em modelos de serviço, sendo eles a infraestrutura como serviço (IaaS), a plataforma como serviço (PaaS) e o software como serviço (SaaS), em modelos de implantação, sendo eles a nuvem privada, pública e híbrida, e em camadas, sendo elas a camada de hardware, de infraestrutura, de plataforma e de aplicação.

Em uma rede de consumo dos serviços oferecidos com a Nuvem, podem ser observados três papéis de responsabilidade: o provedor, o consumidor e o desenvolvedor. O consumidor pode ser o usuário final ou uma organização e seu papel é utilizar os serviços oferecidos pelo provedor, não havendo a necessidade dele possuir conhecimento sobre computação em nuvem. O provedor disponibiliza os serviços e seu papel varia conforme o modelo de serviço oferecido; Caso seja de IaaS, o papel do provedor é manter o armazenamento, as filas de mensagens, a base de dados e hospedar o ambiente para máquinas virtuais; Caso seja de PaaS, o provedor gerencia a infraestrutura da Nuvem que mantém a aplicação, onde o usuário não tem acesso à essa infraestrutura; Por fim, caso seja de SaaS, o provedor instala, gerencia e mantém o software, onde o usuário tem acesso somente ao software executado. Já o papel do desenvolvedor é criar, publicar e monitorar os serviços da Nuvem, sendo ele geralmente contratado do provedor.

## B. Virtualização

Para entender computação em nuvem é fundamental entender o conceito de virtualização. Para a computação em nuvem, as duas principais formas de virtualização são: A virtualização de servidores e a virtualização de aplicativos.

A virtualização de servidor refere-se à instalação de várias máquinas virtuais em uma única máquina

física de servidor. Isso permite a redução no número de máquinas físicas, o que acarreta em menor gasto de energia, menos espaço necessário no datacenter, menos gasto com refrigeração e menos necessidade de manutenção.

Virtualização de aplicativo refere-se ao uso de um aplicativo em uma única máquina compartilhado simultaneamente com um grande número de usuários. Assim, cada usuário não precisa de uma máquina com as exigências que o aplicativo precisa para rodar e os custos da aplicação acabam sendo compartilhados por todos os usuários, o que gera uma aplicação final mais barata [1].

## C. Nuvem privada

A Nuvem privada se trata de um modelo de infraestrutura de Nuvem onde a mesma é alugada e operada exclusivamente por uma única empresa, podendo ser local ou remota. O que a diferencia dos outros modelos também é a sua restrição de acesso, sendo muitas vezes protegida por um firewall e sendo, portanto, muito mais segura que outros modelos de Nuvem, tendo o usuário um controle mais detalhado sobre a mesma com a desvantagem de um maior custo operacional [1].

## D. Nuvem pública

Na Nuvem pública, a infraestrutura pertence a uma organização que vende os serviços para o público e pode ser acessada por qualquer um que saiba sua localização, não possuindo restrições de acesso e sendo custeado unicamente pelo provedor, sendo completamente desprovidas de segurança [1].

## E. Nuvem híbrida

Uma Nuvem híbrida é formada por duas ou mais Nuvens dos modelos anteriores que podem se conectar através da internet para a transmissão de informações, obtendo as vantagens de ambos os modelos e sendo, porém, mais complexa de se operar e manter, sendo muitas vezes necessária a obtenção de serviços de mais de uma fonte [1].

## F. Infraestrutura como serviço

Uma Nuvem pode ser formada a partir de um modelo conceitual de prestação de serviços e estrutura, sendo o modelo mais comum formado por três camadas, sendo uma delas a Infraestrutura como Serviço, que se trata da camada de base e da

capacidade de um provedor de oferecer infraestrutura de processamento e armazenamento, tornando mais acessível o fornecimento desses recursos, oferecendo vantagens como a possibilidade dos clientes em ficarem em seus negócios ao invés do gerenciamento e manutenção da infraestrutura dos mesmos, previsão de custos futuros com ambiente de trabalho, garantia de evolução tecnológica dos recursos fornecidos, redução de paradas, redução de investimento em hardware e de custos em segurança e manutenção, a otimização de desempenho, a maior disponibilidade de espaço físico que seria ocupado pelo hardware e a flexibilidade em ampliar ou reduzir a capacidade de processamento ou armazenamento. Representa um modelo de ambiente mais completo e gerenciado e entre as suas características encontramos as Máquinas Virtuais, que se tratam de computadores físicos virtualizados, Discos Virtuais, que complementam as Máquinas Virtuais e oferecem, permanentemente, espaço de armazenamento variável, Regiões Geográficas, que se tratam de espaços físicos onde residem os recursos que alimentam os ambientes virtuais, Computação utilitária, que se trata do encapsulamento de recursos como processamento, largura de banda e armazenamento, a medição de serviços para cobrança adequada de utilização da Nuvem, plataformas de virtualização e conectividade com redes e internet [2].

#### G. Plataforma como serviço

Já a Plataforma como Serviço se trata da camada conceitual intermediária e do hardware virtualizado como serviço, oferecendo ambientes de desenvolvimento de software e facilitando a implementação de aplicações sem o custo e complexidade da compra e gerenciamento de hardware e software necessários para o desenvolvimento. Em suma, a diferença entre IaaS e PaaS se encontra no fato de que, no primeiro, o cliente gerencia seus próprios sistemas operacionais e dados e, no segundo, os clientes gerenciam suas aplicações. Entre as características da PaaS podemos citar a abstração, onde é oferecido ao usuário um ambiente ilimitado de recursos, automação, serviços na Nuvem, interface de usuário configurável, banco de dados configuráveis e controle sobre segurança e compartilhamento. Como vantagens podemos citar redução de custos com investimentos iniciais, evolução tecnológica do ambiente de trabalho, suporte ágil e transparente com os usuários, aumento da disponibilidade e segurança de dados [2].

#### H. Software como serviço

Por último, o Software como Serviço disponibiliza ao cliente aplicativos que são executados na Nuvem, disponibilizados a partir da web e que podem ser acessados em qualquer dispositivo, em qualquer local. Por compreender diversos aplicativos utilizados pelas pessoas em geral que utilizam a internet, se trata do modo mais comum de consumo e que pode ser observado em serviços como o Google Docs, Facebook e etc. As suas características incluem a arquitetura multi-cliente, aplicativos configuráveis, desenvolvimento e atualização mais rápidos, funcionalidades colaborativas, que permitem aos usuários colaborarem no desenvolvimento de projetos, disponibilidade abundante, modelo de licenças e gerenciamento completo pelo fornecedor. As vantagens incluem alta disponibilidade e interatividade, não existência de taxamento de licenças, transparência nas atualizações, que são responsabilidade do provedor, e a possibilidade de escalabilidade e especificidade das aplicações [2].

## II. COMPUTAÇÃO SEM SERVIDOR

Diferentemente do que propõe o nome, a computação sem servidor não dispensa o uso de servidores em seu funcionamento, mas retira do desenvolvedor a responsabilidade de configurar os recursos computacionais, sendo essa infraestrutura computacional completamente a cargo do provedor de serviços, que disponibiliza desde o básico até a escalabilidade de serviço [3]. A função sem servidor também é conhecida como Function Platform as a Service (FPaaS), o desenvolvedor não tem acesso direto aos recursos do servidor, pois estes são alocados automaticamente pelo provedor de serviço de nuvem para que seja feita a execução do código criado. Desse modo, a escalabilidade de aplicações na computação serverless é automática, ou seja, o provedor da nuvem, em seu gerenciamento de recursos, automaticamente libera ou restringe os recursos da aplicação de acordo com a demanda. Assim, é possível facilmente lidar com picos de demanda e aumentar a escalabilidade sem que ajustes manuais precisem ser feitos.

Em outras palavras, a maior vantagem se encontra no modo como proporciona a capacidade dos clientes ficarem exclusivamente no desenvolvimento do software. Ao se abstrair o uso da infraestrutura, a curva de aprendizado dos desenvolvedores referente ao uso de computação em nuvem é minimizada, visto que estes podem passar a se concentrar apenas na

lógica de suas aplicações. Consequentemente, a produtividade nesses cenários se torna maior e a entrega de novos produtos e aplicações se torna mais ágil.

Outra consequência é que a complexidade das aplicações diminui. Sem que servidores precisem ser gerenciados ou redes configuradas, é possível tornar os projetos mais simples para compreensão, o que é especialmente útil para pequenas e médias empresas onde a quantidade de colaboradores em um time é limitada. A robustez desse tipo de arquitetura também permite maior confiabilidade. O gerenciamento automático garante que o serviço estará sempre no ar, uma vez que falhas no hardware são contornadas sem nenhuma intervenção externa.

Dito isto, os casos de uso são os mais variados, considerando que essencialmente o que esta forma computação faz é mudar o lugar onde a parte principal do processamento de dados ocorre, é possível fazer praticamente qualquer coisa. Como exemplos temos aplicações web e mobile; processamento de dados em tempo real; integrações e automações, desenvolvimento de API's e funções focadas no backend e aplicações de machine learning, ou inteligência artificial.

No entanto, o modelo Serverless apresenta limitações devido ao fato de serem horizontalmente escaláveis e independentes de infraestrutura adjacente. Ou seja, sistemas que necessitam de dados persistentes devem armazená-los e retirá-los de outra fonte, como de um banco de dados, que podem ser fornecidos pelo mesmo fornecedor e que também deverá ser configurado. Entre outras desvantagens temos a precificação (o custo inclui a administração do sistema, embora que recursos não utilizados não sejam cobrados), a possibilidade do contratante ficar preso e dependente a um fornecedor, o ciclo de vida limitado, os gargalos nas entradas e saídas de dados, já que depende de outros serviços de infraestrutura e da transferência entre equipamentos, a comunicação entre dispositivos, já que esse modelo não pode ser diretamente endereçável na rede enquanto em execução, e a ausência de máquinas especializadas para a execução desse modelo [3].

Se tratando do aspecto comercial, ambos contratantes e fornecedores possuem responsabilidades. Por parte do fornecedor, a plataforma deve ser disponibilizada e assegurada por meio de um Service Level Agreement, bem como a escalabilidade do serviço sob os parâmetros definidos pelo cliente, modos de modificar, inserir ou monitorar o uso da função e segurança contra ataques virtuais ou

vírus. Por parte do cliente, este deve ter conhecimento da infraestrutura que sua aplicação demanda, da plataforma fornecida pelo fornecedor (tempo máximo de execução, limite de memória, custos, utilização de ferramentas e otimização) e de como tornar o ambiente seguro (exposição de informações, níveis de acesso, etc.).

Resumidamente, o modelo Serverless escala automaticamente, não havendo necessidade de ação por parte do cliente como subir novas máquinas em caso de aumento de demanda, não requer manutenção (instalação, monitoramento e atualização de softwares necessários), é cobrado somente pelo tempo que é utilizado, e é baseado em respostas a eventos que ocorrem no sistema e não há armazenamento incluso.

#### A. AWS Lambda

Plataforma Serverless da Amazon que faz parte do Amazon Web Services (AWS), sendo a mais utilizada atualmente é responsável por popularizar e moldar o ambiente de Serverless do mercado. As funções do Lambda são executadas em resposta a eventos, como requisições HTTP, e usam um serviço chamado Firecracker para criar máquinas virtuais leves. Possui alta compatibilidade com os outros serviços oferecidos pela AWS, além de uma vasta documentação da própria empresa e de seus usuários.

O lambda também é complementado pelo Lambda@Edge, que oferece suporte semelhante às funções lambda para executar funções com latência extremamente baixa usando roteamento otimizado por borda, e pelo GreenGrass, que permite que as funções sejam executadas em dispositivos conectados. É um serviço pago.

#### B. Azure Functions

Plataforma Serverless da Azure, fornecida pela Microsoft, sendo a segunda mais utilizada. Ela oferece um subproduto chamado *durable functions* que permite escrever funções com estado dentro de um ambiente serverless, funcionando como um checkpoint entre duas funções e possuindo restrições de linguagem e serviços suportados. Possui alta compatibilidade com outros serviços da Microsoft Azure. É um serviço pago.

#### C. Google Cloud Functions

Plataforma Serverless da Google Cloud que provém computação local ou em nuvem, não

necessitando de provisionamento de recursos e se integrando facilmente aos outros recursos da empresa. Ele usa framework de funções como um serviço de código aberto, evitando dependências de um único fornecedor.

Recentemente foi disponibilizado o Cloud Functions de Segunda Geração, que oferece recursos como simultaneidade de instâncias, processando até mil solicitações simultâneas em uma única instância, reversões rápidas, já que uma revisão é criada a cada alteração, processamento de solicitação seis vezes mais longo, que permite executar funções por até uma hora, instâncias quatro vezes maiores, com até 16gb de RAM e 4 vCPUs para utilização, instâncias pré-aquecidas que garantem que o tempo de inicialização do aplicativo não interfira em seu desempenho, mais regiões e extensibilidade e portabilidade. É um serviço pago.

### III. ESTADO DA ARTE

Estudantes da Universidade Federal de Santa Catarina criaram um projeto de conclusão de curso demonstrando a computação em nuvem sem servidor através de uma aplicação nos serviços da AWS, possibilitando a utilização do machine learning para automatizar um processo de batimetria, que mede a profundidade de colunas d'água, economizando trabalho manual que, além de limitado, gera mais custos [11].

Lima, estudante do Instituto Federal do Espírito Santo, demonstra em seu projeto como o Serverless pode ser utilizado para a construção de Api 's de jogos para o desenvolvimento dos mesmos, o que se demonstrou eficaz no aumento da eficiência de desenvolvimento e redução de custos [12].

Igor, estudante da universidade Federal de Uberlândia, desenvolveu uma aplicação mobile com todo o processamento de backend sendo feito na nuvem da AWS usando o Serverless e apresentou como esta aplicação resultou em benefícios de confiabilidade, escalabilidade e boa experiência de usuário [13].

Esses projetos demonstram as vantagens que a computação Serverless pode apresentar e o motivo da sua crescente demanda nos dias de hoje, principalmente por parte de empresas.

### IV. DESAFIOS PARA O FUTURO

No que se refere à computação sem servidor, existem ainda perguntas não respondidas e que não estão sendo muito exploradas. Por exemplo:

Quais são os limites da computação sem servidor? Ela se limita apenas a FaaS? Qual seu relacionamento com SaaS e MBaaS?

A quantidade de código feita fora de ambientes sem servidor ainda é grande. Caso este código precise ser adaptado para rodar no formato sem servidor, isso exigirá um investimento de horas de trabalho de desenvolvedores para adequar o código, tornando esta uma importante questão.

Um grande desafio para esse modelo de computação em nuvem é a redução de custo, isto é, diminuindo a quantidade de recursos gasto por uma função, seja ela rodando ou em espera. O formato da computação sem servidor acaba sendo mais efetivo para aplicações CPU-bound do que aplicações I/O bound, visto que estas ainda consomem recursos quando estão esperando pela entrada do usuário.

Um ponto essencial para esse modelo de computação em nuvem é a possibilidade de escalar a zero, isto é, de reduzir o gasto de recursos quando a aplicação estiver em espera. Isso implica no problema de cold start, que é o tempo necessário para colocar a aplicação para rodar quando é chamado após um determinado período de tempo

Fornecer escalabilidade e elasticidade aos usuários também é fundamental, o que inclui prever a demanda que será necessária para a aplicação com o conhecimento mínimo de seu funcionamento.

Além disso, outros temas que precisam ser abordados para o futuro são o gerenciamento de dependências, pois a atualização dos processos e das funções terá dependências de bibliotecas e serviços de terceiros que podem desencadear problemas de compatibilidade; a limitação do tempo de execução, pois há limitações na criação e manutenção de códigos e scripts usados nas funções de computação serverless; e segurança, pois ainda há vulnerabilidades que podem permitir acessos ou vazamentos indevidos.

Com isso em mente, é importante salientar que estes temas serão abordados no futuro para que o aprimoramento do serverless seja feito e que com ele possamos criar e desenvolver com mais agilidade, eficácia e segurança.

Quando falamos de tendência, é esperado que a computação serverless se integre mais com outras tecnologias como contêineres, micro serviços e até

mesmo aprendizado de máquina para que se possa criar aplicações cada vez mais robustas.

## V. METODOLOGIA

A metodologia utilizada no presente trabalho foi realizada através de uma pesquisa bibliográfica utilizando textos, livros e artigos científicos. Ademais, foi realizada a construção de uma aplicação Api Rest escrita na linguagem Java utilizando a arquitetura MVC com versionamento de código no Github e containerização com Docker, afim de simular o processo de alocação dinâmico de recursos de plataformas Cloud em aplicações sem servidor. Para isso foi desenvolvido um vídeo (Anexo 1) para introduzir, explicar e testar a aplicação em ambiente local.

## VI. RESULTADOS

A partir da aplicação desenvolvida e os testes realizados em conjunto com o referencial teórico utilizado neste trabalho, em ambiente local 4 endereços de requisição de dados foram expostos, no método POST foi garantido uma simulação da alocação de recursos dinâmicos realizados por uma Paas - Platform as Service como a AWS em uma implementação de um serviço sem servidor de maneira que, assim que a Cloud é utilizada ocorre uma alocação dinâmica de recursos de processamento de mensagens em um determinado período de tempo e assim que o serviço finaliza seu trabalho não existirão mais recursos alocados até que exista novamente a necessidade.

Na sequência, para uma melhor visualização dos recursos alocados em testes da aplicação os 3 endereços de requisições GET fornecem ao usuário um relatório completo de todos os serviços de todas as contas, de todos os serviços de uma conta específica ou de serviços ativos em tempo real de uma conta específica, de maneira que plataformas reais de cloud são capazes de utilizar destas informações para converter a utilização de sua própria plataforma em cobranças financeiras para seus clientes.

O primeiro endereço de requisição POST <http://localhost:8080/resources> precisa dos seguintes elementos para ser utilizado: conta, quantidade de mensagens e serviço, não sendo necessário seguir esta ordem, mas todas as chaves devem estar presentes, em formato JSON como na imagem a seguir:

```
1 {  
2   "conta": "joao",  
3   "servico": "compras",  
4   "quantidadeMensagens": 1000  
5 }
```

Fig. 1. exemplo de payload POST

A chamada da aplicação nesta etapa se assemelha a uma aplicação qualquer de uma empresa que precisa que os recursos de uma Cloud Serverless sejam alocados naquele mesmo instante para processar as informações, sendo assim, por sua vez, o retorno desta chamada garante ao usuário um registro de chamada, informando a conta, o nome do serviço, a quantidade de núcleos alocados para processamento realizado pela CLOUD e uma mensagem de retorno, indicando serviço criado ou atualizado.

```
1 {  
2   "account": "joao",  
3   "service": "compras",  
4   "allocatedCores": 1,  
5   "message": "Serviço criado"  
6 }
```

Fig. 2. retorno do método POST

Conforme Fig.2 acima, a mensagem de serviço criado indica que naquele instante de chamada dos serviços da CLOUD o usuário não possuía o mesmo serviço sendo processado, pois caso essa condição fosse atendida, a plataforma CLOUD não realizaria a alocação de recursos para processamento em outro endereço, utilizaria aquela mesma chamada que estava em processamento para incluir as novas mensagens a serem processadas, garantindo um retorno de Serviço Atualizado.

Os demais endereços de requisição GET garantem relatórios de utilização de recursos para fins de monitoração, com o principal objetivo de simular um registro da própria plataforma CLOUD de como seus próprios recursos foram utilizados, garantindo uma possibilidade de utilizarem estas informações das mais diversas maneiras: campanhas de publicidade, região com maior utilização, sistema de fatura, etc.

Todas as informações do método POST são guardadas em um banco de dados MySQL que por sua vez garante os seguintes relatórios:

```
GET ▼ http://localhost:8080/resources
```

Fig. 3. método GET todos os recursos

Na Fig.3 é possível verificar o endereço do relatório mais generalista da aplicação, devolvendo todos os recursos alocados de todas as contas.

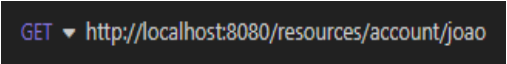


Fig. 4. método GET recursos por conta

A Fig.4 apresenta um relatório mais específico, ideal para informações expostas em cada sessão de usuário, garantindo as informações de todos os recursos alocados por uma conta específica, como neste caso a conta do “joao”, porém, qualquer conta pode ser informada ao final do endereço.

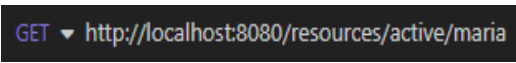


Fig. 5. método GET recursos ativos por conta

O último endereço, apresentado na Fig.5 devolve ao usuário um relatório ainda mais específico que os recursos por conta, este por sua vez garante um relatório dos recursos ativos por conta, ou seja, levando em consideração que cada chamada do método POST possui um tempo diferente de processamento de acordo com a quantidade de mensagens, neste método o usuário recebe um retorno variado de acordo com aquilo que está sendo processado na conta dele naquele instante da chamada, garantindo uma monitoração instantânea da alocação de recursos.

## VII. CONCLUSÃO

Com a criação de uma aplicação simples foi possível demonstrar além da alocação dinâmica de recursos de uma plataforma cloud, também garante ao usuário uma visão dos desenvolvedores da própria plataforma e como ela registra cada requisição da alocação de seus próprios recursos em bancos de dados e a partir disso tomam as decisões internamente na empresa, expondo ao usuário seus serviços ativos, devolvendo todos os recursos alocados por uma conta em um período variado de tempo, além de gerar faturas e analisar as próximas direções que a própria plataforma seguirá.

Com o estudo bibliográfico é esperado que o conhecimento sobre a modalidade serverless seja mais difundido, bem como suas vantagens, suas desvantagens, como esse tipo de tecnologia se encontra hoje em dia e seus desafios atuais. Em especial, é esperado instigar a curiosidade e o

interesse nesse tema para que outras pesquisas e estudos sejam feitos e o que é discutido para o futuro possa se aproximar da realidade cada vez mais rápido.

Como foi visto, existem diversas formas de computação em nuvem, bem como diversos conceitos no qual esta é fundamentada. Essa modalidade permite a criação de diversas aplicações que não seriam possíveis sem ela, bem como reduz custos, aumenta escalabilidade, agiliza desenvolvimento e até mesmo aumenta a confiabilidade. Assim, são concretizados diversos motivos para que seu uso e popularidade sejam amplamente elevados.

Mesmo assim, nenhuma tecnologia está livre de desafios e é importante manter estes em vista quando pensamos em características como segurança, tempo de inicialização, previsibilidade de custos ou o próprio controle sobre o ambiente de execução. Os desafios pela frente são inúmeros e cada vez mais se buscará minimizar seus pontos fracos e tornar suas vantagens cada vez mais fortes.

## VIII. ANEXOS

Anexo 1 - *Demonstração projeto prático - simulação de alocação de recursos de uma cloud serverless:*  
<https://youtu.be/Sf-8Md4gVGO>

## REFERÊNCIAS

- [1] Esférica Tecnologia, “Computação em nuvem”: <https://www.esferica.com.br/wp-content/uploads/2020/12/ebook-Cloud-Computing-v1.0-Capa.pdf>
- [2] B. Hélder, N. José, S. Bruno e M. Antonio, “Computação em nuvem”: <https://livroaberto.ibict.br/bitstream/1/861/1/COMPUTA%C3%87%C3%83O%20EM%20NUVEM.pdf>
- [3] V. André, P. Gustavo, F. Jean, D. Lucas, P. Racyus, V. Marcos, V. Luiz e N. José. “Computação Serverless: Conceitos, Aplicações e Desafios”: <https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/50/232/469-1?inline=1>
- [4] G. Shaji e G. Hovan, “Serverless Computing: the Next Stage in Cloud Computing's Evolution and an Empowerment of a New Generation of Developers”: [https://www.researchgate.net/publication/350580133\\_Serverless\\_Computing\\_the\\_Next\\_Stage\\_in\\_Cloud\\_Computing's\\_Evolution\\_and\\_an\\_Empowerment\\_of\\_a\\_New\\_Generation\\_of\\_Developers](https://www.researchgate.net/publication/350580133_Serverless_Computing_the_Next_Stage_in_Cloud_Computing's_Evolution_and_an_Empowerment_of_a_New_Generation_of_Developers)
- [5] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski e P. Suter. “Serverless Computing: Current Trends and Open Problems”: [https://www.researchgate.net/publication/322092289\\_Serverless\\_Computing\\_Current\\_Trends\\_and\\_Open\\_Problems](https://www.researchgate.net/publication/322092289_Serverless_Computing_Current_Trends_and_Open_Problems)
- [6] J. Wen, Z. Chen, Y. Liu, Y. Lou, Y. Ma, G. Huang, X. Jin e X. Liu. “An Empirical Study on Challenges of Application Development in Serverless Computing”: [https://chenzhenpeng18.github.io/papers/FSE21\\_2.pdf](https://chenzhenpeng18.github.io/papers/FSE21_2.pdf)
- [7] H. Augusto, “Computação sem servidor: tecnologia para criar, consumir e integrar aplicativos em nuvem”: <https://blog.qinetwork.com.br/computacao-sem-servidor/>

- [8] Amazon Web Services, “Otimização da economia empresarial com arquiteturas sem servidor”:  
[https://d1.awsstatic.com/whitepapers/pt\\_BR/optimizing-enterprise-economics-serverless-architectures.pdf](https://d1.awsstatic.com/whitepapers/pt_BR/optimizing-enterprise-economics-serverless-architectures.pdf)
- [9] B. Violino, “VEJA COMO E ONDE USAR A COMPUTAÇÃO SEM SERVIDOR”:  
<https://www.netinstruments.com.br/blog/veja-como-e-onde-usar-a-computacao-sem-servidor>
- [10] A. Khonsari, P. Mousavi e H. Shafiei. “Serverless Computing: A Survey of Opportunities, Challenges, and Applications”:  
<https://arxiv.org/pdf/1911.01296.pdf>
- [11] S. Ricardo, “Implementação de uma API em arquitetura serverless no ambiente da AWS para predição de batimetria em estuários através da aplicação de métodos de aprendizado de máquina em imagens multiespectrais de satélites”:  
[https://repositorio.ufsc.br/bitstream/handle/123456789/223655/ricardo\\_santacatarina\\_tcc.pdf?sequence=1&isAllowed=y](https://repositorio.ufsc.br/bitstream/handle/123456789/223655/ricardo_santacatarina_tcc.pdf?sequence=1&isAllowed=y)
- [12] L. Gustavo, “CAPI : UMA ABORDAGEM DE DESENVOLVIMENTO USANDO TECNOLOGIAS SERVERLESS”:  
<https://repositorio.ifes.edu.br/bitstream/handle/123456789/1810/TC-C%20FINAL.pdf?sequence=1&isAllowed=y>
- [13] S. Igor, “Aplicação mobile Serverless”:  
<https://repositorio.ufu.br/bitstream/123456789/28573/1/Aplicacao-MobileServerless.pdf>