FICHA TÉCNICA

Revista Ubiquidade, ISSN 2236-9031, v. 4, n. 2, jul./dez. 2021

Capa: Larissa Conelheiro Kovelis

Editoração e Diagramação: Prof. Dr. Juliano Schimiguel

Revisão Textual: Regiane Maria Pankoski

Editora: UNIANCHIETA

REVISTA UBIQUIDADE

data de publicação Dezembro/2021

EXPEDIENTE

A revista Ubiquidade é uma publicação semestral vinculada ao Curso de Bacharelado em Ciência da Computação do UniAnchieta, exclusivamente eletrônica, que pretende divulgar contribuições originais, teóricas ou empíricas, relacionadas às áreas de Tecnologia de Informação e Comunicação (TICs) e está aberta para trabalhos científicos de pesquisadores nacionais ou internacionais.

O envio de trabalhos para apreciação, assim como o pedido de informações, pode ser feito por meio do endereço: ubiquidade@anchieta.br.

PREFÁCIO

Neste número V.4, N.2 (2021) - Ago/Dez, da Revista Ubiquidade, apresentamos artigos com temas relevantes e interessantes para a comunidade científica e acadêmica, focado em áreas relacionadas às TICs - Tecnologias de Informação e Comunicação. Este número traz artigos de pesquisadores importantes, de instituições como Centro Universitário Padre Anchieta (Unianchieta), Universidade Cruzeiro do Sul, Universidade Cidade de São Paulo (Unicid), Faculdade Fernão Dias, etc.

No artigo "Computação Desplugada aplicada no Ensino Fundamental I", André Souza e Juliano Schimiguel mostram como podemos melhorar o ensino básico de forma dinâmica com a Computação Desplugada, dando alternativas de como ensinar a computação sem o uso de computadores ou tecnologias, melhorando a interatividade dos professores com os alunos e sem necessitar de maiores investimentos da instituição de ensino.

O artigo “Análise da Plataforma AWS Lambda em Ambientes de Nuvens Computacionais”, de Yasmin Oliveira Pisoni e Carlos Eduardo Câmara, busca preencher a lacuna em relação a estudos que abordem a *serverless computing*, focando principalmente na plataforma AWS Lambda. Foi observado que o AWS Lambda elimina a complexidade de lidar com servidores, além de apresentar um modelo de faturamento de pagamento mediante solicitação, em que são eliminados os custos da capacidade computacional considerada ociosa.

O artigo “Aplicando Arquitetura de Microsserviços no Desenvolvimento de Software”, de Roni da Cruz Silva e Rodrigo Kiyoshi Saito, analisou os princípios da Arquitetura de Microsserviços e como ela pode ajudar a resolver a necessidade de construir sistemas que possam ser escalados, que tenham tolerâncias a falhas, que possam implementar mais de uma linguagem de programação e que se adaptem a evolução constante que vivemos atualmente.

O artigo “Um Estudo aplicado à Segurança de Aplicações Web”, de Felipe Delboni Ortega e Carlos Eduardo Câmara, apresenta um estudo teórico orientado à segurança de aplicações Web, a partir da análise de alguns de seus principais pontos de vulnerabilidade, permitindo seu entendimento, a mitigação e uma visão compreensiva da importância de uma estratégia de desenvolvimento seguro, baseada em estudos analíticos e metodologias de mercado.

Finalmente, o artigo “Xô, Dengue! Um Aplicativo de Apoio ao Combate do Mosquito Aedes Aegyptis”, de Fernando Tamataya, Lucas Fermiano Teixeira, Juliano Schimiguel e Carlos Adriano Martins, descreve um aplicativo para dispositivos móveis, com apoio ao ambiente WEB, visando o combate ao mosquito Aedes Aegypti e suas doenças. O aplicativo foi construído utilizando-se do framework Android Java, com funcionalidade compatível com dispositivos móveis e também em ambiente Web, Java e tecnologias diversas, tendo múltiplas funções.

Prof. Dr. Juliano Schimiguel
Coordenador da Revista Ubiquidade

SUMÁRIO

COMPUTAÇÃO DESPLUGADA APLICADA NO ENSINO FUNDAMENTAL I (*André SOUZA, Juliano SCHIMIGUEL*) 6

ANÁLISE DA PLATAFORMA AWS LAMBDA EM AMBIENTES DE NUVENS COMPUTACIONAIS *(Yasmin Oliveira PISONI, Carlos Eduardo CÂMARA)* 34

APLICANDO ARQUITETURA DE MICROSSERVIÇOS NO DESENVOLVIMENTO DE SOFTWARE (*Roni da Cruz SILVA, Rodrigo Kiyoshi SAITO*) 59

UM ESTUDO APLICADO A SEGURANÇA DE APLICAÇÕES WEB *(Felipe Delboni ORTEGA, Carlos Eduardo CÂMARA)* 87

XÔ, DENGUE! UM APLICATIVO DE APOIO AO COMBATE DO MOSQUITO AEDES AEGYPTIS *(Marco Aurélio Fontes VENÂNCIO, Fernando TAMATAYA, Lucas Fermiano TEIXEIRA, Juliano SCHIMIGUEL, Carlos Adriano MARTINS)* 130

COMPUTAÇÃO DESPLUGADA APLICADA NO ENSINO FUNDAMENTAL I

UNPLUGGED COMPUTING APPLIED IN ELEMENTARY EDUCATION I

André Souza

souza.andre0408@gmail.com

Ciência da Computação, Centro Universitário Padre Anchieta, Jundiaí/SP

Prof. Dr. Juliano Schimiguel

schimiguel@gmail.com

Ciência da Computação, Centro Universitário Padre Anchieta, Jundiaí/SP; e Universidade Cruzeiro do Sul, São Paulo/SP

Resumo

A educação básica é considerada um dos pilares mais importantes para transformação social de uma pessoa. No Brasil destaca-se 3 etapas importantes da educação, sendo elas: educação infantil, ensino fundamental e o ensino médio. O ensino fundamental I serve como base para as demais etapas que complementam a estrutura de aprendizado dos alunos por prepararem os mesmos a dominarem as leituras, os cálculos e as escritas. Também na educação observamos o pensamento computacional que veio para contribuir com a adaptação dos alunos na sala de aula, com base nisso surge como forma de estratégia principalmente para escolas públicas com pouca estrutura a computação desplugada que consiste em ensinar os alunos de forma simples e objetiva sem a necessidade da tecnologia e um meio externo. O objetivo desse estudo é mostrar como podemos melhorar o ensino básico de forma dinâmica com a Computação desplugada, nos dando alternativas de como ensinar a computação sem o uso de computadores ou tecnologias, melhorando a interatividade dos professores com os alunos e sem necessitar de maiores investimentos da instituição. Por meio disso, foi realizado um estudo para identificar a capacidade das crianças em desenvolver o aprendizado sem um meio externo. O estudo foi realizado a domicílio com 10 crianças do ensino fundamental I, sendo elas 6 meninas e 4 meninos, a metodologia aplicada foram baseados em exercícios da computação desplugada. Para composição dos exercícios 5 cartões foram feitos a mão, numerados com seus respectivos valores: 1, 2, 4, 8, 16. Complementando o notebook foi utilizado para leitura das atividades, por fim, as respostas foram escritas pelos alunos em uma folha sulfite. Dentre as 10 crianças todas conseguiram resolver as atividades, “Trabalhar com números binários” e “Enviar mensagens secretas”, porém o desafio somente um aluno conseguiu resolver, ao fim foram feitas 5 questões com o intuito de obter informações e características sobre os alunos.

Palavras-Chaves

Computação; Educação; Tecnologia; Ensino fundamental;

Abstract

Basic education is considered one of the most important pillars for a person's social transformation. In Brazil, there are 3 important stages of education, namely: kindergarten, elementary school, and high school. Elementary school I serves as the basis for the other stages that complement the students' learning structure by preparing them to master reading, calculating and writing. Also in education, we observe the computational thinking that came to contribute to the adaptation of students in the classroom, based on this, unplugged computing, which consists of teaching students in a simple and objective way, emerges as a strategy mainly for public schools with little structure. without the need for technology and an external means. The objective of this study is to show how we can dynamically improve basic education with unplugged computing, giving us alternatives on how to teach computing without the use of computers or technologies, improving the interactivity of teachers with students and without requiring major investments of the institution. Through this, a study was carried out to identify the ability of children to develop learning without an external means. The study was carried out at home with 10 children from elementary school I, 6 girls and 4 boys, the applied methodology was based on unplugged computing exercises. To compose the exercises, 5 cards were made by hand, numbered with their respective values: 1, 2, 4, 8, 16. Complementing the notebook was used to read the activities, finally, the answers were written by the students on a bond sheet . Among the 10 children all managed to solve the activities, "Working with binary numbers" and "Send secret messages", but the challenge only one student managed to solve, in the end, 5 questions were asked in order to obtain information and characteristics about the students.

Keywords

Computing; Education; Technology; Elementary School;

INTRODUÇÃO

A educação tem como forma de abranger os processos formativos que se desenvolvem na vida familiar, na convivência humana, no trabalho, nas instituições de ensino e pesquisa, nos movimentos sociais e organizações da sociedade civil e nas manifestações culturais.

No Brasil, a educação básica é formada por três grandes etapas: a educação infantil, o ensino fundamental e o ensino médio. De acordo com a Lei de Diretrizes e Bases da Educação, a educação básica é obrigatória a partir dos quatro anos de idade e o estado libera o ensino na iniciativa privada, desde que sejam atendidas as condições estabelecidas na legislação. É durante o período de vida escolar que se faz necessário os conhecimentos mínimos necessários para uma cidadania completa.

No ensino infantil encontra-se as crianças de 0 a 5 anos de idade, nessa fase se dá início ao primeiro contato com a escola sendo de estrema importância, onde a criança aprende as coisas de forma lúdica, por meio do aspecto cognitivo, físico, motor e psicológico.

O ensino fundamental I funciona como uma base para as demais etapas da formação educacional, onde prepara o estudante para domínio de leitura, cálculos e escritas. Ao longo desta etapa os alunos começam a aprender os conceitos educacionais assuntos que os guiarão durante toda a educação básica, como alfabetização, independência e responsabilidade, e também aprendem a ler e escrever. Esse processo de alfabetização permite que os estudos se tornem mais complexos e que as crianças ampliem a sua visão do mundo.

Já no ensino médio ele tem a duração de 3 anos e tem por objetivo principal preparar os jovens para os vestibulares, além disso, trabalha o auto-conhecimento, autonomia intelectual e pensamento crítico.

Dentro da educação observamos o pensamento computacional que veio para contribuir com a adaptação da sala de aula às necessidades da sociedade contemporânea. A transformação digital surgida com as novas tecnologias exige que os indivíduos estejam cada vez mais aptos a lidar com as ferramentas e linguagens da cultura digital. (Noemi, Debora 2020).

Como meio de estratégia encontramos a computação desplugada definida por uma metodologia que proporciona o aprendizado dos conceitos computacionais de forma simples e interativa, sem a utilização de hardware ou software. Como observamos nos estudos há um crescimento da estratégia no ensino em relação a computação desplugada, "sendo uma alternativa a ser considerada para a inclusão de conceitos de computação e do pensamento computacional dos alunos, principalmente aqueles que estudam em escolas que não possuem estrutura básica para o ensino da computação através do uso dos computadores". (Santos et al. 2017).

Em relação aos computadores nem sempre foram da forma como são hoje, acontece que hoje, em meio a smartphones, notebooks e desktops, é difícil pensar que, nos anos 1940, computadores não eram uma caixinha conectada a um monitor, mouse e teclado. Eles estavam mais para caixas gigantescas cheias de cabos, capacitores e resistores. Os computadores de hoje são mais simples de operar. A nível mais baixo, o seu processador continua funcionando com instruções 0 e 1, cabe ao sistema operacional traduzir tudo o que você faz para essa linguagem binária que é passada para o computador, que por sua vez pode fazer o que você quer.

Portanto ao analisar a realidade em que vivemos hoje em dia, pode-se notar que em nossas escolas públicas é nítido a ausência de laboratórios estruturados de Computação, com raríssimas exceções. Dessa forma, compreendemos que, para se conseguir um trabalho proveitoso no ensino, se faz necessário buscar alternativas viáveis, como o método da "Computação Desplugada" (Bell, T., Witten, I. H., Fellows, M., Adams, R., & McKenzie, J.. 2011).

Sendo assim a dificuldade de aprendizagem sem a tecnologia em locais de baixa renda despertou interesse em aprofundar os conhecimentos sobre o assunto, confirmando que com pouco pode-se obter grandes resultados e consequentemente despertando interesses e ate paixão pela tecnologia.

Desta forma, o presente estudo teve por objetivo mostrar que é possível ensinar computação sem o uso de equipamentos caros, como computadores, celulares, notebook, entre outros. Nota-se também o engajamento dos alunos em conhecer como um computador pensa que é algo que usamos todos os dias em praticamente tudo e a todo momento.

O artigo foi dividido em sessões para facilitar o entendimento do leitor, afim de obter o maior proveito da leitura sobre o estudo gerido.

Referencial Teórico

Pensamento Computacional

O pensamento computacional é o processo de entender aspectos da computação em nosso mundo e aplicar ferramentas e técnicas para facilitar sistemas e processos. Na escola, pode ser exemplificado quando alunos resolvem problemas, dividindo-os em parte e utilizando a lógica. Esse conceito compreende a habilidade crítica, estratégica e criativa, utilizando os fundamentos da área da

computação em diferentes áreas da vida. Assim, seja individualmente, seja em grupo, o(a) aluno(a) consegue pensar racionalmente e resolver questões. Os alunos educados com a colaboração do pensamento computacional conseguem melhoras no desenvolvimento cognitivo e trabalham algumas habilidades importantes, dentre elas raciocínio lógico, capacidade de aprendizado, planejamento, resolução de problemas e autonomia.

Computação desplugada

A computação desplugada é uma técnica que consiste em ensinar os fundamentos da Computação, através de atividades, sem o uso do computador. Tais atividades têm despertado o interesse de professores e pesquisadores, e tem sido empregada em diversos países ao redor do mundo. [BELL et al, 2011 apud VIEIRA et al, 2003]. A Computação Desplugada é aplicada a partir de uma sequência de atividades que não possuem ligações com a tecnologia, sendo executadas de uma forma dinâmica e didática que se propõe a desenvolver o raciocínio lógico e rápido do indivíduo ao qual está passando por este processo de aprendizagem, logo o mesmo é estimulado a adquirir habilidades de resolução de problemas chamada de Pensamento Computacional, ao qual se caracteriza como competência inicial, obtida a partir desse procedimento. Na atualidade em um mundo cercado por tecnologia, torna-se cada vez mais raro encontrar material educacional lúdico e de fácil aplicação em escolas. [BELL et al, 2011]. Tim Bell, Ian H. Witten e Mike Fellows desenvolveram o livro "Computer Science Unplugged" como forma de amenizar a falta de conteúdos lúdicos para o ensino da computação que tem o intuito de ensinar fundamentos da computação através de uma coleção de atividades a serem aplicadas de forma desplugada. As atividades são baseadas em conceitos matemáticos e lógicos que possui uma abordagem lúdica, desenvolvendo o pensamento computacional.

Muitos dos nossos problemas são resolvidos através do raciocínio lógico, os computadores e jogos de vídeo game funcionam da mesma maneira. Porém, se nós, seres humanos, não estimularmos a mente durante a vida, acabamos perdendo essa capacidade. Essa estratégia propõe estimular seu raciocínio através de jogos e dinâmicas divertidas. O objetivo é apresentar algumas atividades lúdicas longe das telas dos computadores, para despertar o aumentar o seu interesse pela computação. Com a computação desplugada sendo uma estratégia que traz resultados podemos perceber que o ensino da computação sem uso de computadores através de atividades lúdicas que simulam o funcionamento do computador torna o ensino da computação prazeroso e promissor.

Lógica computacional

Pode ser entendida como a relação entre a lógica formal, mais tradicional, e a computação. Trata-se de um modo de pensar e estruturar ideias para escrever um software ou um algoritmo.

A lógica formal estuda como provar a veracidade de algum argumento e as relações entre diferentes ideias, ao passo que a computacional estuda como organizar instruções e criar um raciocínio baseado em pequenas etapas para chegar a uma solução.

Ambas estão ligadas ao pensamento computacional, que, por sua vez, não necessariamente está associado a contextos puramente tecnológicos. Essa estratégia tem como objetivo preparar alguém para resolver um problema, independentemente do seu escopo.

No dia a dia, enfrentamos uma série de situações complexas, em que temos que organizar pensamentos e chegar a uma solução ordenada, sendo que o raciocínio lógico e o pensamento estruturado viabilizam essa etapa. Afinal, esses conceitos ajudam a separar o que é razoavelmente possível do que não faz sentido.

Assim, você consegue avançar na solução sem se preocupar com aspectos que não serão úteis. O pensamento computacional, que utiliza a lógica como base, pode ser dividido em quatro etapas:

- **decomposição do problema**: quebrar a situação em pequenas partes, seguindo a ideia do "dividir para conquistar";

- **reconhecimento de padrões**: separar categorias e identificar soluções específicas para cada uma, considerando as limitações e as características de cada grupo;

- **abstração**: dividir o problema em questões mais simples e profundas, enxergando além;

- **algoritmo**: criar uma série de etapas para a resolução de um problema.

Essa separação em fases ajuda a pessoa a organizar suas ações e seu esforço, bem como o tempo para a resolução de um problema. Desse modo, é possível alcançar a eficiência e conseguir resultados mais claros.

Número Binário

O sistema de número binário utiliza apenas os algarismos 0 e 1 para representação das coisas. A primeira contagem binária registrada é do século 3 a.C., feita por um matemático indiano. Desde então, o sistema jamais deixou de ser estudado, mas só em 1937 foi usado pela primeira vez, da maneira que vemos hoje, nos circuitos digitais.

É utilizado em várias coisas atualmente, como em máquinas com circuitos digitais para interpretar informações e executar ações. Essa é a linguagem responsável por fazer o computador entender e assim exibir e processar textos, números e imagens, por exemplo. Destaca-se que o computador não consegue entender como nós humanos entendemos. Ele só lê sinais elétricos na sua forma mais simples: sem corrente ou com corrente, representados respectivamente pelos números 0 e 1, assim como explica o engenheiro de software Eugeni Dodonov.

A partir disso, consegue-se entender como funciona esse sistema: todos os comandos e dados processados pelo equipamento são formados por sequências desses algarismos. O branco puro na tela, por exemplo, equivale a 11111111 em código binário e o número 8, para o computador, é 1000.

Número Hexadecimal

É um outro sistema de numeração, assim como o sistema binário. O Hexadecimal é muito utilizado na programação de microprocessadores, em especial nos equipamentos e máquinas de estudo e sistemas de desenvolvimento. Trata-se de um sistema de numeração posicional que representa os

números em base 16, sendo assim, utilizando 16 símbolos. Este sistema utiliza os símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9 do sistema decimal, além das letras A, B, C, D, E e F. Este sistema é muito utilizado para demonstrar números binários de uma forma mais compacta, visto ser muito mais fácil converter hexadecimal em binários e vice-versa.

Na figura abaixo, está demonstrado como funciona os dois sistemas quando comparados ao sistema mais utilizado: o decimal. Dessa forma, conseguimos visualizar algumas vantagens de cada sistema, como o Hexadecimal ser mais fácil para representar números maiores do que os outros dois sistemas, uma vez que utiliza apenas um símbolo, como F.

BINARIOS	DECIMAL	PALAVRA
00000	0	
00001	1	A
00010	2	B
00011	3	C
00100	4	D
00101	5	E
00110	6	F
00111	7	G
01000	8	H
01001	9	I
01010	10	J
01011	11	K
01100	12	L
01101	13	M
01110	14	N
01111	15	O

BINARIOS	DECIMAL	PALAVRA
10000	16	P
10001	17	Q
10010	18	R
10011	19	S
10100	20	T
10101	21	U
10110	22	V
10111	23	W
11000	24	X
11001	25	Y
11010	26	Z
11011	27	?
11100	28	!
11101	29	.
11110	30	,
11111	31	;

Figura 1: Representação Sistemas Binário, Decimal, Hexadecimal

Número Decimal

Os números decimais são caracterizados por ter uma parte inteira e uma parte decimal separadas por uma vírgula. De modo geral, dizemos que números decimais não são inteiros, pois eles representam quantidades “quebradas”, ou seja, partes fracionadas de algo inteiro.

Os números decimais têm como principal característica a presença da vírgula. Assim como os números inteiros, os decimais também utilizam o sistema de numeração decimal, ou seja, podemos diferenciar os números pela posição em que os algarismos se encontram.

Os números decimais aparecem com frequência em nosso dia a dia, como ao realizar compras em um supermercado ou abastecer um carro. Assim, é importante entender como funciona o sistema de posição e, consequentemente, a nomenclatura desses números. Analisando o número 1,9960:

1 - Parte inteira;

9 - Décimos;

9 - Centésimos;

6 - Milésimos;

0 - Décimo de Milésimos;

Abaixo podemos ver com mais facilidade e a sequência das casas decimais que se segue.

Valor	Nome	Quantidade de casas decimais
10^{-1}	Décimo	1
10^{-2}	Centésimo	2
10^{-3}	Milésimo	3
10^{-4}	Décimo de milésimo	4
10^{-5}	Centésimo de milésimo	5
10^{-6}	Milionésimo	6
10^{-7}	Décimo de milionésimo	7
10^{-8}	Centésimo de milionésimo	8
10^{-9}	Bilionésimo	9
10^{-10}	Décimo de bilionésimo	10
10^{-11}	Centésimo de bilionésimo	11
10^{-12}	Trilionésimo	12
10^{-13}	Décimo de trilionésimo	13
10^{-14}	Centésimo de trilionésimo	14
10^{-15}	Quatrilhonésimo	15
10^{-16}	Décimo de quatrilhonésimo	16
10^{-17}	Centésimo de quatrilhonésimo	17
10^{-18}	Quintilhonésimo	18
10^{-19}	Décimo de quintilhonésimo	19
10^{-20}	Centésimo de quintilhonésimo	20

Figura 2: Sequência - Casas Decimais

Método científico

O método científico é o caminho sistemático que você deve seguir para chegar a uma conclusão científica. Portanto podemos dizer que é um conjunto de procedimentos através dos quais um pesquisador realiza uma experiência para que, ao final, produza um novo conhecimento ou atualize e integre conhecimentos que já existiam. A utilização desses métodos garante autenticidade,

confiabilidade e valor científico à pesquisa. Já que direcionam a pesquisa à produção de conhecimentos válidos e científicos.

Para Gil (1999), o método científico é um conjunto de procedimentos intelectuais e técnicos utilizados para atingir o conhecimento. Para que seja considerado conhecimento científico, é necessária a identificação dos passos para a sua verificação, ou seja, determinar o método que possibilitou chegar ao conhecimento. Segundo o autor, já houve época em que muitos entendiam que o método poderia ser generalizado para todos os trabalhos científicos. Os cientistas atuais, no entanto, consideram que existe uma diversidade de métodos, que são determinados pelo tipo de objeto a pesquisar e pelas proposições a descobrir.

A pesquisa científica é a investigação de um fenômeno. O objetivo de qualquer pesquisa vai ser sempre encontrar a solução para algum problema. Dessa forma o método científico é o caminho que a experiência vai trilhar para alcançar um conhecimento válido.

Seguindo essa lógica a pesquisa científica faz parte de toda a vida acadêmica, seja na graduação, pós-graduação ou extensão. Para o desenvolvimento de qualquer pesquisa, há a necessidade de se elaborar um projeto. Ele é realizado por:

a) Alunos de graduação
b) Alunos de pós-graduação
c) Professores

Etapas da pesquisa científica:

a) Escolha do tema
b) Elaboração da pesquisa bibliográfica e seleção das obras relevantes
c) Formulação do problema
d) Especificação dos objetivos (gerais e específicos)
e) Justificativa da escolha da pesquisa
f) Definição da metodologia a ser empregada
g) Coleta dos dados
h) Tabulação dos dados
i) Análise, comparações e discussão dos dados
j) Conclusões
k) Relatório final

Metodologia

Para realizar o estudo de caso foi aplicado atividades a domicilio devido as escolas não estarem recebendo visitas por conta do COVID-19. Segundo o método científico acima, criado pelo autor Gil (2002), foram realizados estudos com base na estratégia do ensino da Computação desplugada e suas atividades sem o uso de tecnologia. Portanto, conseguimos identificar a importância do ensino da computação no ensino fundamental I, além de não ter gastos excessivos e obtendo um aproveitamento muito alto em relação ao aprendizado dos alunos.

As mudanças oriundas do novo século demandam pessoas capazes de solucionar problemas em diferentes situações, capacitando-se em diversas áreas.

Como forma de criar oportunidades para que os alunos conheçam mais sobre a área computacional e exercitem o raciocínio lógico, foi proposto atividades desplugada, relacionada à números binários,

com objetivo de apresentar a base numérica binária e compreender como o computador armazena as informações a partir da conversão decimal-binária e binária-decimal, realizando a conversão de palavras e números através da interpretação de binários.

Antes de começarem a atividade, foi feito uma breve explicação do que é a base binária, composta pelos número 0 e 1, explicitando como e onde ela é utilizada, fazendo comparações e conversões com a base decimal, 0 a 9, que já é de conhecimento dos alunos.

As atividades propostas foram: Trabalhar com números binários e Enviar mensagens secretas visando ensinar os fundamentos da Ciência da Computação através de atividades lúdicas e envolventes e, mais uma vez, colocando o aluno como autor do seu processo de ensino aprendizagem.

Materiais

Para a realização dos exercícios foram utilizados 5 cartões feito a mão, numerados com seus respectivos valores: 1, 2, 4, 8, 16. O notebook foi utilizado para leitura das atividades e as respostas foram feitos pelos alunos em uma folha sulfite separadamente utilizando lápis borracha e régua. Alguns retratos foram tirados durante a aplicação das atividades com permissão de cada aluno presente, destaca-se o engajamento do alunos ao realizarem uma atividade diferente do qual estão acostumados a fazer em aula. Abaixo temos imagens dos materiais utilizados na aplicação.

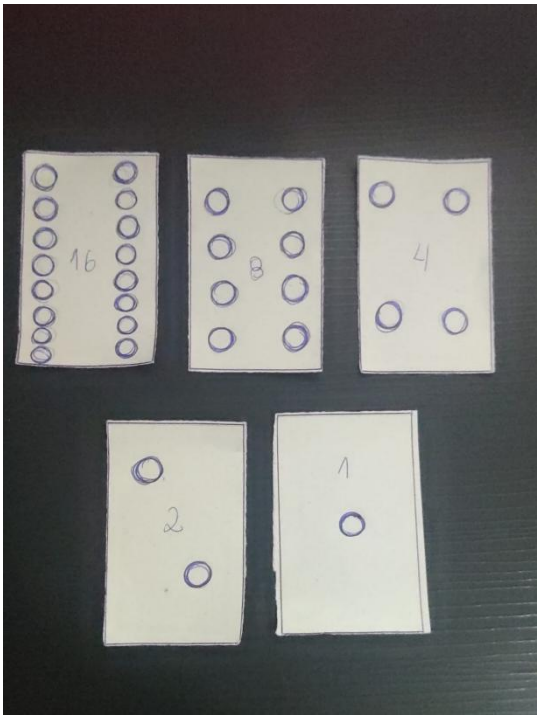

Figura 3: Cartões Utilizados

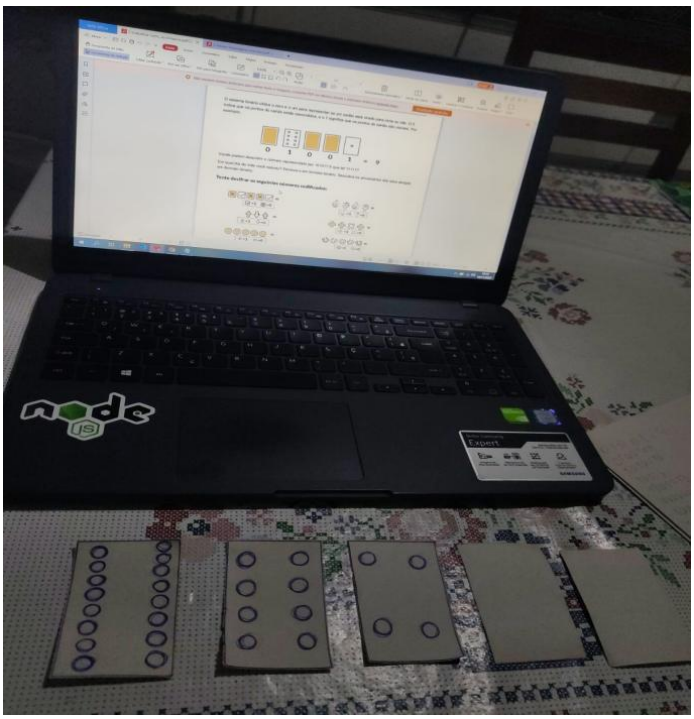

Figura 4: Atividade apresentada no Notebook

Atividades

A atividade Trabalhar com números binários consiste em decifrar códigos transformando-os em binários (01010), logo após transformar em número decimal (0, 1, 2, 3 ...), utilizando as cartas para ajudar a entender e chegar no resultado desejado. Após a explicação do exercício foram resolvidos alguns como exemplo até que ficasse claro para o aluno de como chegar no resultado esperado.

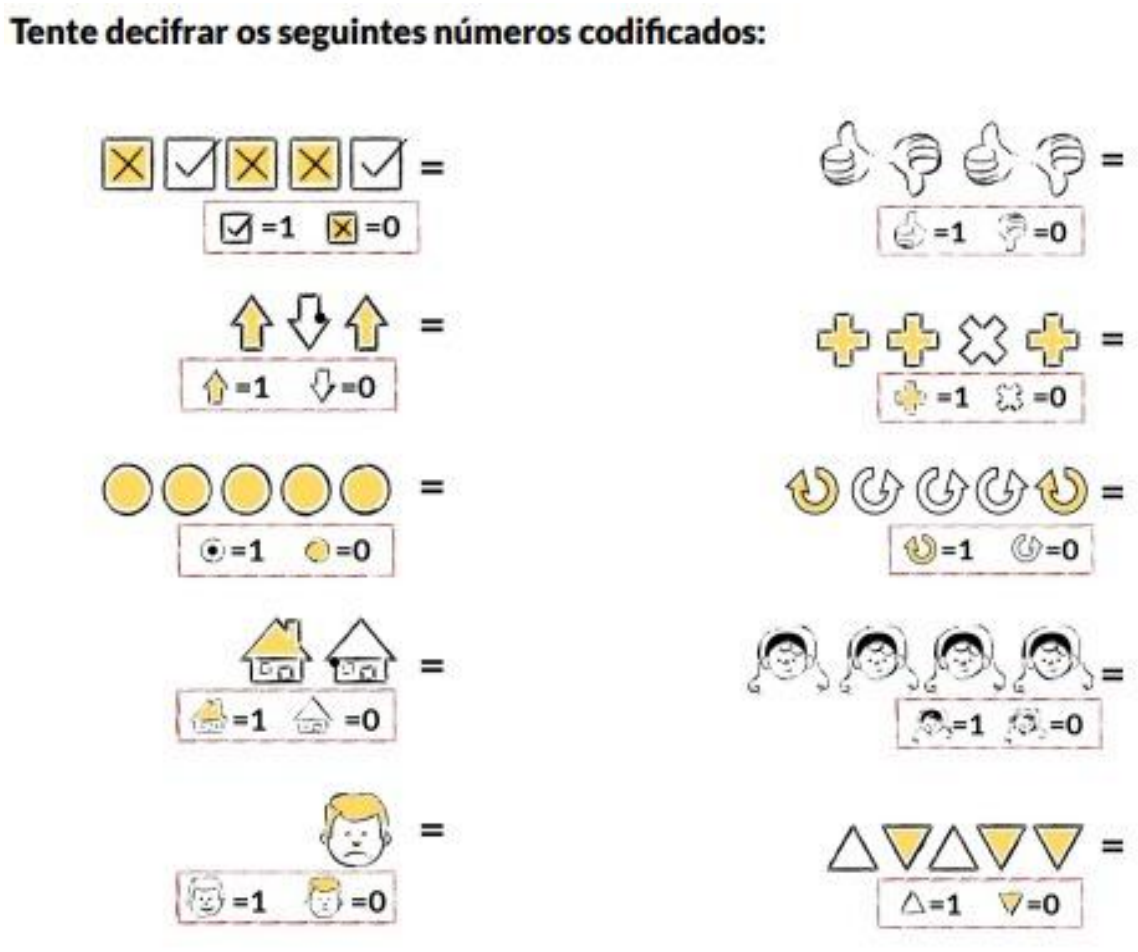

Figura 5: Atividade para decifrar números codificados

Neste exercício acima, foi pedido para que os alunos primeiramente substituírem as cores dos símbolos por 0 e 1 de acordo com o exemplo que consta na caixinha: quadradinho branco vale 1 e quadradinho amarelo vale 0. Dessa forma podemos descobrir o número binário que ele representa. Depois de encontrá-lo, iremos solucionar quanto corresponde o resultado em decimal. Para chegar ao resultado os alunos precisam usar as cartas que contém os valores 16, 8, 4, 2, 1 (A sequência é lida da direita para esquerda), assim precisam deixar as cartas viradas para cima somente os valores iguais a 1 e por fim as cartas com o valor 0 ficam para baixo. Vejamos o exemplo: quadradinhos branco (1) equivale a carta virada para cima, quadradinho amarelo (0) equivale a carta virada para baixo, portanto chegamos ao binário 01001. A partir disso fica fácil chegar ao decimal, soma-se os valores das cartas viradas para cima e vamos obter o resultado que é 9. Com essa lógica conseguimos resolver todos os exercícios propostos acima de forma lúdica sem usar quaisquer interação com o computador. Vale ressaltar que todos os alunos que tiveram dúvidas foram feitos mais exemplos até que entendessem o conceito aplicado para a resolução das atividades e pudessem realizar a atividade sozinhos.

A atividade Enviar mensagens secretas consiste em encontrar os números binários, depois disso encontrar o decimal correspondente a ele, utilizando novamente as cartas para facilitar a compreensão do exercício, por fim faz a troca dos valores em decimal com as letras para formar a mensagem secreta.

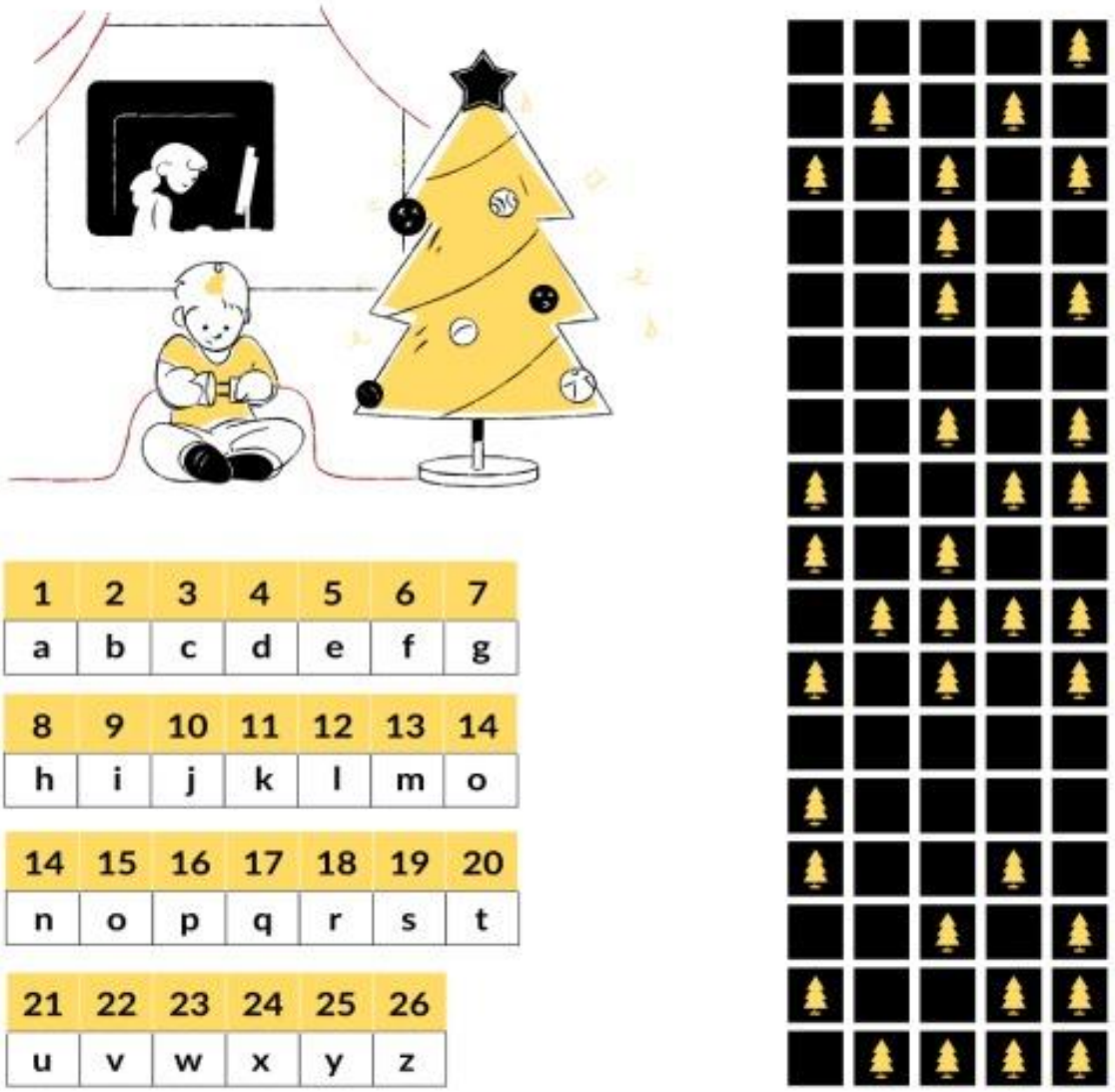

Figura 6: Atividade - Enviar Mensagens Secretas

Esta atividade tem um grau de complexidade um pouco mais difícil, portanto a explicação dela levou mais tempo. O exercício pede que os alunos encontrem uma mensagem secreta deixada por Tom que está preso no último andar de uma loja, para encontrar a mensagem é preciso separar um grande problema em pequenas partes. A primeira é encontrar os números binários, depois converter para decimal e por fim substituir o valor encontrado com a letra que corresponde ao decimal. Para descobrir o número binário que está em cada linha entende que caixinhas pretas representam o número 0 e a árvore amarela o número 1, portanto a primeira linha corresponde ao binário 00001. Com o primeiro passo feito vamos ao encontro do número decimal, para isso vamos utilizar as cartas com os valores 16, 8, 4, 2, 1 (A sequência é lida da direita para esquerda). Usando a lógica de que quadradinhos pretos correspondem a 0 e árvores amarelas correspondem a 1, chega à conclusão de que carta preta fica virada para baixo e árvores amarelas virada para cima. Dessa forma chegamos ao primeiro decimal que é 1. Depois de encontrar o decimal vamos para o último passo que é substituir o número 1 por sua letra que está numa tabela ao lado, logo podemos notar que a letra que corresponde

ao decimal encontrado (1) é “A”. Com a lógica em mãos é só aplicar para as demais linhas até que chegue no último binário e então iremos formar a frase (“AJUDE ESTOU PRESO ”).

Desafio: no fim dos exercícios foi aplicado duas atividades para ver o nível dos alunos e a dificuldade que teriam em resolvê-lo. O exercício consistia em converter binário-decimal e decimal-binário utilizando divisão para a conversão (decimal-binário) e multiplicação com potência (ex: $1x2^3$). Durante a aplicação por se tratar de um exercício mais complexo para eles, foram resolvidos alguns exemplos e explicado todas as vezes em que se encontravam confusos com a atividade, até que ficasse claro como se chegava a tal resultado da forma que vemos nesse exemplo abaixo:

Conversão Decimal-Binário

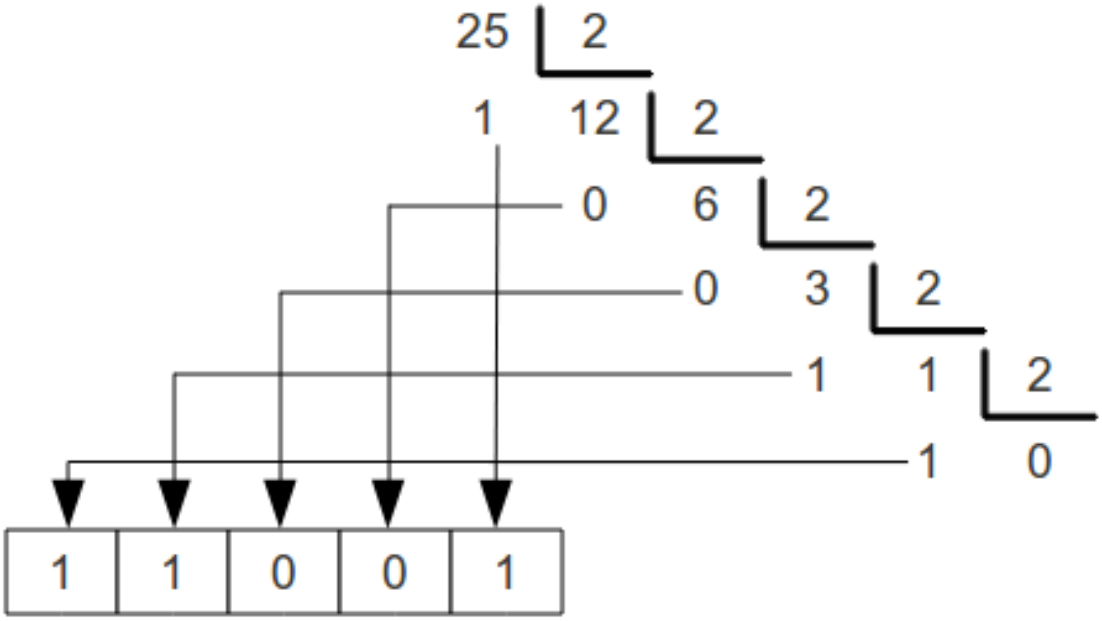

Figura 7: Conversão Decimal-Binário

Neste desafio de converter decimal-binário os alunos precisavam resolver fazendo a divisão do número 25 na base 2 até que o quociente seja 0. Então o que realmente importa para nós é o resto da divisão que sempre será 0 ou 1. Para finalizar e obter nosso resultado ignoramos o “0” e começamos a reescrever de baixo para cima, dessa forma vamos obter 11001 que é nosso 25 em binário.

Conversão Binário-Decimal

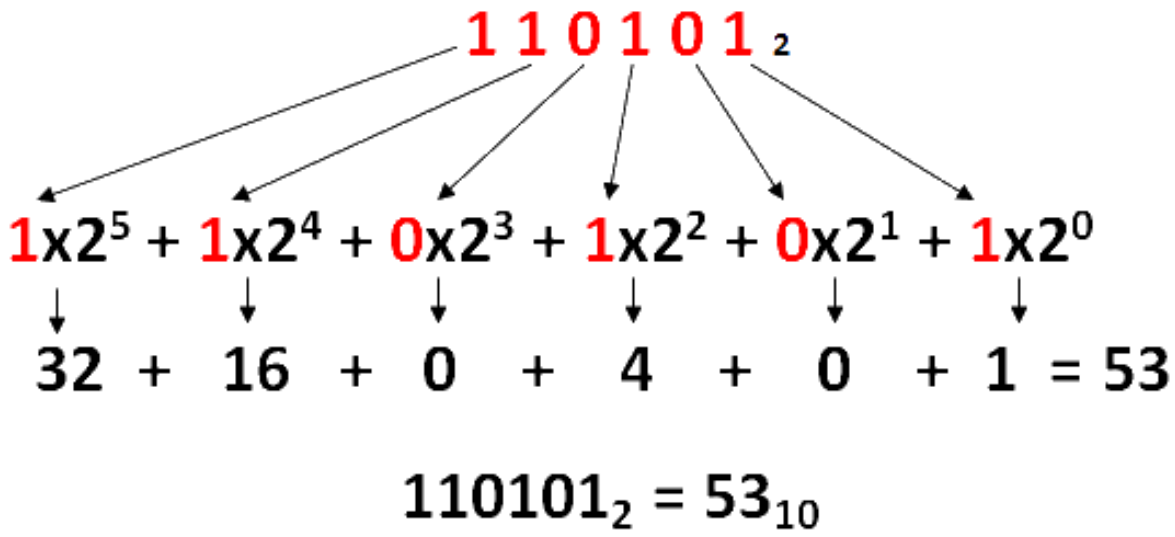

Figura 8: Conversão Binário-Decimal

Na conversão do Binário-decimal precisamos dividir o problema em pequenas partes por se tratar de uma conta mais complexa. A partir disso para encontrar o nosso decimal precisamos reescrever nosso binário, o primeiro passo é separar a multiplicação que é composta por um número binário multiplicado por 2 que é nossa base, depois disso elevamos o 2 pela posição em que se encontra cada binário contando da direita para esquerda e iniciando no 0. Após separar soma-se o resultado de cada multiplicação como mostra na imagem acima, com isso obtemos o número 53 em decimal.

Questões finais

No fim das atividades os alunos responderam a 5 perguntas, são elas:

1 - Qual sua idade?

2 - Possui computador?

3 - Tem acesso à Internet?

4 - O que achou das atividades?

5 - Teve dificuldade em resolver?

Resultados

Foram aplicados os exercícios “Trabalhar com números binários”, “Enviar mensagens secretas” e por fim um desafio de converter binário-decimal e decimal-binário, com alunos do ensino fundamental I, da 3ª, 4ª, 5ª série. Sendo um total de 10 alunos de escolas públicas e particulares com as idades de 8 a 11 anos, dentre eles haviam 6 meninas e 4 meninos ambos foram aplicados às atividades individualmente. Pude notar as características de cada aluno, a forma de como lidam com o problema, as dificuldades que tiveram na resolução dos exercícios e todo engajamento deles.

Aproveitamento

Todos os alunos conseguiram resolver os exercícios alguns com mais facilidades do que outros, mas todos chegaram ao fim. A média de acerto dos alunos foram de quase 100% na primeira atividade "Trabalhar com números binários", na segunda "Enviar mensagens secretas" a média de acerto foi de quase 100%, mas quando chegamos no desafio na atividade decimal-binário tivemos um aproveitamento de 70%. Na atividade "Binário-decimal" tivemos 25% e foi onde encontramos a maior dificuldade deles, isso nos mostra a importância do ensinamento lúdico e como ele pode ensinar coisas complexas de maneira muito mais simples e atingindo os mesmos resultados.

Aplicação

Durante a explicação dos exercícios vale ressaltar que cada aluno tem seu tempo para entender as questões e maneiras diferentes de compreender cada atividade. Por ter aplicado a Computação Desplugada em um aluno por vez, nota-se que é preciso ter vários métodos e maneiras de explicar uma atividade, pois cada aluno entende de uma forma, apenas falar não adianta; é preciso fazer junto, mostrar passo a passo, interagir com os alunos, sempre sorrir para motivá-los e vibrar a cada acerto deles. Depois de um tempo eles embalam e conseguem resolver sozinhos até o fim dos exercícios. A maioria deles gostaram das atividades e métodos que foram utilizados para ensinar, a gratificação deles em conseguir resolver um exercício nunca feito antes, os motivaram a resolver até o fim e mesmo sem saber aceitaram fazer o desafio.

Dificuldades

As maiores dificuldades dos alunos foram no início, pois nenhum deles tinham feito esse tipo de exercício antes, o que deixou eles assustados, porém com o método da computação desplugada que foi passado para eles, logo perderam o medo, interagiram e se divertiram com as atividades.

Fotos dos exercícios feitos pelos alunos

Aluno S. 11 anos, 5° série - Escola Pública

Figura 9: Momento da resolução dos exercícios com o aluno S.

Características do aluno

S. foi o melhor aluno da turma, teve muita facilidade em compreender os exercícios que lhe foram passados, o único que conseguiu resolver os desafios de conversão binário-decimal e decimal-binário, mesmo com dificuldade chegou no valor final esperado. O aluno ficou curioso para saber mais sobre programação, desta forma foi orientado que seguisse alguns passos como: pesquisar no Youtube, praticar com exercícios até que se sinta cada vez mais atraído pela tecnologia. Conseguiu ser o mais empenhado e o que levou menos tempo para resolver as atividades. Dentre todos os alunos que foram aplicados, ele se sobressaiu e se destacou-se pelo foco e entendimento das questões em poucos minutos de explicação, nenhum outro aluno teve essa desenvoltura.

Atividades: Trabalhar com números binários; Enviar mensagens secretas e Desafio converter binário-decimal e decimal-binário

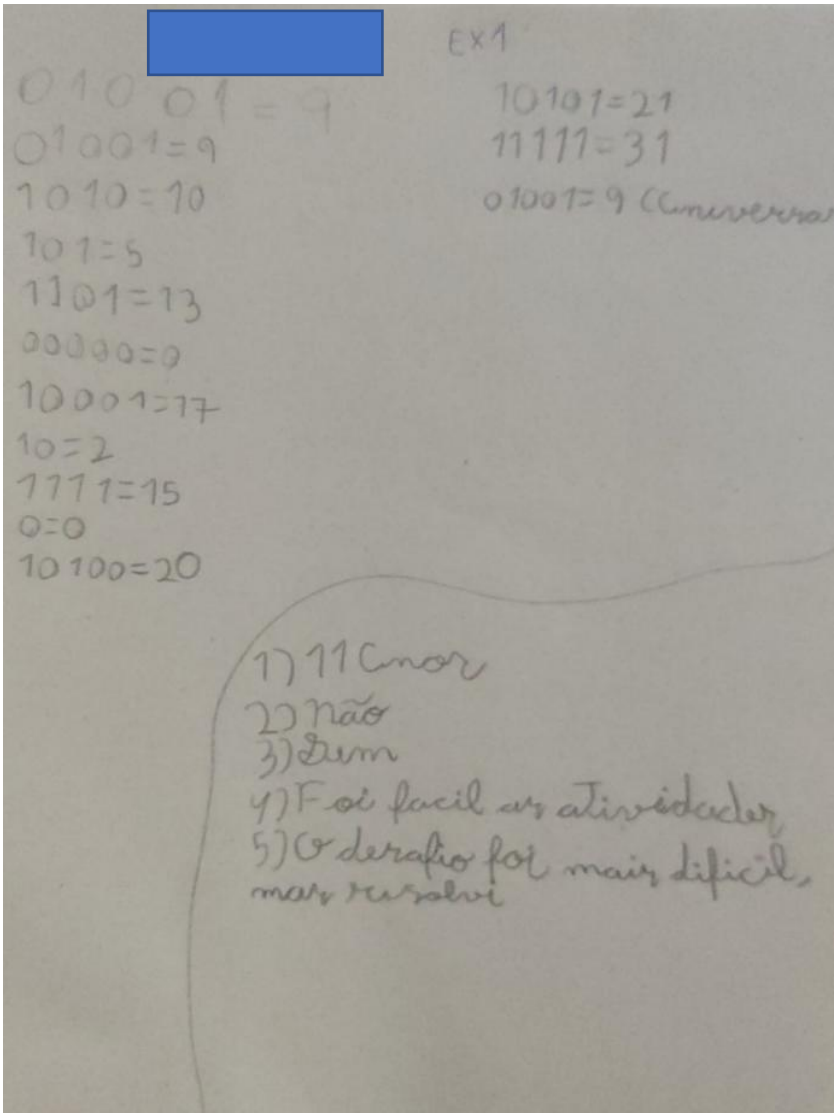

Figura 10: Atividade números binários do aluno S.

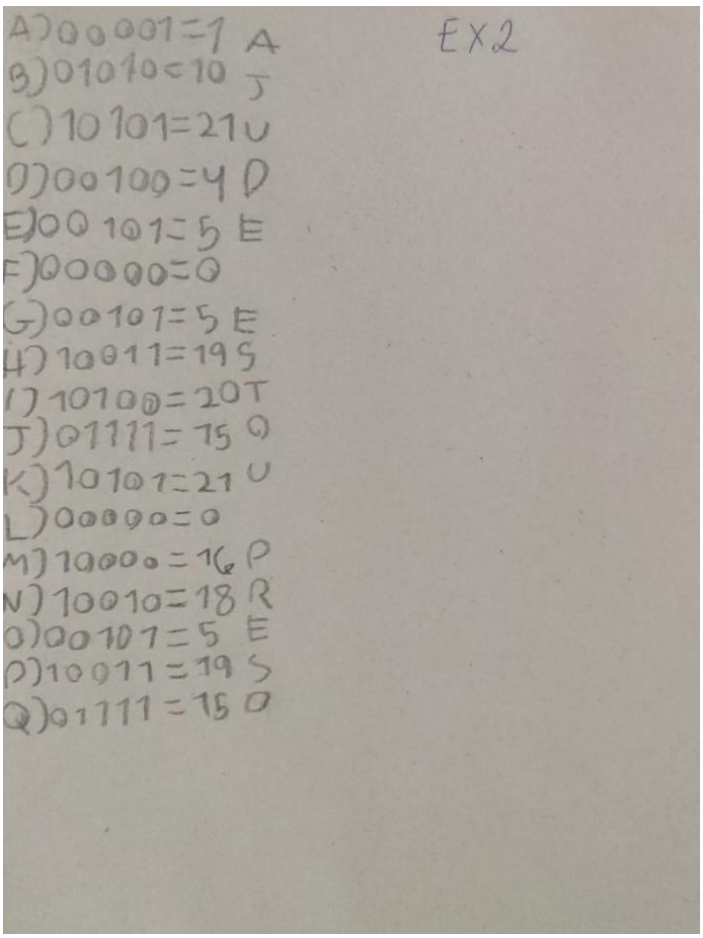

Figura 11: Atividade - Enviar Mensagens Secretas do aluno S.

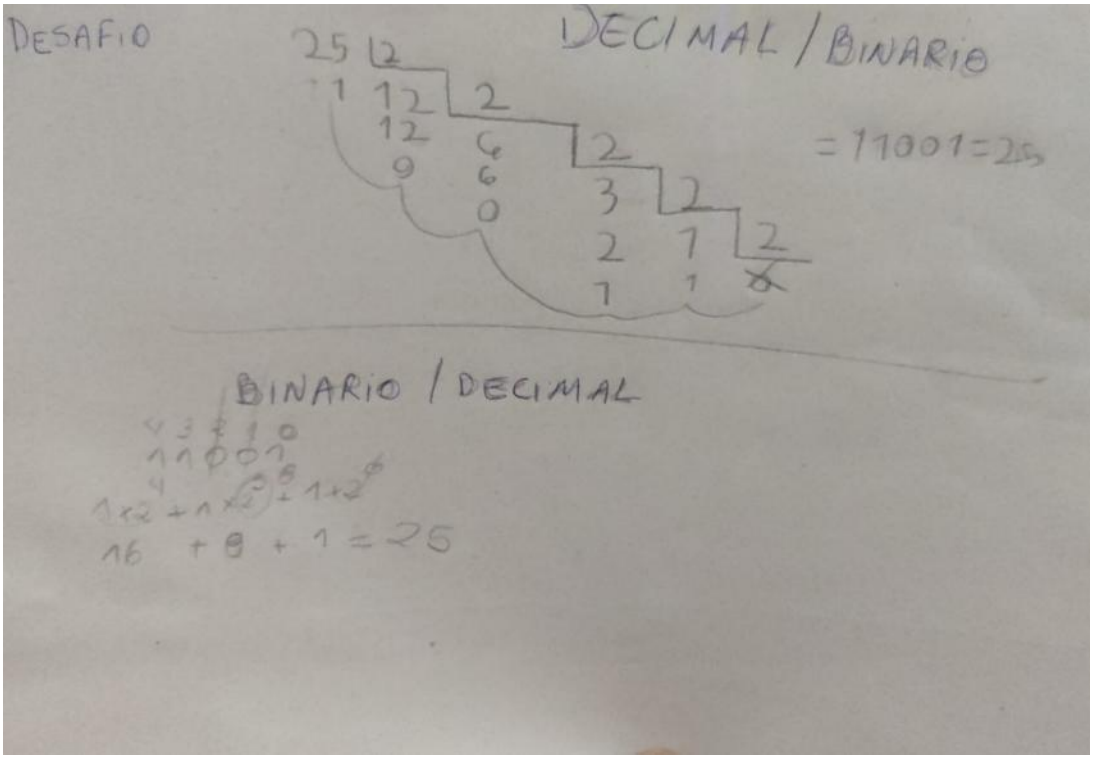

Figura 12: Atividade - Desafio Conversão Decimal-binário e Binário-decimal

Aluna L., de 8 anos, 3° Série - Escola Particular

Figura 13: Momento de Resolução dos Exercícios da aluna L.

Características do aluno

A aluna L. estuda em uma escola particular e apesar de ser a mais nova acabou se destacando pela sua concentração, dedicação e interação que teve no momento da resolução dos exercícios. A aluna teve uma leve dificuldade no começo para entender a lógica, é muito dedicada, interagiu constantemente com as cartas, perguntou quando teve dúvida e logo estava fazendo sozinha os exercícios. Não conseguiu resolver os desafios, mas aceitou em receber a explicação de como se chegava no resultado final. É a mais caprichosa e atenciosa.

Atividades - Trabalhar com números binários e Enviar mensagens secretas

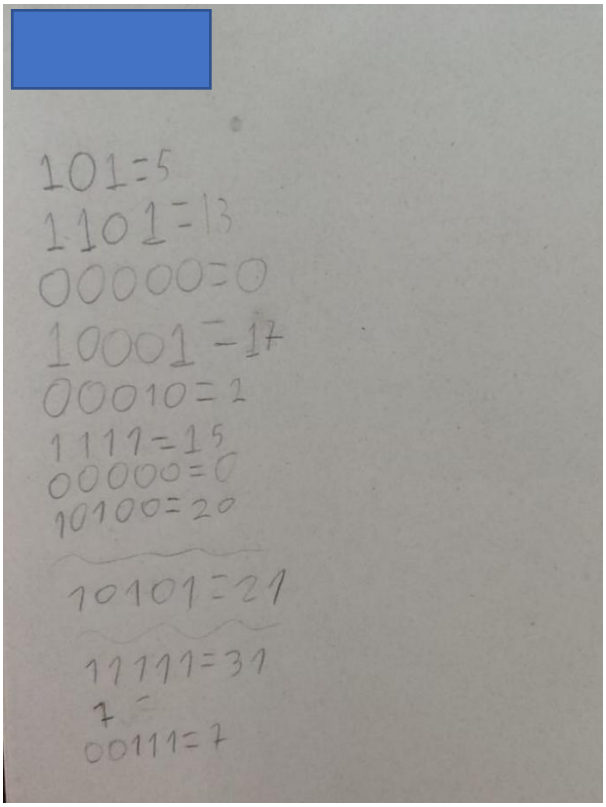

Figura 14: Atividade - Números Binários da aluna L.

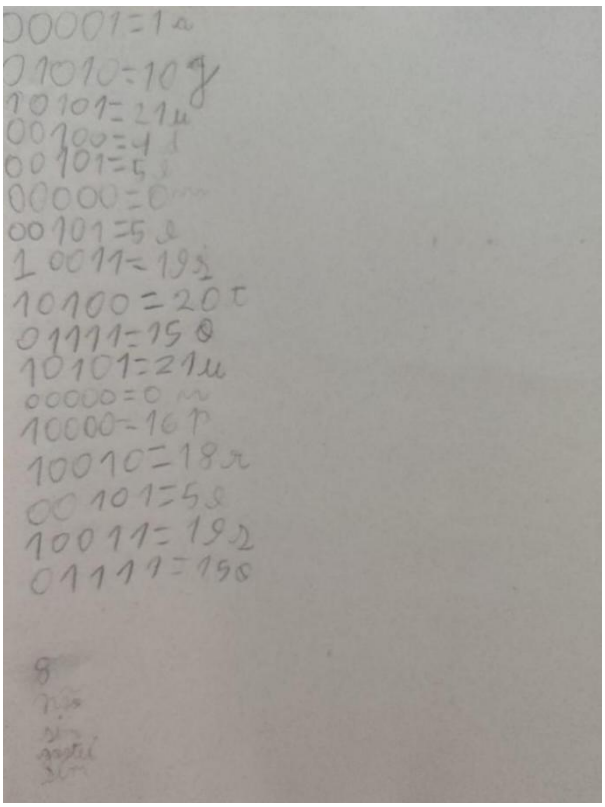

Figura 15: Atividade - Enviar Mensagens secretas da aluna L.

Aluna S. de 9 anos, 4° Série - Escola Pública

Figura 16: Momento da resolução dos exercícios da aluna S.

Características do aluno

A aluna S. estuda em escola pública, teve bastante dificuldade no começo para entender como funcionava as cartas, depois de explicar algumas vezes e fazer alguns exemplos conseguiu entender como resolver as atividades. Foi a única que descobriu a frase antes de resolver por completo o exercício. A aluna não conseguiu resolver os desafios, mas aceitou receber a explicação de como chegar ao resultado final. Se destacou por ser a única a ver o problema de modo geral, descobrindo a frase antes mesmo de resolver todo o exercício, estava atenta aos detalhes.

Atividades - Trabalhar com números binários e Enviar mensagens secretas

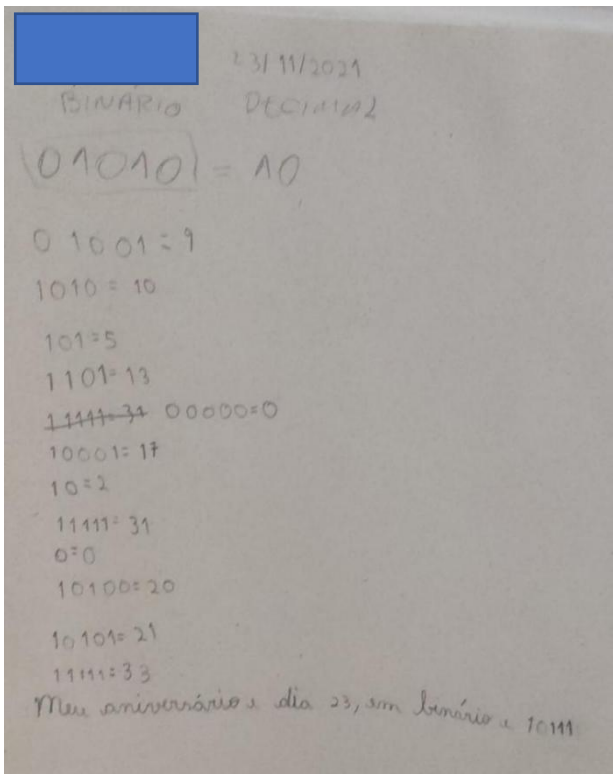

Figura 17: Atividade - Números Binários da aluna S.

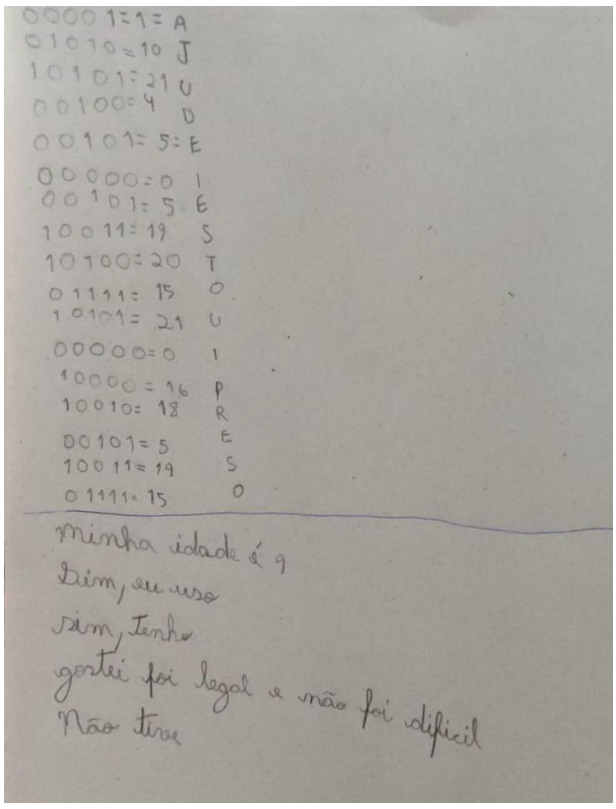

Figura 18: Atividade Enviar Mensagens secretas da aluna S.

Tabela de perguntas

Esta tabela mostra idade e respostas de todos os alunos que participaram do trabalho. De forma resumida estão as respostas para facilitar o entendimento do leitor.

Tabela 1: Perfil Alunos

Nome	Idade	Possui Computador?	Tem acesso a internet?	O que achou das atividades?	Teve dificuldade em resolver?
L[redacted]	8	Sim	Sim	Díficil	Sim
S[redacted]	9	Sim	Sim	Fácil	Sim
S[redacted]	11	Não	Sim	Fácil	Apenas os desafios
G[redacted]	10	Não	Não	Fácil	Sim
H[redacted]	9	Não	Sim	Média	Sim
J[redacted]	9	Não	Não	Média	Sim
J[redacted]	10	Sim	Sim	Média	Sim
Is[redacted]	11	Não	Sim	Fácil	Sim
F[redacted]	10	Não	Sim	Média	Sim
R[redacted]	9	Não	Sim	Média	Sim

A seguir, elencamos Tabelas e Gráficos detalhados sobre as idades de todos os alunos e todas as perguntas por eles respondidas enfatizando cada resultado obtido.

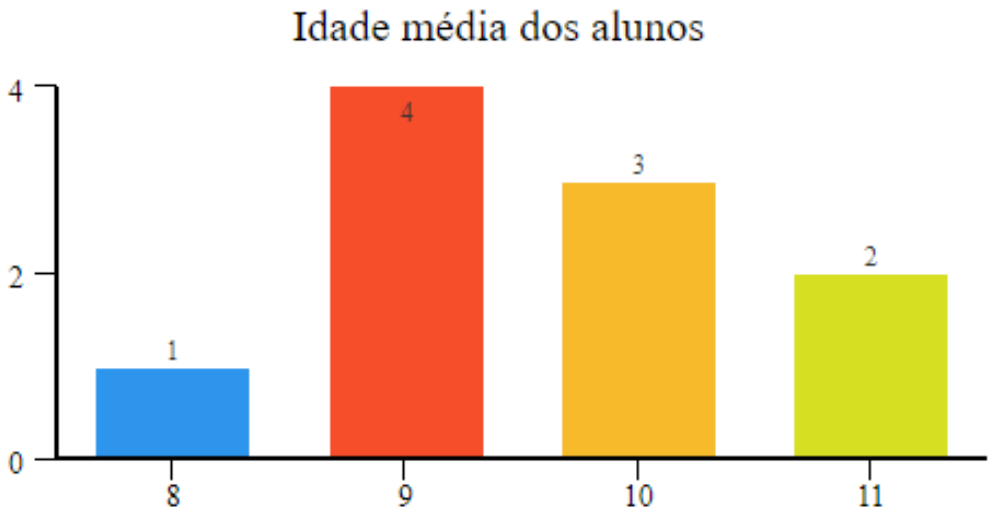

Figura 19: Idade Média dos Alunos

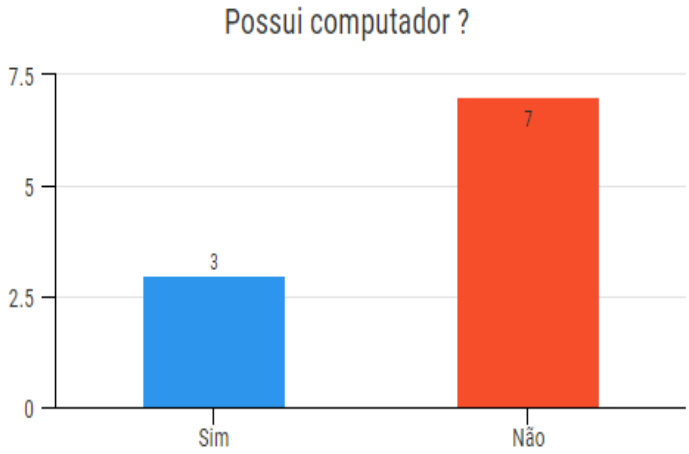

Figura 20: Possui computador?

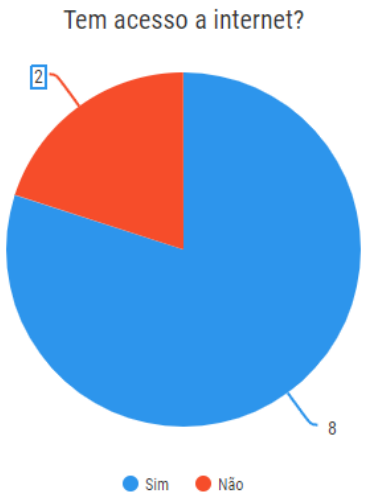

Figura 21: Tem acesso à Internet?

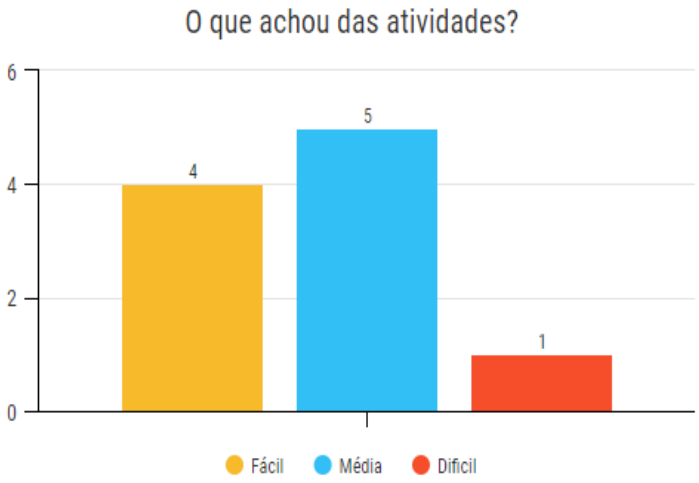

Figura 22: O que achou das atividades?

Figura 23: Teve dificuldades em resolver?

CONSIDERAÇÕES FINAIS

Devido ao desenvolvimento desse trabalho conseguimos ressaltar a importância de aplicar a Computação desplugada no ensino fundamental I. Com o auxílio das 5 perguntas feitas para complementar o estudo, ao fim das atividades, podemos enfatizar ainda mais a necessidade de aplicar esse método desde o ensino básico. Uma vez que percebemos que as crianças do mundo atual têm muita facilidade com a Internet e computadores, notamos pouco incentivo para usá-la como meio de aprendizado e vemos apenas como meio de diversão. Além disso, há uma procura muito grande por profissionais que saibam programar ou que estejam no ramo da tecnologia e essa demanda aumenta consideravelmente todos os anos. Com a aplicação da estratégia podemos notar que conseguimos afetar positivamente os alunos dando a eles um gostinho de como os computadores pensam e trabalham, dessa forma despertamos o gosto pela Computação desde o ensino fundamental.

REFERÊNCIAS BIBLIOGRÁFICAS

BELL, Tim et al. Ensinando Ciência da Computação sem o uso do computador. Computer Science Unplugged ORG, 2011.

Noemi Debora. Pensamento computacional: saiba como aplicar à realidade das escolas, 2020.

Gil, Antônio Carlos, Como elaborar projetos de pesquisa/Antônio Carlos Gil. - 4. ed. - São Paulo : Atlas, 2002.

LDB : Lei de diretrizes e bases da educação nacional. – Brasília : Senado Federal, Coordenação de Edições Técnicas, 2017. 58 p.

CURY, Carlos R. J, A educação Básica no Brasil, 2002.

Oliveira, Maxwell Ferreira de. Metodologia científica: um manual para a realização de pesquisas em Administração / Maxwell Ferreira de Oliveira. -- Catalão: UFG, 2011.

NASCIMENTO, Antonio W. P. T, et al. Uso da Computação Desplugada para ensino da Lógica Computacional nos Anos Iniciais do Ensino Fundamental II do Município ed Capitão Poço (Pará), 2011.

NÚMEROS decimais. Disponível em: https://escolakids.uol.com.br/matematica/numeros-decimais.htm

O que é sistema HEXADECIMAL? Disponível em: https://canaltech.com.br/produtos/O-que-e-sistema-hexadecimal/

COMO funciona o sistema BINÁRIO? Disponível em: https://canaltech.com.br/produtos/como-funciona-o-sistema-binario/

O que são NÚMEROS decimais? Disponível em: https://www.todamateria.com.br/o-que-sao-numeros-decimais/

Aprendendo sobre DECIMAIS e HEXADECIMAIS. Disponível em: https://digitalinnovation.one/artigos/aprendendo-sobre-decimais-e-hexadecimais

COMPUTAÇÃO Desplugada. Disponível em: http://www.desplugada.ime.unicamp.br/

Computação DESPLUGADA. Disponível em: https://fatechgirls.org/computacao-desplugada/

Pensamento COMPUTACIONAL: entenda o que é e sua importância. Disponível em: https://idocode.com.br/blog/programacao/pensamento-computacional/

PENSAMENTO computacional e programação como ferramentas de aprendizagem. Disponível em: https://institutoayrtonsenna.org.br/pt-br/meu-educador-meu-idolo/materialdeeducacao/pensamento-computacional-e-programacao-como-ferramentas-de-aprendizagem.html

Pensamento COMPUTACIONAL: saiba como aplicar á realidade das escolas. Disponível em: https://escolasdisruptivas.com.br/metodologias-inovadoras/pensamento-computacional/

Pensamento Computacional: o que é e como contribui para o desenvolvimento das crianças. Disponível em: https://blog.academia.com.br/pensamento-computacional/

Método CIENTÍFICO: saiba como escolher o melhor para os objetivos da pesquisa. Disponível em: https://blog.mettzer.com/metodo-cientifico/

Como é formada a EDUCAÇÃO Básica brasileira? Disponível em: https://www.educamaisbrasil.com.br/educacao/escolas/como-e-formada-a-educacao-basica-brasileira?gclid=cj0kcqia15ynbhdtarisagnwe0uanwhb-ppf6_qvgtwlavy_nxr7yu5ifmbqbdfxehtzijuelnxm1kkaahv6ealw_wcb

ANÁLISE DA PLATAFORMA AWS LAMBDA EM AMBIENTES DE NUVENS COMPUTACIONAIS

ANALYSIS OF THE AWS LAMBDA PLATFORM IN COMPUTATIONAL CLOUD ENVIRONMENTS

Yasmin Oliveira PISONI

yasmin.pisoni@hotmail.com

Aluna do curso de Bacharelado em Ciência da Computação

Centro Universitário Padre Anchieta - Jundiaí - SP

Carlos Eduardo CÂMARA

dinhocamara@gmail.com

Pesquisador na Área de Ciência da Computação

Doutorado e Mestrado em Engenharia Elétrica - FEEC/UNICAMP – Campinas/SP

Resumo

O conhecimento e o desenvolvimento de novos modelos de computação em nuvem podem levar a uma grande economia, nesse contexto, o modelo serverless computing apresenta-se como uma das possíveis soluções para disponibilização de aplicações e serviços online, tem como uma de suas principais características a disponibilidade de recursos por demanda. Esse estudo busca preencher a lacuna em relação a estudos que abordem a serverless computing, focando principalmente na plataforma AWS Lambda. Com esse estudo foi visto que o AWS Lambda elimina a complexidade de lidar com servidores, além de apresentar um modelo de faturamento de pagamento mediante solicitação, em que são eliminados os custos da capacidade computacional considerada ociosa.

Palavras-Chave

armazenamento; nuvem; serveless computing; AWS lambda, serveless, AWS

Abstract

The knowledge and development of new cloud emergency models can lead to great savings, in this context, the serverless computing model presents itself as one of the solutions for the provision of services and services online, with availability as one of its main characteristics of resources on demand. This study seeks to fill a gap in relation to studies that address serverless computing, focusing primarily on the AWS Lambda platform. With this study, it was seen that AWS Lambda eliminates the complexity of dealing with servers, in addition to presenting a pay-on-demand billing model, in which the costs of computing capacity considered idle are eliminated.

Keywords

storage; a cloud; serveless computing; AWS lambda, serveless, AWS

1.INTRODUÇÃO

O armazenamento em nuvem vem apresentando um crescimento exponencial e adquirindo mais usuários nos últimos anos. Esse tipo de armazenamento inovou a maneira como as informações são gerenciadas, acessadas e utilizadas. As nuvens computacionais disponibilizam diferentes recursos que são oferecidos como serviços pelo provedor. Os recursos desse tipo de armazenamento estão disponíveis de acordo com a capacidade e a quantidade que o usuário necessita, onde futuramente realizará o pagamento a respeito da quantidade de armazenamento adquirida (VAQUERO et al. 2009).

Desse modo, diante do alto crescimento de usuários de Internet observado nos últimos anos (IBGE, 2018), o volume de acessos para serviços e aplicações online tende a seguir aumentando de forma contínua. Portanto, o conhecimento e a descoberta de novos modelos de infraestruturas em nuvem podem gerar impactos diretos em custos computacionais.

Com o rápido avanço da tecnologia, surgem novos modelos de computação em nuvem e um dos considerados mais importantes é o denominado Serverless Computing, esse sistema é baseado na transparência ao cliente das informações do sistema operacional. A principal diferença desse sistema é a ausência dos servidores nas aplicações. A lógica da aplicação é definida pelo usuário, dessa forma quando é requisitada a execução, uma cópia da função é executada pelo provedor, que envia o resultado da execução de volta para o usuário (ADZIC; CHATLEY, 2017).

O sistema Serverless Computing é considerado um grande avanço em relação ao sistema de armazenamento em nuvem pois fornece toda a responsabilidade de gestão dos recursos para o provedor. Seu funcionamento é baseado no escalonamento de recursos de computação para executar funções *stateless* fornecidas por programadores e acionadas mediante eventos. O escalonamento e gestão dos recursos são de responsabilidade do provedor do serviço, e o preço é calculado com base na utilização durante o tempo de execução das funções em milissegundos. Exemplos de aplicações que utilizam o ambiente *serverless* são aplicações web, mobile, IoT, Big Data e *chatbots* (SREEKANTI et al. 2020).

Um dos principais modelos de computação que utiliza *serverless* é o de Function-as-a-Service (FaaS), que permite desenvolver, executar e gerenciar funcionalidades de forma mais prática. São *stateless* e temporárias, podendo permanecer ativas durante a execução de uma única requisição, e gerenciadas inteiramente pelo provedor, que fica responsável por garantir sua escalabilidade e disponibilidade. O usuário não é cobrado por recursos inativos. Dessa forma, esse modelo é econômico quando comparado aos mais tradicionais (ADZIC; CHATLEY, 2017).

As plataformas FaaS são capazes de escalonar centenas ou até milhares de funções em segundos ou minutos através da implementação da separação entre computação e armazenamento, esse tipo de designer está cada vez mais comum no armazenamento em nuvem. No entanto, a execução das funções utiliza o modelo *stateless*, isto é, sem persistência de dados, exigindo o uso de serviços de armazenamento externo para a troca de estado em aplicações *stateful*. Também há um *trade-off* entre a latência e o custo que deve ser considerado no uso das plataformas *serverless* (KLIMOVIC et al. 2018).

Para as empresas, com a nuvem não existe a necessidade da utilização de um hardware próprio, qualquer arquitetura baseada em servidor ainda exige que a escalabilidade e a confiabilidade sejam estruturadas (AWS, 2017). Elas serão responsáveis por escalar as possíveis frotas de servidores, levando em consideração a redução dos picos de cargas, quando possível para assim reduzir os custos, levando sempre em consideração a integridade dos usuários e dos sistemas (AWS, 2017).

O *serverless computing* foi desenvolvido para enfrentar os desafios que o mercado procura, oferecendo às empresas uma redução específica nos custos e no tempo de entrada no mercado (AWS, 2017). Nesse contexto, o modelo *serverless computing* apresenta-se como uma das possíveis soluções para disponibilização

de aplicações e serviços online, pois tem como uma de suas principais características a disponibilidade de recursos por demanda (AWS, 2017).

A AWS Lambda permite que as empresas executem código sem a necessidade de manterem seus servidores. Os desenvolvedores tornam-se responsáveis apenas pela execução do código e os valores cobrados são referentes ao número de instâncias que foram utilizadas após a execução dos mesmos. Ele possui uma grande gama de linguagens já pré-definidas que são aceitas, o AWS Lambda permite que eles se concentrem em suas tarefas sem ter que gerenciar servidores ou até mesmo que sejam responsáveis pelas atualizações do sistema operacional e todas as aplicações instaladas estejam atualizados (AWS, 2020).

A AWS garante que os ambientes de execução usados nas funções do Lambda estejam sempre em constante evolução. Isso significa que os ambientes de execução estão constantemente sendo substituídos por novos. Devido a isso, os dados salvos em um evento permanecerão lá apenas por um curto período, a não ser que os mesmos sejam solicitados novamente dentro da função (AWS, 2020).

A utilização do modelo *serverless* permite às empresas utilizarem um provedor de serviços de nuvem para reduzir despesas relacionadas às operações e manutenções dos servidores e aos demais processos associados, como gerenciamento de *patches* e escalonamento (OLIVEIRA, 2020). Dessa maneira, elas tornam-se responsáveis apenas pelo desenvolvimento de códigos, ao invés de usar recursos humanos para manter e proteger a infraestrutura do servidor. Isso significa que as empresas que optam por trabalhar com o *serverless* acabam se beneficiando devido a maior flexibilidade, automação, economia, agilidade e segurança de dados que o *serverless* oferece (OLIVEIRA, 2020).

Desse modo, a justificativa desse estudo é auxiliar na compreensão do s*erverless computing* com foco na plataforma AWS, assim como na análise de seus impactos em termos de eficiência, redução de custos e tempo de execução, com foco direto na AWS Lambda.

A divisão deste artigo está organizada da seguinte maneira: a seção 2 apresenta os modelos de computação em nuvem existentes e os conceitos de AWS Lambda. Na seção 3, a análise sobre a plataforma AWS Lambda. Na seção 4, apresenta o algoritmo simples de AWS Lambda sendo o “Hello Word”. E a seção 5 conclui o trabalho abordando os possíveis impactos da AWS Lambda em relação às empresas.

2.COMPUTAÇÃO EM NUVEM

O desenvolvimento da computação em nuvem, também denominada como Cloud Computing, está na frente para a centralização e gerenciamento de sites e aplicativos, vem causando muitas transformações digitais e já tem garantido o seu lugar dentro das grandes empresas atualmente (SOFTLINE, 2017). Ela considera a computação um serviço ao invés de um produto, sendo assim os desenvolvedores usam suas máquinas como um determinado serviço, facilitando a aplicação e os acessos aos dados em qualquer lugar que estiverem. Com isso as empresas não têm a necessidade de possuírem uma plataforma de desenvolvimento ou um servidor e através disso torna-se possível que as execuções das tarefas sejam realizadas de maneira mais simples e de forma remota (SOFTLINE, 2017). Alguns dos serviços prestados pela computação em nuvem são: servidores virtuais, armazenamento, softwares e desenvolvimento de softwares (SOFTLINE, 2017).

Com o grande leque de possibilidades quando se fala da computação em nuvem elas acabaram sendo divididas em três categorias. De acordo com (MELL, 2011), os três modelos principais de computação em nuvem são a Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS), e o Software como Serviço (SaaS).

No modelo SaaS o usuário é o responsável pelo gerenciamento da capacidade do sistema operacional. Nesse modelo as aplicações estão armazenadas em data centers de nuvem. Na maioria dos casos o usuário não é um desenvolvedor, embora também seja possível oferecer serviços que podem ser incorporados em outras aplicações. Nesse modelo é oferecido um produto completo, ou seja, o gerenciamento é executado pelo próprio

provedor, fazendo com que o usuário não se preocupe com a manutenção ou gerenciamento da infraestrutura (MELL, 2011)

Já o modelo PaaS é voltado para usuários que necessitam de maior controle sobre os recursos computacionais utilizados, permitindo o desenvolvimento de aplicações na nuvem, nesse modelo o usuário não possui a necessidade de gerenciar os sistemas operacionais e o hardware tendo a possibilidade mais eficiência e foco maior nas implantações e gerenciamento já que não realizará as tarefas repetitivas necessárias (MELL, 2011)

No modelo IaaS o gerenciamento é realizado pelo provedor de serviços e são oferecidos recursos, como processamento, rede e armazenamento, que são configurados e gerenciados pelo usuário (MELL, 2011). O IaaS possui os componentes básicos da infraestrutura na nuvem e apesar de os serviços serem realizados pelo provedor, ele possui flexibilidade e controle de gerenciamento. Esse modelo é o mais conhecido dentro dos departamentos de TI por possuir recursos semelhantes aos já existentes (SOFTLINE, 2017).

Além dessas categorias de nuvens que existem, a IaaS e a PaaS dividem outros três tipos de classificações que são: pública, privada e híbrida. Na nuvem pública todas as informações ficam disponíveis na internet e são compartilhadas entre diversos usuários. No modelo de nuvem privada os dados são de acesso próprio do servidor da empresa, mantendo os arquivos privados e mais seguros. Na híbrida ocorre a junção das duas anteriores (SOFTLINE, 2017).

A necessidade de constante crescimento de redes compartilhadas, com taxas de respostas mais rápidas para os serviços solicitados, com velocidade para armazenar recursos e com o tempo de execuções acelerados (MULLER et al. 2020) fez com que houvesse a necessidade de avanços na computação em nuvem, com isso as vantagens e competitividade dentro das empresas cresceram bastante com o decorrer dos anos, e dentre elas estão: a redução dos custos já que a Cloud Computing permite que as empresas tenham serviços fornecidos de acordo com as suas necessidades sem realizar grandes investimentos, a praticidade pois os procedimentos como instalação, manutenção e atualização ficam por conta dos fornecedores. Os modelos de computação em nuvem conseguem atender todos os tipos de demandas e vieram para auxiliar as empresas através de inovações independentes. A partir disso deu-se início a FaaS que é considerado também um modelo de computação em nuvem, já que com ele não é necessário que o usuário se preocupe com a manutenção da sua infraestrutura (CLEAN CLOUD, 2018).

Com isso, a computação em nuvem sofreu diversas evoluções com o decorrer dos anos e assim surgiu a *Serverless Computing* ou também conhecido como, computação sem servidores. É composta por uma arquitetura em que a execução dos códigos são gerenciados por provedores em nuvens diferente dos métodos tradicionais onde os desenvolvimentos são feitos através de instalações diretas nos servidores (IPSENSE, 2020), porém essas aplicações não estão completamente sem os servidores como o próprio nome já diz, a origem de "sem servidor" é em relação a experiência do cliente com os servidores já que eles acabam sendo invisíveis para os usuários que não possuem acessos ou qualquer tipo interação com os servidores (IBM, 2021).

O Serverless Computing transfere as responsabilidades de gerenciamento das tarefas operacionais e a infraestrutura da nuvem em back-end para o provedor da nuvem. (IBM, 2021).

Dentre as vantagens da *Serverless Computing* podemos listar as principais que são:

- A redução de custos: uma vez que só será realizada a cobrança caso o seu código esteja em execução (IPSENSE, 2020).
- Possui poucos riscos de ameaças ao sistema: os provedores possuem equipes especializadas em segurança que são responsáveis pela atualização, monitoramentos aos possíveis ofensores do sistema (IPSENSE, 2020).
- Ampliação Global: com a utilização das nuvens é possível ampliar as atividades de

determinadas regiões e realizar a implementação em questão de minutos, de onde estiver (IPSENSE, 2020).

Para o usuário conseguir obter os sucessos desejados é necessário possuir uma boa conexão com a internet, garantindo velocidade e escalabilidade necessárias para a aplicação da *Serverless Computing* e também garantir que as ferramentas utilizadas no dia-a-dia estejam alocadas em nuvem e, caso não estejam, o usuário precisará garantir que os seus recursos sejam escaláveis. Atualmente diversos provedores como Azure, Google e IBM oferecem a plataforma "sem servidor", porém, esse formato de plataforma ainda está em expansão no território nacional, com algumas empresas como Nubank, SemParar e MaxMilhas apostando na *Serverless* Computing um cenário diferente do que encontramos em muitos países do exterior (KOHN, 2018).

Uma das empresas que deu início a *Serverless Computing* foi, Amazon, uma empresa norte-americana que começou com a computação sem servidor no ano de 2014, com a plataforma Amazon Web Service, que fornece diversos serviços para os usuários. A AWS (Amazon Web Service), possui diversos recursos fornecidos, porém o foco deste estudo será o AWS Lambda.

2.1. Conceitos do Lambda

O AWS Lambda possui termos que são muito utilizados quando se faz referência às suas execuções. Algumas denominações para as chamadas de execução são:

- Função: "Recurso" onde após ser chamado, ocorre a execução do código. Utilizada para processar eventos, ou para realizar outros serviços solicitados por outras partes da AWS (AWS, 2021).
- Evento: Trata-se de uma documentação em formato de JSON que contém os dados para serem processados dentro da função Lambda (AWS, 2021).
- Ambiente de execução: Fornece um ambiente seguro e isolado para assim conseguir gerenciar e executar as funções chamadas (AWS, 2021).
- Arquiteturas de conjuntos de instruções: É onde é definido o tipo de processador no qual o Lambda irá executar as tarefas. As arquiteturas aceitas por ele são: arm64 (ARM de 64 bits) e x86_64 (x86 de 64 bits). Após definir a arquitetura não é possível alterá-la posteriormente (AWS, 2021).
- Pacote de implantação: O Lambda aceita dois modelos de pacote de implantação: o arquivo .zip com o código do usuário já inserido dentro do arquivo para assim o Lambda fornecer o seu sistema para realizar determinada função. E também o modelo imagem de contêiner, porém a mesma deve ser compatível com as especificações desejadas pela AWS (AWS, 2021).
- Runtime: O tempo de execução fornece um ambiente específico de código, onde ele transmite eventos com informações de resposta entre o Lambda e outras funções. O tempo escolhido pode ser criado pelo usuário ou escolher os disponíveis dentro da AWS, os tempos de execução devem ser definidos de acordo com os seus objetivos dentro da plataforma (AWS, 2021).
- Extensão: A extensão tem como objetivo permitir que o usuário aumente o seu número de funções, podendo criar várias extensões já existentes nas ferramentas parceiras do AWS ou até mesmo criar as suas próprias (AWS, 2021).
- Concurrency: ou simultaneidade representa o número de solicitações que uma determinada função recebe a cada momento, as funções só conseguem ser invocadas quando a anterior já estiver terminado (AWS, 2021).

Os termos acima serão citados no decorrer do artigo, e podem ser levados em consideração apenas quando forem abordados os temas referente a AWS Lambda.

3.AWS LAMBDA

O Amazon Web Services Lambda é um serviço de computação serverless computing orientado a eventos que possibilita o usuário a criar pequenas funções com funcionalidades únicas, onde cada uma é

responsável por desenvolver determinadas tarefas necessárias (MACHADO, 2020). Sua utilização tem similaridade com algumas das categorias mais comuns, como:

- Aplicativos Web: Automatização da implantação e hospedagem utilizando os recursos fornecidos pela AWS (AWS, 2021).
- Back-End Web e de aplicativos: Integração entre o backend e o frontend utilizando os recursos fornecidos pela AWS (AWS, 2021).
- Processamento de dados: Processamento de execução de dados, após serem acionados para realizar o armazenamento de dados (AWS, 2021).
- Atuações paralelas: divisão entre as tarefas de curta e longa duração, para conseguir executar os dados em paralelo, levando em consideração o tempo de duração de cada um (AWS, 2021).
- Carga de trabalho para internet das coisas: processamento de dados para os dispositivos ligados à internet das coisas (AWS, 2021).

O Lambda é o responsável por disponibilizar os recursos necessários para a execução das funções com base na solicitação ou eventos de entrada, para qualquer demanda, é responsável também por executar automaticamente os códigos dos eventos, por meio do Amazon API Gateway, Amazon S3, Amazon DynamoDB e, também, AWS Step Functions (AWS,2021).

O modelo seguido pelo Lambda é o de execução orientada a eventos, que é quando as funções são executadas através de *containers*. O Lambda é um dos recursos mais importantes que a AWS oferece, pois pode ser considerado o "futuro" da interação desenvolvedor x infraestrutura. Ele permite que os desenvolvedores usem a infraestrutura da AWS sem ter preocupações, já que o mesmo é responsável pela implantação, gerenciamento, manutenção do servidor e do sistema operacional, escalabilidade automática e a implantação do código, registro das execuções do código, monitoramento da integridade do servidor e disponibilização das estatísticas detalhadas além de toda a infraestrutura necessária, tudo que o usuário precisará fornecer é apenas o código em uma linguagem que o sistema suporte. (AWS, 2021)

Com os recursos oferecidos pela AWS o usuário é capaz de acionar mais de 200 serviços e aplicações (AWS, 2021). E através do exemplo abaixo é possível notar que para criar determinadas aplicações é necessário apenas combinar o Lambda com outros serviços como os citados anteriormente que são: Amazon S3, Amazon API e Amazon DynamoDB.

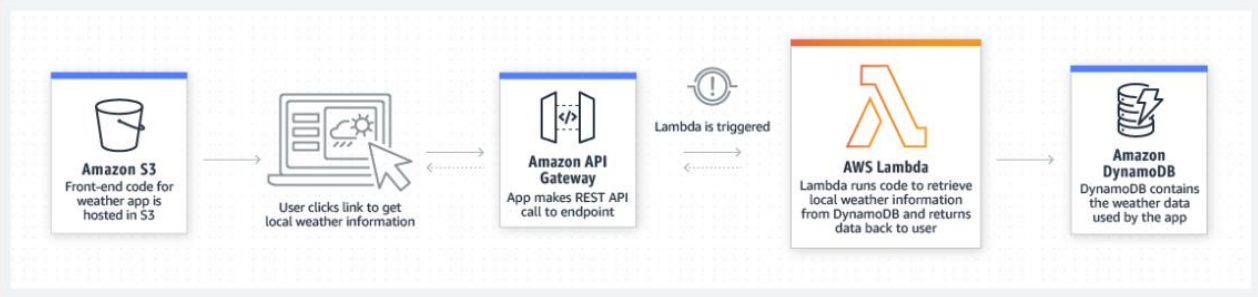

Figura 1. Um exemplo de aplicação Web (AWS, 2021)

O código responsável pela execução no AWS Lambda é chamado de "função Lambda". Após criar a sua função ela ficará disponível para futuras execuções e sempre que forem acionadas pelo servidor. As funções não possuem afinidades com a infraestrutura, pois apenas assim elas podem ser escaladas mais rapidamente de acordo com o número de solicitações que forem recebidas (AWS, 2021). Além disso, as funções são executadas apenas quando são chamadas automaticamente, podendo ser algumas por dia até milhares por segundo, sendo o Lambda um serviço altamente disponível (AWS, 2021). Ele é configurado para

dimensionar instantaneamente uma determinada demanda para um processo de execução de código em paralelo. Ele possui também o *downscale* que permite que as funções desnecessárias parem de funcionar automaticamente a partir da execução do código (AWS, 2021).

O Lambda aceita diferentes configurações para as funções utilizando códigos em formato ZIP ou de imagens de contêiner. As linguagens aceitas no Lambda são: Java, Python e Node.JS e outras linguagens, que possuem frameworks de código aberto. Há também ferramentas para *deploys* ainda mais convenientes e rápidos, como Apex, *Serverless* e Sparta que podem ser utilizadas (AWS, 2021).

3.1. Segurança da Nuvem AWS Lambda

A segurança da nuvem AWS Lambda é responsável pela infraestrutura que executa os produtos dos usuários, eles contam com auditorias de terceiros que são responsáveis pela realização dos testes de verificação da eficácia da segurança e os seus serviços são protegidos por procedimentos de segurança da rede global da AWS. (AWS, 2021)

Entretanto as responsabilidades de segurança são compartilhadas com os usuários, eles têm como responsabilidade manter o controle dos conteúdos hospedados dentro da AWS, além de garantir que seus dados de acesso na plataforma sejam confidenciais e caso sejam compartilhados para terceiros ambos os lados devem ter ciência. (AWS, 2021).

Em casos em que o AWS Lambda seja usado junto com outro serviço, as empresas devem estar cientes das permissões concedidas às demais. O usuário responsável pelas permissões deve conceder à empresa terceira apenas o privilégio mínimo, já que em casos como esse as permissões para usuários terceiros precisam ser concedidas de forma manual (OLIVEIRA, 2020).

A segurança de Lambda baseia-se na mecânica de que usuário e plataforma possuam responsabilidades em relação às informações confidenciais pertencentes dentro da plataforma, como exemplo, temos usuários e senhas para acessos, codificações, informações compartilhadas entre Lambda e o usuário que variam de suas necessidades (AWS, 2021).

O Lambda atua com a responsabilidade compartilhada favorecendo ambos os lados, já que cada lado torna-se responsável apenas pelas partes que são suas responsabilidades.

3.2. Valores dos serviços

Seu serviço é considerado muito econômico, pois o seu modelo de precificação é o "pay-as-you-go", ou seja, os clientes pagam apenas pelos recursos utilizados, sendo eles, número de solicitações para as funções configuradas e a duração necessária para realizar a execução do código. Os valores começam a ser calculados a partir do momento em que a função começa a ser executada e só para de calcular a partir do momento em que ela se encerra, o valor é sempre arredondado para um milissegundo mais próximo (AWS, 2021). O usuário ao configurar os recursos que deseja utilizar deverá escolher a quantidade de memória que será necessária para cada função, o que irá determinar a capacidade da CPU e deve também definir o tempo de execução.

As funções podem ser executadas a partir de processadores com arquiteturas x86 ou ARM, e de acordo os valores obtidos na AWS Lambda, “Usando uma arquitetura de processador baseada em ARM projetada pela AWS, oferecem performance de preço até 34% menor em comparação com as funções em execução em processadores x86. Isso se aplica a uma variedade de workloads sem servidor, como processamento de backends da Web e móveis, de dados e de mídia.” (AWS, 2021).

A partir desses dados é possível obter uma amostra dos valores com relação à arquitetura escolhida pelo usuário, como localização foi utilizado “Leste dos EUA (Ohio)”, podendo ocorrer alterações dos valores de acordo com a região.

Região: Leste dos EUA (Ohio)

Arquitetura	Duração	Solicitações
Preço do x86	0,0000166667 USD por cada gigabyte por segundo	0,20 USD por um milhão de solicitações
Preço do ARM	0,0000133334 USD por cada gigabyte por segundo	0,20 USD por um milhão de solicitações

Figura 2. Relação dos valores do AWS Lambda em relação à arquitetura e estão baseados em preços no Leste dos EUA. (AWS, 2021)

E, também, em relação à capacidade de memória que é responsável por definir a tempo de execução para cada função, neste exemplo foi utilizado “Leste dos EUA (Ohio)”.

Os valores disponíveis são referentes à arquitetura x capacidade de memória, as quais serão selecionadas através da necessidade de cada usuário. Nas imagens não foram disponibilizados todos os tipos de memória que o AWS Lambda oferece para os usuários.

Preço do x86 | Preço do ARM

Região: Leste dos EUA (Ohio)

Memória (MB)	Preço por 1 milissegundo
128	0,0000000021 USD
512	0,0000000083 USD
1.024	0,0000000167 USD
1536	0,0000000250 USD
2048	0,0000000333 USD
3072	0,0000000500 USD
4096	0,0000000667 USD
5120	0,0000000833 USD

Figura 3. Relação dos valores do AWS Lambda em relação à capacidade de memória desejada pelo usuário. A arquitetura utilizada é a x86. Os valores estão baseados em preços no Leste dos EUA (AWS, 2021).

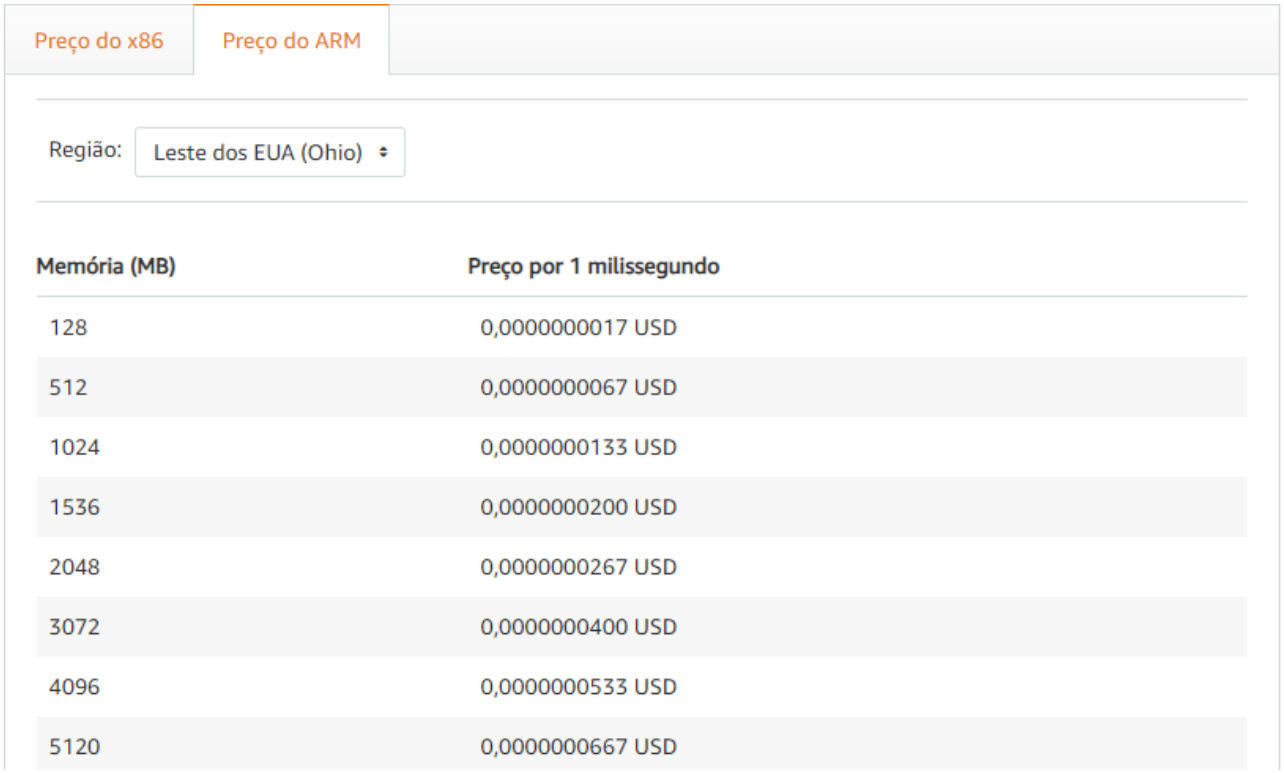

Figura 4. Relação dos valores do AWS Lambda em relação à capacidade de memória desejada pelo usuário. A arquitetura utilizada ARM. Os valores estão baseados em preços no Leste dos EUA. (AWS, 2021)

Além desses custos citados acima existe também a 'simultaneidade provisionada', isso significa que o usuário paga apenas pela quantidade de simultaneidade solicitada e pelo seu tempo de configuração e os valores acima entram apenas quando o limite de execução é ultrapassado. E em casos que ocorra transferência de dados será cobrado os valores por demanda de tarefas.(AWS, 2021)

Quando se fala sobre os valores de uma determinada plataforma nova no mercado, pode-se acabar gerando alguma insegurança para muitos dos usuários, já que para muitas plataformas é necessário contratá-las realizando o seu pagamento, sem ao menos ter a possibilidade de realizar alguns testes, para realmente entender como funciona a plataforma e se ela irá se encaixar dentro das atividades que uma determinada empresa irá precisar. Porém, com AWS Lambda o usuário pode realizar os testes desejados pagando apenas por um valor baixo, ou seja, ele irá pagar apenas o que utilizará sem a necessidade de contratar o serviço. Em alguns casos é possível realizar alguns testes de forma gratuita, para isso é necessário verificar na documentação se as funções que precisará executar se encaixam nos requisitos.

Os valores acima foram apresentados para tranquilizar e informar aos usuários que ainda possuem algum receio, devido a algumas experiências de que a plataforma possui valores consideravelmente baixos, apesar dos valores apresentados estarem em dólar. Com a grande necessidade de economia de muitos usuários e empresas a computação em nuvem chegou para realizar essa tarefa, pois além de possuírem valores extremamente baixos não necessitam de servidores mantendo apenas os em nuvem, sendo assim também acaba não tendo a necessidade de manutenção dos mesmos, diminuindo ainda mais os custos.

3.3. Tempo de execução

O tempo de execução do Lambda pode ser escolhido pelo próprio usuário de acordo com a função selecionada. Ele é gerado junto com a Amazon Linux ou Amazon Linux 2. Para realizar o cálculo do tempo de execução o Lambda envia a função para o denominado "ambiente de execução", que é considerado seguro e, a partir de lá, consegue gerenciar os recursos necessários para executar a função. O ambiente de execução é responsável por administrar os recursos que são responsáveis por realizar uma determinada função, fornecendo suporte para o tempo de execução e determinadas extensões externas (AWS, 2021).

Na imagem a seguir é possível visualizar de maneira clara, como funcionam as comunicações dentro de Lambda para determinar o tempo de execução de uma função. As extensões utilizam o API extensões para se comunicarem, o Lambda utiliza a API Runtime, e as extensões recebem todas as mensagens que contenham logs da função. (AWS, 2021).

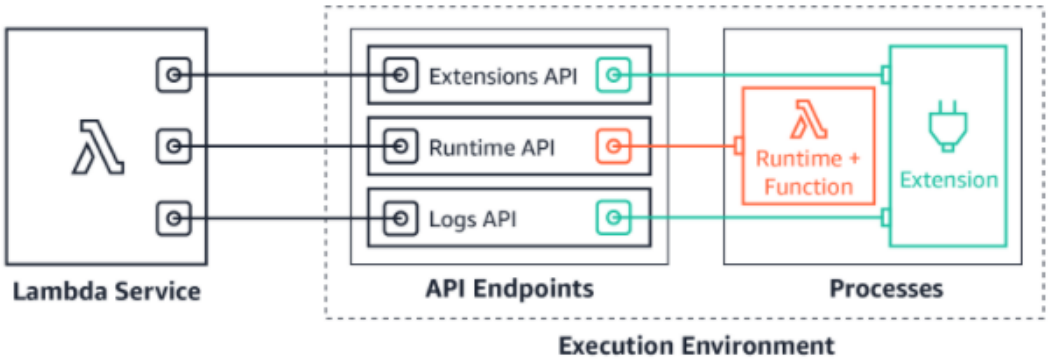

Figura 5. O processo entre as comunicações dentro do Lambda (AWS, 2021).

O tempo de execução pode rodar por várias vezes e só parar quando atingir a sua vida útil. As funções possuem um ciclo de vida que contém três etapas: inicial (*Init*), invocação (*Invoke*) e a final (*Shutdown*).

- Init: Ocorre criação do ambiente de execução ou descongelamento, as seguintes tarefas que são executadas nessa fase são: Inicialização das extensões (*Extension init*); Tempo de execução da inicialização (*Runtime init*) e a execução da função de código estático (*Function init*). Essas fases garantem que todas as extensões concluam as suas tarefas já configuradas antes que o código seja executado (AWS, 2021).
- Invoke:O Lambda envia os eventos para dentro de cada extensão e seu tempo de execução, fazendo com que a função seja executada dentro do tempo limite, sendo esse, já pré definido pelo usuário (AWS, 2021).
- Shutdown: Na fase final o Lambda envia eventos para as extensões informando que serão encerradas (AWS, 2021).

E em casos em que o usuário deseje utilizar outras linguagens que não estão nas configurações automáticas do AWS Lambda, podem ser utilizados tempo de execução personalizados (AWS, 2021).

O tempo de execução ilustrado abaixo é referente a utilização do python, a mesma linguagem que será utilizada para desenvolvimento do “Hello World” no decorrer do artigo.

Python runtimes

Nome	Identificar	SDK AWS para Python	Sistema operacional	Arquiteturas
Python 3.9	`python3.9`	boto3-1.18.55 botocore-1.21.55	Amazon Linux 2	x86_64, arm64
Python 3.8	`python3.8`	boto3-1.18.55 botocore-1.21.55	Amazon Linux 2	x86_64, arm64
Python 3.7	`python3.7`	boto3-1.18.55 botocore-1.21.55	Amazon Linux	x86_64
Python 3.6	`python3.6`	boto3-1.18.55 botocore-1.21.55	Amazon Linux	x86_64
Python 2.7	`python2.7`	boto3-1.17.100 botocore-1.20.100	Amazon Linux	x86_64

Figura 6. Tempo de execução para a Linguagem de Phyton, listando variações de sistema operacional x arquitetura (AWS, 2021).

De modo geral o tempo de execução do Lambda é consideravelmente rápido podendo executar determinadas funções em questão de milissegundos, utilizando diferentes linguagens de programação e sistemas operacionais. O Lamba consegue diferenciar uma função que demora mais tempo do que outras que demoram menos, e a partir disso ele é capaz de fazer com que as funções sejam executadas de forma paralela, para que não ocorra uma fila na execução impedindo que tarefas rápidas, levam um grande tempo para serem executadas devido às demandas (AWS, 2021).

A utilização do Lamba não é recomendada para usuários que desejam realizar grandes funções, visto que o tempo limite de execução do Lambda é de 15 minutos, em casos que o tempo ultrapasse o usuário terá que repartir em partições menores executando apenas uma por função, ao invés de executar todas apenas uma vez, sendo que algumas dessas partições podem rodar simultaneamente, porém isso dependerá do Lambda (MACHADO, 2020).

4.Desenvolvimento de um programa simples - "Hello Word em AWS LAMBDA"

Nessa seção será abordado como criar uma função Lambda sendo ela, “Hello World” através do console. Os exemplos a seguir foram retirados da página *‘Execute um aplicativo “Hello, World!” sem servidor’* (LAMBDA, 2021), fornecida pela AWS como tutorial para inicialização dos conceitos básicos sem a necessidade de gerenciamento de servidores.

Antes de conseguir acessar a plataforma Lambda foi necessário criar uma conta gratuita dentro da AWS. É necessário verificar na plataforma quais serviços são fornecidos gratuitamente.

O algoritmo apresentado a seguir mostra o passo a passo que é necessário realizar antes de chegar na parte de testes do Lambda.

Configurações Iniciais da AWS Lambda

Passo 01: Criar uma conta no AWS Lambda.

Passo 02: Acessar o console da AWS.

Passo 03: Selecionar AWS Lambda.

Passo 04: Criar uma função.

Passo 05: Em "criar uma função" selecionar "Usar um esquema".

Passo 06: Na barra de pesquisa procurar por "hello-world-python".

Passo 07: Após selecionar "hello-world-python" clicar em configurações.

Passo 08: Configurando as funções: Inserir o nome da função que desejar.

Passo 09: Selecionar papel de execução "Criar uma execução a partir da política de AWS templates".

Passo 10: Salvar as informações inseridas na etapa anterior.

Passo 11: Configurar o tempo de execução da função que será definido a partir da linguagem que irá ser utilizada. Foi utilizado o "Python 3.7".

Passo 12: Definir também o manipulador. Foi utilizado "lambda function.lambda handler".

Passo 13: Selecionar a arquitetura. Foi utilizado o x86_64.

Passo 14: Salvar as informações inseridas na etapa anterior.

Passo 15: Editar as configurações básicas.

Passo 16: Inserir uma descrição, essa é uma etapa opcional.

Passo 17: Inserir a quantidade de memória que será necessária para realizar a função. Foi utilizado 128 MB.

Passo 18: Definir qual será o tempo limite para a execução da função. Foi utilizado 3 segundos.

Passo 19: Salvar as informações inseridas na etapa anterior.

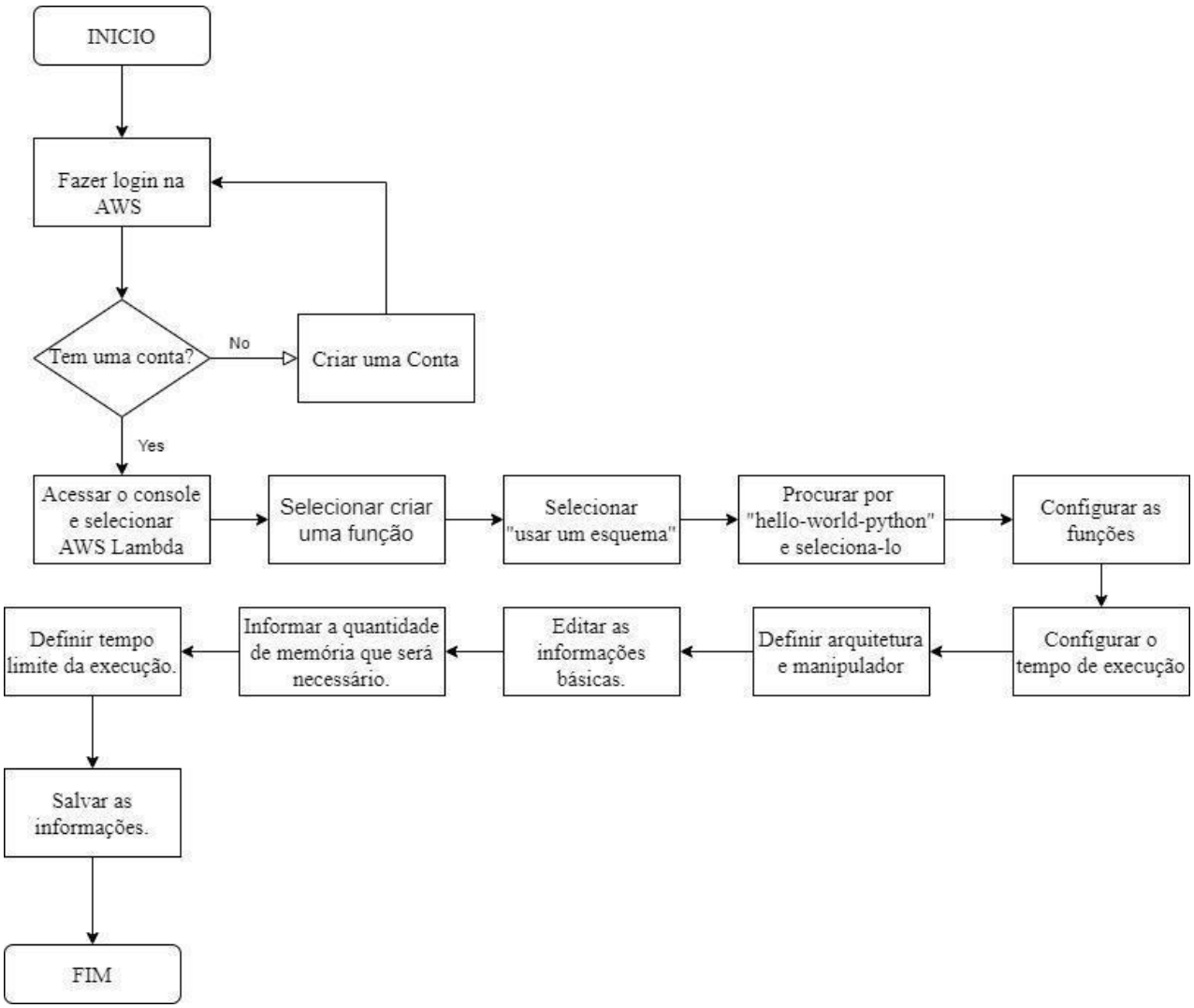

Figura 7. Fluxograma com o passo a passo para configurar uma função dentro da AWS Lambda (Elaborado pela autora, 2021).

Após realizar o registro na plataforma, e as devidas configurações o usuário deverá seguir o seguinte algoritmo e fluxograma para realização dos testes:

Início da realização dos testes:

Passo 01: Abrir a origem do código. O código já estará preenchido com as configurações da "hello-world-python".

Passo 02: Abrir “configurar evento de teste”.

Passo 03: Selecionar "Criar novo evento de teste".

Passo 04: Definir Modelo de evento. Foi selecionado "hello world".

Passo 05: Definir nome do evento. Pode ser definido pelo usuário.

Passo 06: Inserir os valores que deseja que apareça no console.

Passo 07: Salvar as informações definidas na etapa anterior.

Passo 08: Clicar em realizar o teste.

Passo 09: Após isso, será exibida a informação de sucesso, junto com a resposta desejada "Hello, world".

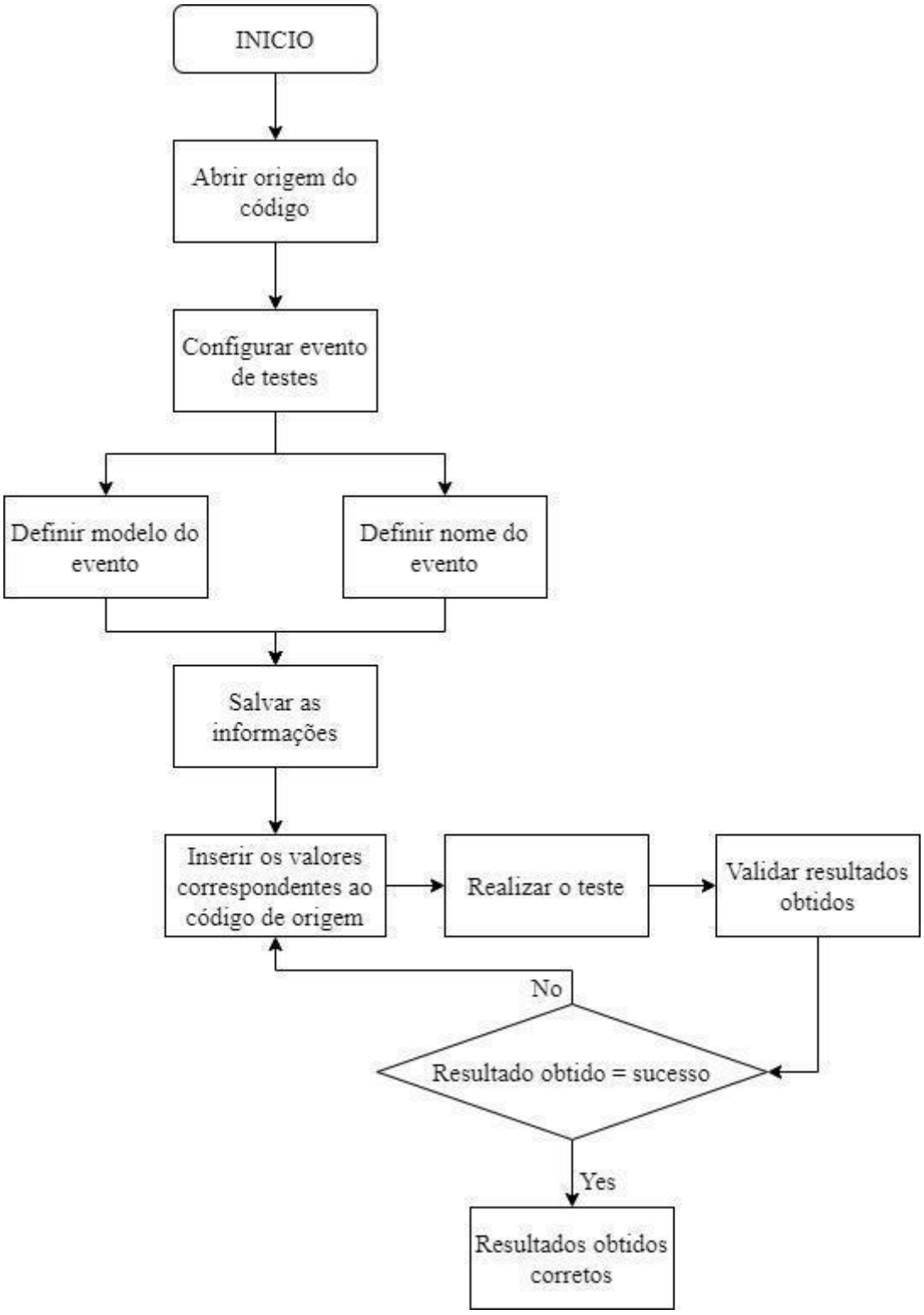

Figura 8. Fluxograma com o passo a passo para a realização dos testes dentro da AWS Lambda (Elaborado pela autora, 2021).

Todos os passos listados acima foram seguidos e realizados com auxílio do tutorial, as imagens abaixo foram realizadas dentro da plataforma AWS Lambda, durante a realização da função.

Segue de forma detalhada todo o processo listado acima:

Ao acessar AWS Service, selecionar: Todos os serviços> Computação -> Lambda e assim abrir o console do Lambda.

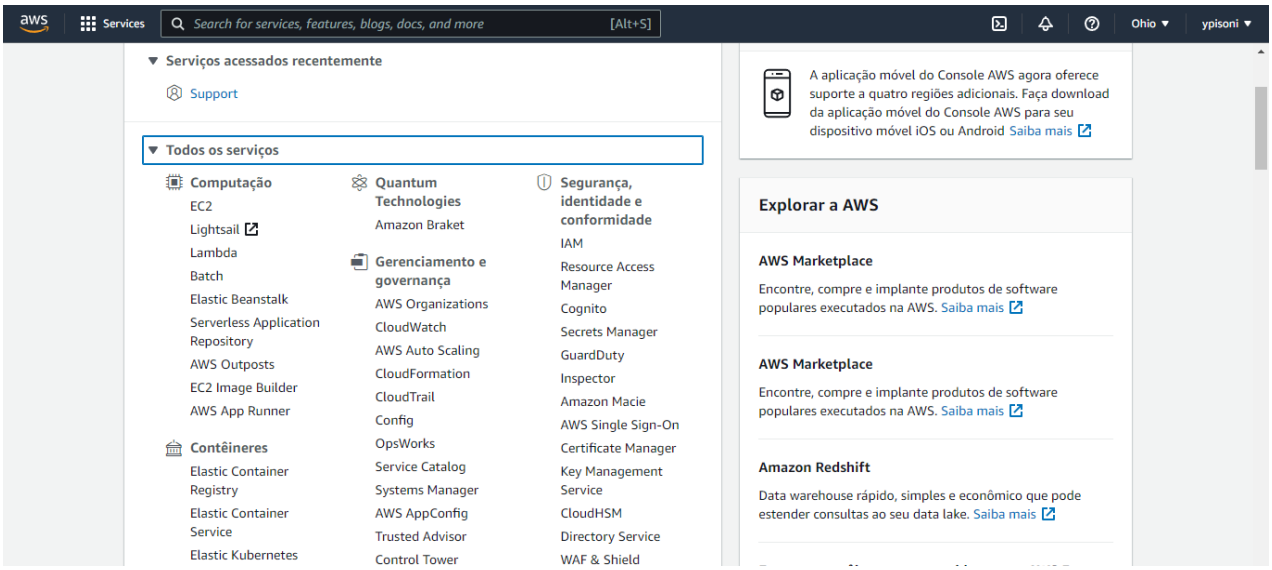

Figura 9. Console AWS, onde existem os serviços oferecidos pela AWS (Desenvolvido pela autora, 2021).

Após clicar em "Lambda", o usuário será redirecionado para a página responsável pelo desenvolvimento das funções do Lambda, a página aparece com as funções zeradas. Porém, como foram realizados alguns testes é possível visualizar algumas funções já criadas.

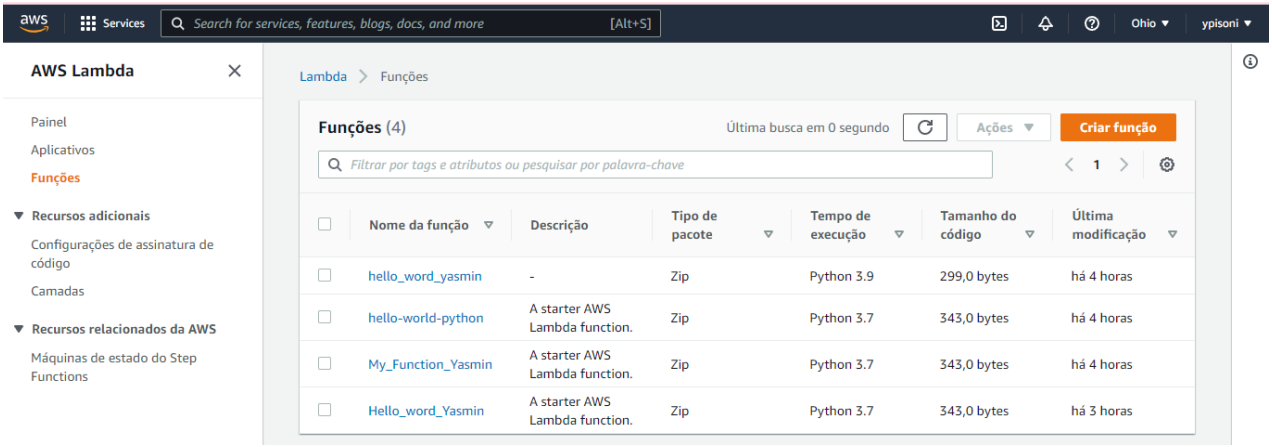

Figura 10. Exibe as funções criadas dentro da plataforma AWS Lambda (Elaborado pela autora, 2021).

Figura 11. Exibe as opções para criação de uma função no AWS Lambda (Desenvolvido pela autora, 2021).

Após a página Lambda> Funções ser aberta o usuário deverá clicar em criar a função. Ao abrir a página de Lambda> Funções -> "Criar função", deverá ser selecionado a seguinte ação "Usar um esquema", foram sugeridos alguns esquemas já pré-definidos pelo Lambda, o esquema que iremos utilizar é o "*hello-world-python*".

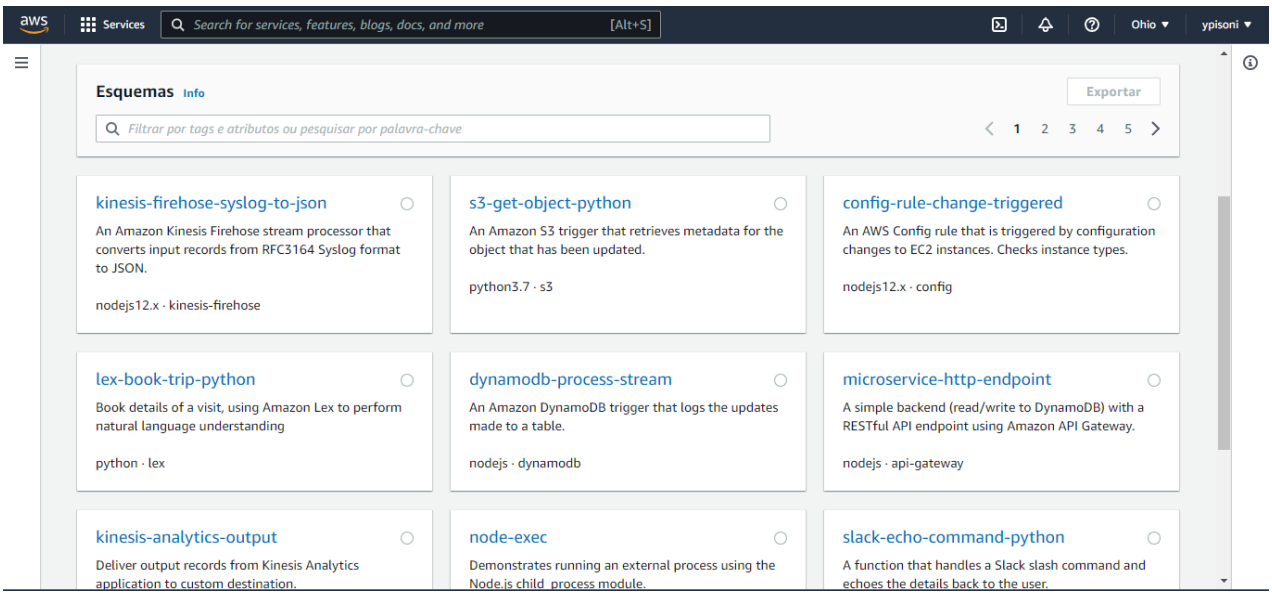

Figura 12. Exibe as opções já pré-definidas de esquema já configurados pela AWS (Desenvolvido pela autora, 2021).

Ao selecionar o usuário deverá escolher "configuração" e será direcionado para a página onde é possível realizar as configurações básicas, como: nome da função e papel da execução, nessa etapa o usuário também criará uma função do IAM escolhendo uma que definirá as permissões para a sua função que o AWS Lambda pode estar assumindo para invocar a função lambda.

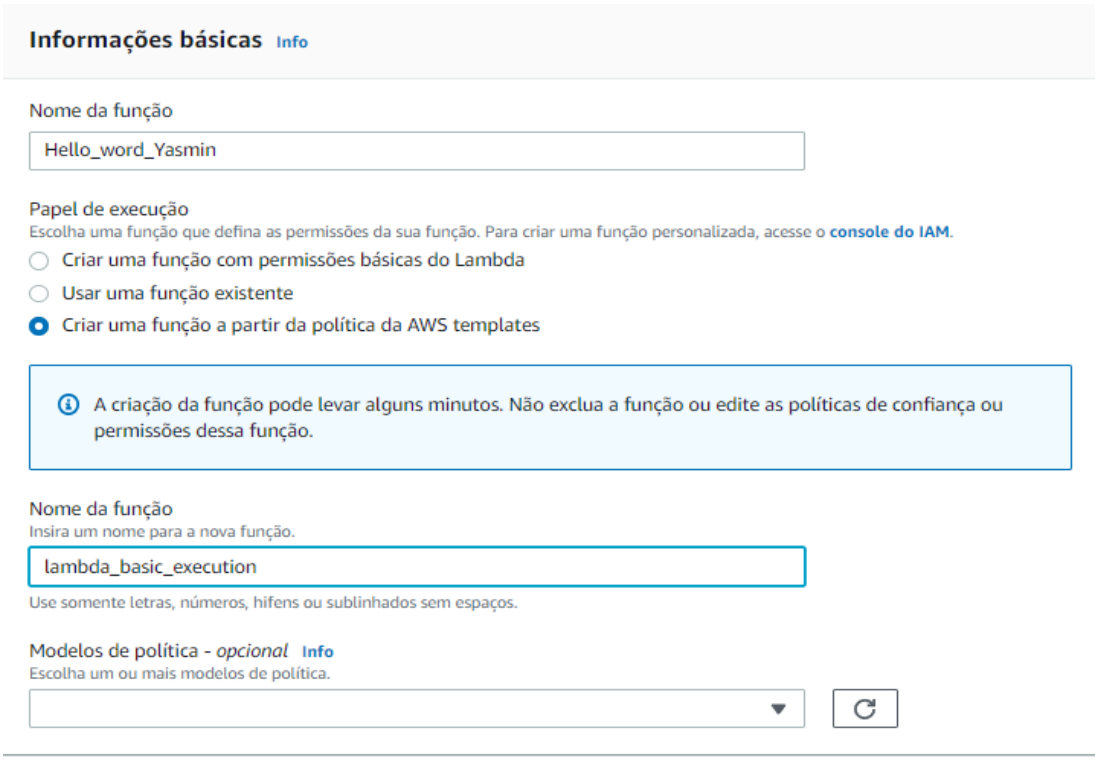

Figura 13. Exibe as opções para as configurações básicas da função Lambda (Desenvolvido pela autora, 2021).

O próximo passo do usuário é definir o tempo de execução, onde deve ser definido o código de função, e nesse caso foi utilizado a linguagem Python 3.7. O usuário também pode especificar um manipulador em que o AWS Lambda possa começar a executar seu código, para o nosso exemplo foi utilizado o "lambda_function.lambda_handler"

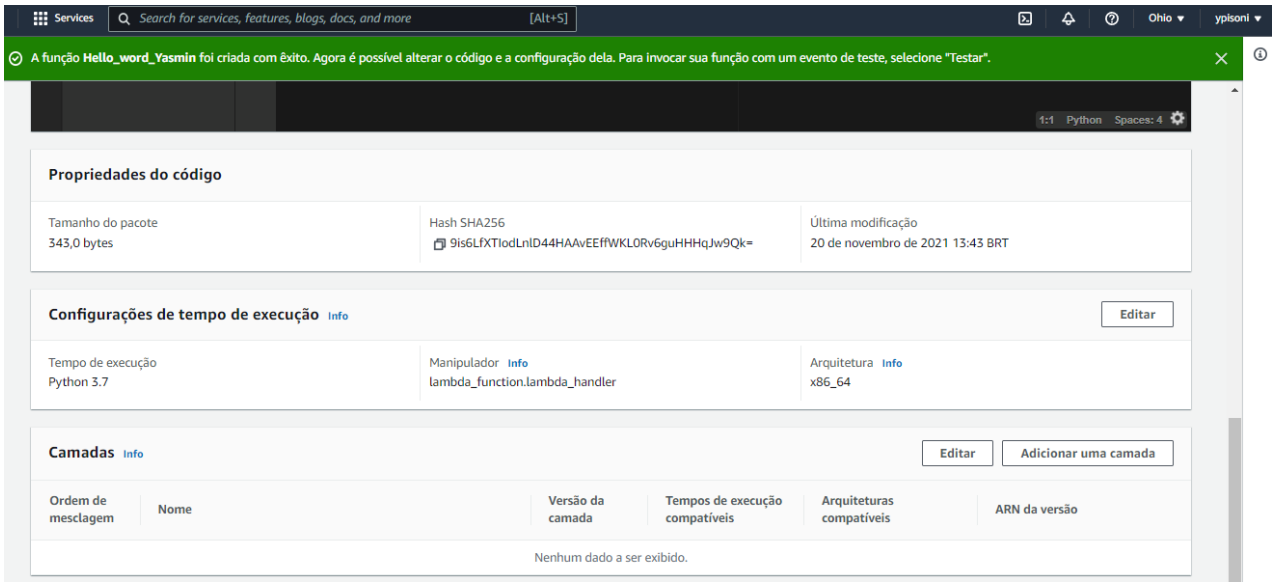

Figura 14. Exibição das configurações básicas da função Lambda (Desenvolvido pela autora, 2021).

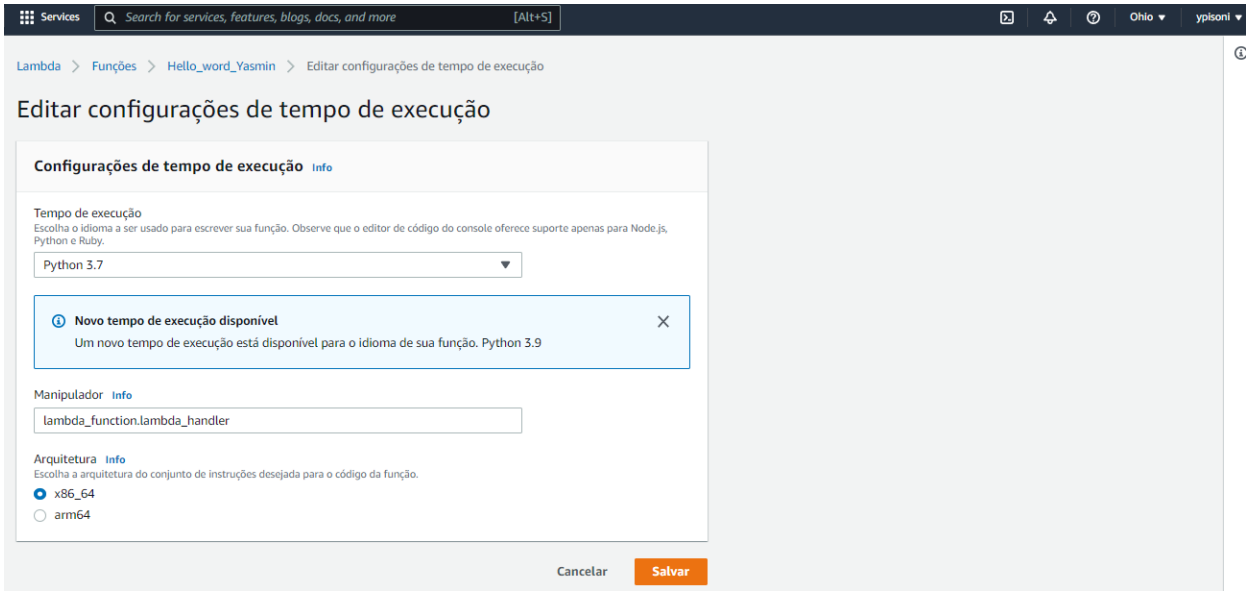

Figura 15. Edição da configuração do tempo de execução da função lambda (Desenvolvido pela autora, 2021).

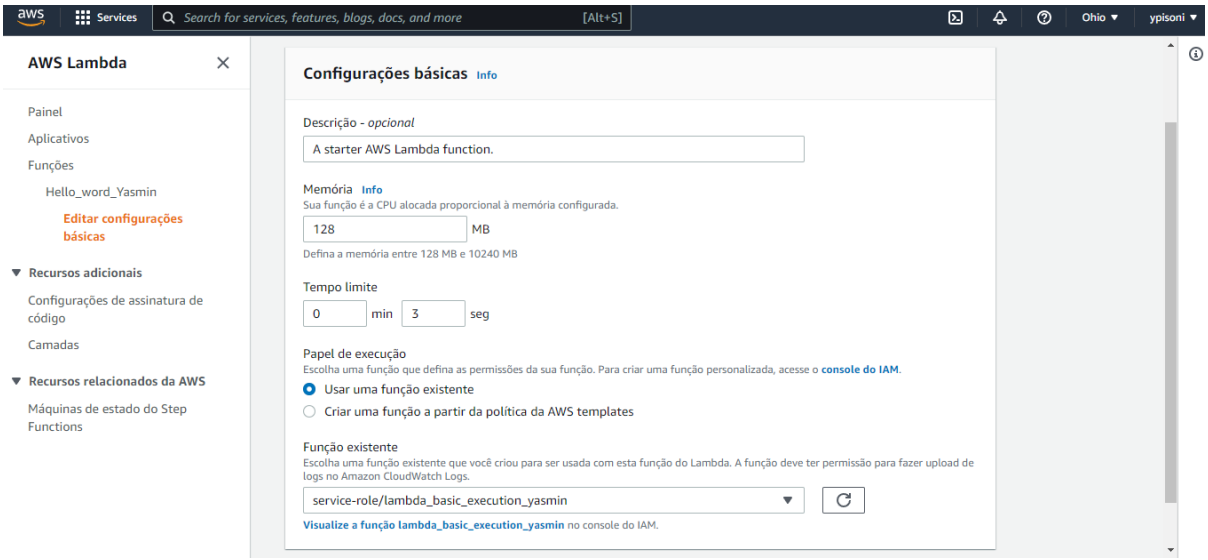

Figura 16. Edição da configuração do tempo de execução da função Lambda (Desenvolvido pela autora, 2021).

Após a finalização das configurações básicas o usuário pode passar a realizar testes. O código retirado através da “AWS Lambda” utilizando JSON é:

```
import json

print('Loading function')
```

```
def lambda_handler(event, context):
    print("value1 = " + event['key1'])
    print("value2 = " + event['key2'])
    print("value3 = " + event['key3'])
    return event['key1']
```

Os valores utilizados foram:

```
{
  "key1": "hello, word!",
  "key2": "Yasmin Pisoni",
  "key3": "value3"
}
```

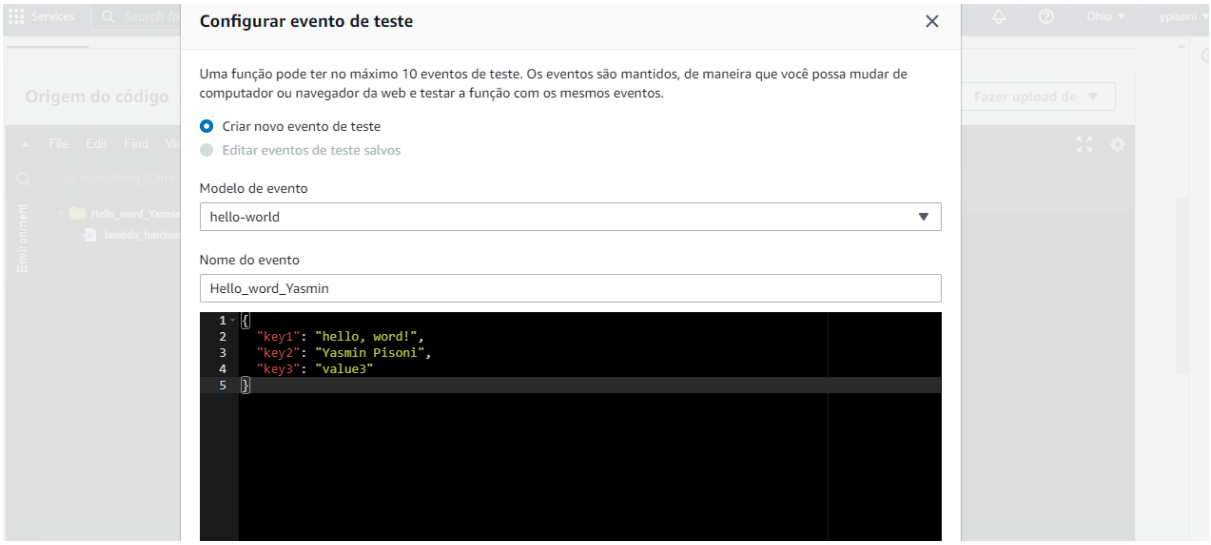

Figura 17. Configuração do evento de teste (Desenvolvido pela autora, 2021).

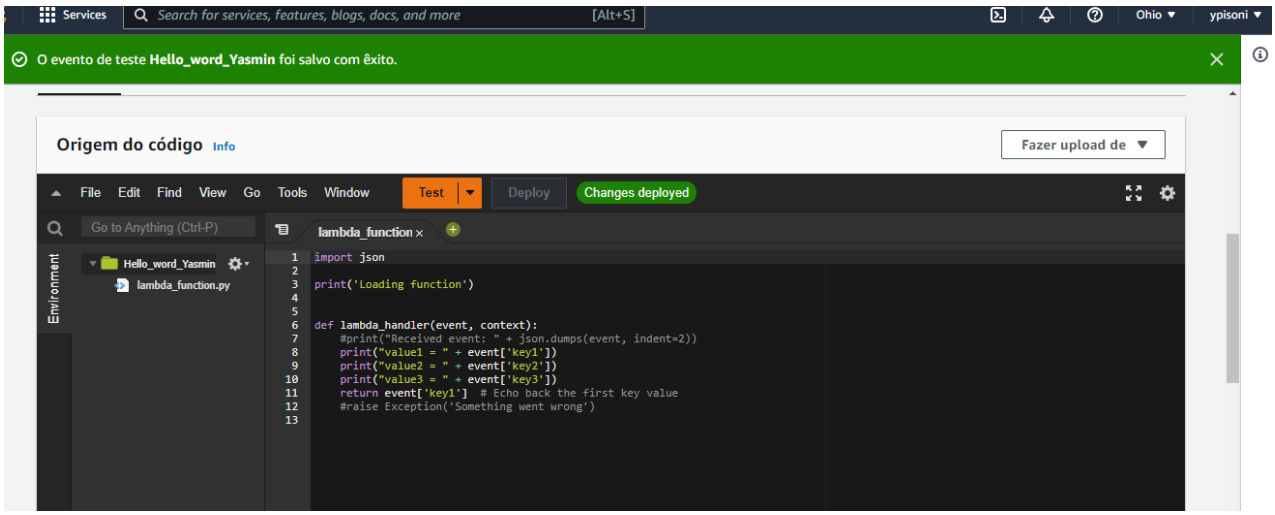

Figura 18. Realização do evento de teste da função lambda (Desenvolvido pela autora, 2021).

Após a realização do teste foi obtido o seguinte resultado:

Nome do evento:

Hello_word_Yasmin

Resposta:

"hello, word!"

Função

START RequestId: f401c829-c5ce-440f-9504-837ad5e4a342 Version: $LATEST

value1 = hello, word!

value2 = Yasmin Pisoni

value3 = value3

FIM RequestId: f401c829-c5ce-440f-9504-837ad5e4a342

REPORT RequestId: f401c829-c5ce-440f-9504-837ad5e4a342

Duração: 2.05 ms

Duração faturada: 3 ms

Tamanho da memória: 128 MB

Memória máxima utilizada: 37 MB

Request ID

f401c829-c5ce-440f-9504-837ad5e4a342

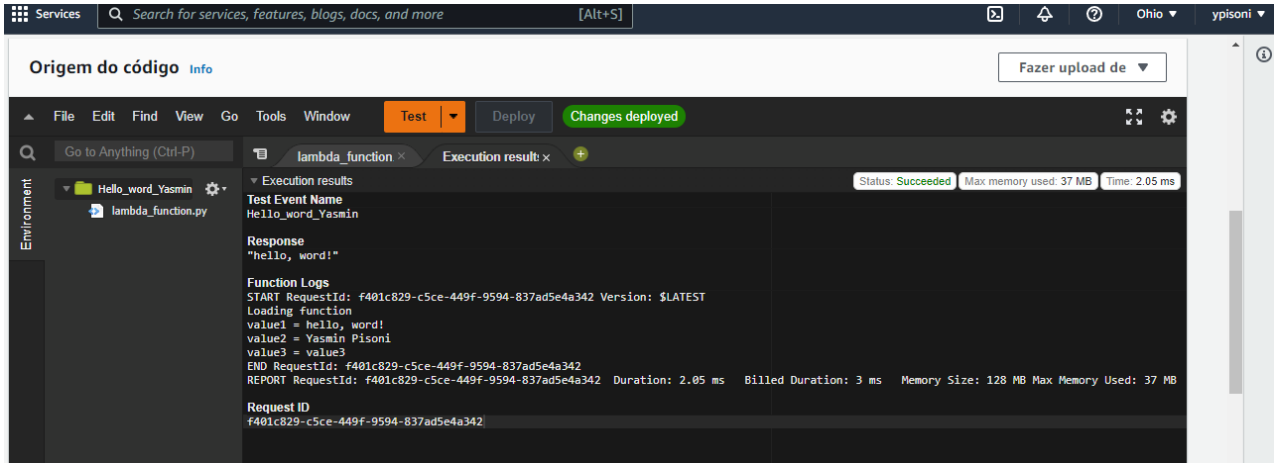

Figura 19. Execução bem sucedida conforme observado na tela do console (Desenvolvido pela autora, 2021).

Os resultados obtidos acima representam que a função foi executada corretamente. Ao finalizar a criação da função é possível verificar através do painel do AWS Lambda quantas funções o usuário possui, sendo elas já criadas anteriormente e qual a quantidade de memória que está sendo utilizada pelas funções.

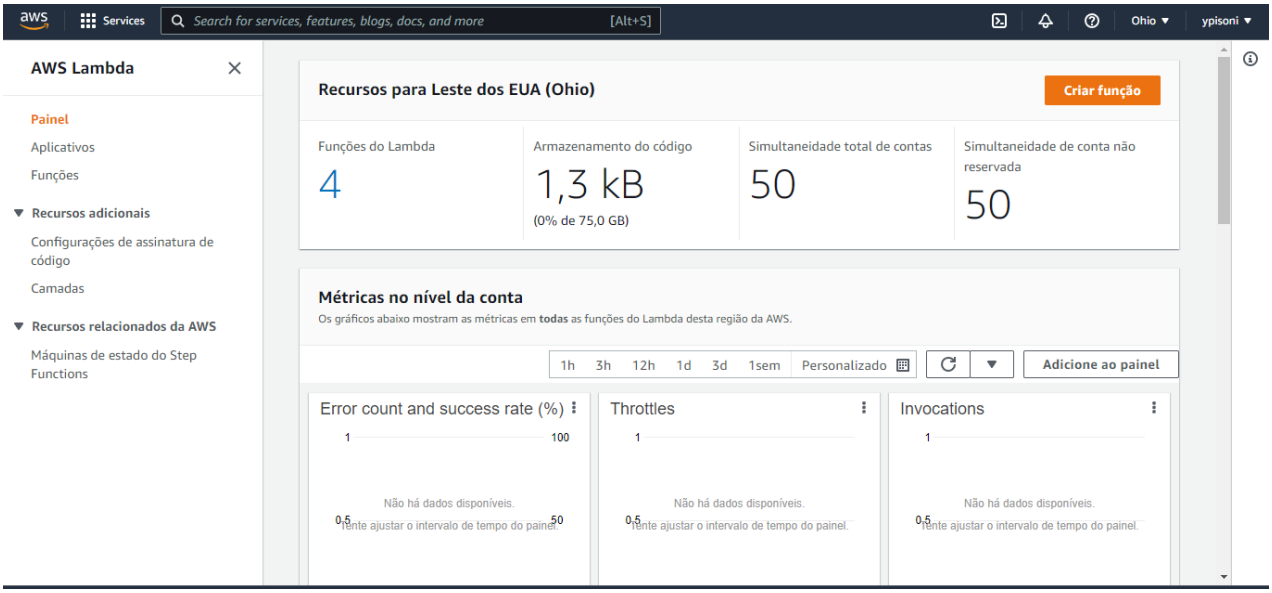

Figura 20. Painel do AWS Lambda (Visão do painel obtida pela autora, 2021)

O tempo utilizado entre a criação de uma conta até o desenvolvimento teve um total de 40 minutos, para cada função desenvolvida. Após estar com o login realizado, o tempo de duração foi em torno de 10 minutos; isso sendo o tempo de desenvolvimento e de execução.

Através do desenvolvimento realizado foi possível notar o quão rápida é a ferramenta a partir do início da criação da função até o momento da execução sendo que nesse caso a execução da função foi de 2.05 milissegundos.

5.CONSIDERAÇÕES FINAIS

Com a chegada da era digital, o mundo corporativo sofreu uma série de mudanças, e a sociedade como um todo. Diversas novidades tecnológicas começaram a fazer parte do cotidiano das pessoas e empresas. É importante reafirmar, que dentre esses avanços, o investimento em infraestrutura em nuvem chegou para criar um diferencial no âmbito empresarial, as soluções de armazenamento sem servidor se mostram cada vez mais estratégicas e possuem resultados mais eficientes. A partir dessas análises foi possível notar que o grande desafio das plataformas, nesse sentido, é alcançar o máximo possível de empresas, pois, a tecnologia se renova constantemente e o que ontem era novidade, hoje se torna obsoleto. Com isso fica claro, quando se faz uma avaliação sobre as empresas renomadas, como a Amazon, que está no 'ranking' das mais rentáveis e pioneira em armazenamento em nuvem.

Os recursos oferecidos pela Amazon Web Services Lambda, são capazes de atingir uma grande gama de empresas, pois além de ser mais rentável, devido a sua forma de pagamento que é realizada através do '*Pay-Per-Use'* , ou seja, as empresas pagam apenas pelas tarefas executadas, sendo esses valores relativamente baixos em relação a empresas que fazem as cobranças a partir de pacotes pré-estabelecidos além de possuir vários tutoriais e micro serviços gratuitos, sendo o formato gratuito o utilizado para realizar o desenvolvimento de uma função.

Com o Lambda as empresas não necessitam de diversos profissionais focados em hardware, uma vez que os servidores digitais não estarão mais presentes. As empresas conseguirão executar as suas funções de maneira rápida, visto que o tempo limite de execução do Lambda é de quinze minutos, porém a maioria das execuções levam apenas milissegundos para serem concluídas.

Sendo notável que através das análises, o uso desses recursos torna-se bastante eficaz, pode-se gerar grandes impactos positivos para as grandes e pequenas empresas, pois ao aderirem aos recursos tecnológicos da AWS Lambda, estarão obtendo a melhor forma de minimizar tarefas e custos e de maximizar os lucros.

REFERÊNCIAS BIBLIOGRÁFICAS

ADZIC, Gojko; CHATLEY, Robert. Serverless Computing: Economic and Architectural Impact. p. 01-06, 2017. Disponível em: https://www.doc.ic.ac.uk/~rbc/papers/fse-serverless-17.pdf. Acesso em: 6 set. 2021.

AWS. Otimização da economia empresarial com arquiteturas sem servidor. 2017. Disponível em: https://d1.awsstatic.com/whitepapers/pt_BR/optimizing-enterprise-economics-serverless-architectures.pdf. Acesso em: 26 nov. 2021.

AWS, AWS Lambda, 2021. Disponível em: https://aws.amazon.com/pt/lambda/. Acesso em: 27 out. 2021.

AWS, LAMBDA. Execute um aplicativo "Hello, World!" sem servidor. (2021) Disponível em: https://aws.amazon.com/pt/getting-started/hands-on/run-serverless-code/. Acesso em: 16=5 nov. 2021.

AWS, LAMBDA. Introdução AWS Lambda. (2021) Disponível em: https://docs.aws.amazon.com/lambda/latest/operatorguide/intro.html. Acesso em: 10 ago. 2021.

AWS, LAMBDA. AWS Lambda execution environment. (2021) Disponível em: https://docs.aws.amazon.com/lambda/latest/dg/runtimes-context.html. Acesso em: 12 nov. 2021.

CHOUDHARY, Brijesh et al. Case Study: Use of AWS Lambda for Building a Serverless Chat Application. O que é Serverless Computing? Entenda mais! Disponível em: https://www.researchgate.net/publication/338391132_Case_Study_Use_of_AWS_Lambda_for_Building_a_Serverless_Chat_Application. Acesso em: 06 nov. 2021.

CLEANCLOUND, FaaS: Um novo conceito de arquitetura sem servidor. 2018. Disponível em: https://cleancloud.io/faas-um-novo-conceito-de-arquitetura-sem-servidor/. Acesso em: 29 nov. 2021.

FEDAK, Vladmir. Serverless Applications with AWS Lambda: 5 Use Cases. 13 ago. 2018. Disponível em: https://dzone.com/articles/serverless-applications-with-aws-lambda-5-use-case. Acesso em: 8 out. 2021.

HOSSEINI, MEHRSHAD et al. AWS Lambda Language Performance. 12 nov. 2019. Disponível em: https://gupea.ub.gu.se/handle/2077/62454. Acesso em: 26 nov. 2021.

IBGE, Uso de internet, televisão e celular no Brasil. Disponível em: https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html. Acesso em: 23 nov. 2021.

IBM, Cloud Education. What is Serverless Computing? (2021). Disponível em: https://www.ibm.com/cloud/learn/serverless. Acesso em: 10 nov. 2021.

IPENSE, O que é Serverless Computing? Entenda mais! (2020). Disponível em: https://www.ipsense.com.br/blog/o-que-e-serverless-computing-entenda-mais/. Acesso em: 16 nov. 2021.

KLIMOVIC, Ana et al. Pocket: Elastic Ephemeral Storage for Serverless Analytics. In: Pocket: Elastic Ephemeral Storage for Serverless Analytics. (2020). Disponível em: https://www.usenix.org/conference/osdi18/presentation/klimovic. Acesso em: 21 out. 2021.

KOHN , Stephanie. O que é e para quem é indicado a Serverless Computing. [S. l.], 8 ago. 2018. Disponível em: https://canaltech.com.br/infra/o-que-e-e-para-quem-e-indicado-a-serverless-computing-119782/. Acesso em: 19 nov. 2021.

LAMBDA CONSOLE, AWS. Console AWS Lambda. In: LAMBDA CONSOLE. (2021) Disponível em: https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/discover. Acesso em: 16 nov. 2021.

MACHADO, Gabriel. O que é AWS Lambda?. 2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-aws-lambda. Acesso em: 27 nov. 2021.

MELL, P. and GRACE, T. (2011) The NIST Definition of Cloud Computing. National Institute of Standards and Technology Special Publication, 53, 1-7. Disponível em: https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/ReferencesPapers.aspx?ReferenceID=1788248. Acesso em: 27 de agosto de 2021.

MULLER , Lisa et al. A break in theA Traffic Analysis on Serverless Computing Based on the Example of a File Upload Stream on AWS Lambda clouds: towards a cloud definition. MDPI, p. 01-29, 10 dez. 2020. Disponível em: https://www.mdpi.com/2504-2289/4/4/38. Acesso em: 29 out. 2021.

OLIVEIRA, Alfredo. Protegendo Pontos Fracos em Arquiteturas Serverless: Riscos e Recomendações. Trend Micro Research, [S. l.], p. 1-32, 8 ago. 2020. Disponível em: https://canaltech.com.br/infra/o-que-e-e-para-quem-e-indicado-a-serverless-computing-119782/. Acesso em: 19 nov. 2021.

SREEKANTI, Vikram et al. Cloudburst: Stateful Functions-as-a-Service. 14 jan. 2020. Disponível em: https://arxiv.org/abs/2001.04592. Acesso em: 20 nov. 2021.

SOFTLINE, IaaS, PaaS e SaaS: entenda os modelos de nuvem e suas finalidades. In: SOFTLINE, Softline. IaaS, PaaS e SaaS: entenda os modelos de nuvem e suas finalidades. 31 out. 2017. Disponível em: https://brasil.softlinegroup.com/sobre-a-empresa/blog/iaas-paas-saas-nuvem. Acesso em: 15 out. 2021.

SRAVAN KUMAR, Nunna et al. Monitoring and Handling the Errors in Serverless Applications Using AWS Lambda. A JOURNAL OF COMPOSITION THEORY, p. 1-8, 1 jan. 2020. Disponível em: http://www.jctjournal.com/gallery/24-april-2020.pdf. Acesso em: 20 out. 2021.

VAQUERO, Luis M et al. A break in the clouds: towards a cloud definition. ACM DIGITAL LIBRARY, p. 01-06, 31 dez. 2008. Disponível em: https://dl.acm.org/doi/abs/10.1145/1496091.1496100. Acesso em: 27 ago. 2021.

AGRADECIMENTOS

Em primeiro lugar, a Deus. Ao meu orientador do trabalho de conclusão de curso: Prof. Dr. Carlos Eduardo Câmara, por compartilhar os seus conhecimentos. E aos meus familiares e amigos, que sempre me apoiaram no decorrer do curso e me mantiveram determinada.

APLICANDO ARQUITETURA DE MICROSSERVIÇOS NO DESENVOLVIMENTO DE SOFTWARE

MICROSSERVICE ARCHITECTURE IN SOFTWARE DEVELOPMENT

Roni da Cruz SILVA

ronidacruz@gmail.com

Ciências da Computação, Centro Universitário Padre Anchieta

Prof. Esp. Rodrigo SAITO

rodrigok@anchieta.br

Ciências da Computação, Centro Universitário Padre Anchieta

Resumo

O desenvolvimento de sistemas evolui a cada dia. Novas tecnologias surgem constantemente e é preciso estar atualizado sempre para não ser superado. A arquitetura de software busca organizar os elementos, suas formas e a lógica envolvida independente da tecnologia escolhida. Neste trabalho, analisamos os princípios da Arquitetura de Microsserviços e como ela pode ajudar a resolver a necessidade de construir sistemas que possam ser escalados, que tenham tolerâncias a falhas, que possam implementar mais de uma linguagem de programação e que se adaptem a evolução constante que vivemos atualmente. Este trabalho apresenta os principais padrões de desenvolvimento que podem ser usados em microsserviços e demonstra de forma prática como aplicar estes conceitos.

Palavras-Chave

Arquitetura de Software; Padrões de Desenvolvimento; Microsserviços.

Abstract

Systems development evolves every day. New technologies are constantly appearing and you must always be up to date in order not to be surpassed. Software architecture aims to organize the elements, their formats and the logic involved regardless of the chosen technology. In this article, we analyze the principles of Microservice Architecture and how it can help solve the need to build systems that can scale, with fault tolerance, that can implement more than one programming language and adapt to the constant evolution we live in today. This work presents the main development patterns that can be used in microservices and demonstrates in a practical way how to apply these concepts.

Keywords

Software Architecture; Design Patterns; Microservices.

INTRODUÇÃO

No processo de desenvolvimento de software, é necessário identificar quais são os problemas que devem ser resolvidos pelo sistema e determinar qual será a arquitetura utilizada para resolver estes problemas.

Conforme Perry e Wolf (1992) a arquitetura de software consiste de três componentes: elementos, formato e lógica. Os elementos podem ser processamentos, dados e elementos de comunicação. O formato é definido pelas propriedades e as relações dos elementos. A lógica fornece a base para a arquitetura de acordo com as restrições do sistema que na maioria das vezes derivam dos requisitos do sistema.

Com a evolução da tecnologia, alguns modelos de arquitetura de software surgiram como cliente-servidor, aplicação monolítica, desenvolvimento em camadas, arquitetura orientada a serviço, etc. De acordo com Chris Richardson (2019), as arquiteturas tradicionalmente conhecidas como monolíticas podem oferecer simplicidade para desenvolver, implantar e escalar a aplicação. No entanto, quando a aplicação se torna um pouco maior, essa abordagem pode apresentar várias desvantagens como: o código fonte pode ser difícil de entender, a aplicação pode não ter os limites dos módulos bem definidos, as alterações necessárias podem trazer perda de qualidade no código, o ambiente de desenvolvimento pode ser lento devido ao tamanho do código fonte, o tempo de inicialização pode ser alto, as implantações de atualizações podem causar inconvenientes ao deixar o sistema indisponível em algum momento, o sistema escala apenas de forma vertical, é complicado dividir responsabilidades entre as equipes e causa dependência de determinada tecnologia utilizada.

Segundo o site da Red Hat, a arquitetura orientada a serviço serve pra resolver essas questões, pois estrutura as aplicações em serviços distintos e reutilizáveis que se comunicam entre si.

Neste contexto, surge o termo microsserviço, que para James Lewis e Martin Fowler (25/03/2014), esta arquitetura é uma abordagem para desenvolver um sistema como um conjunto de pequenos serviços, cada um executando em seu próprio processo e se comunicando através de mecanismos leves, geralmente API's baseadas no protocolo HTTP. Estes serviços são construídos de acordo com uma necessidade do negócio e implantados de forma independente e automática. Há um mínimo de centralização e estes serviços podem ser escritos em diferentes linguagens de programação e usar diferentes tecnologias de armazenamento de dados. Eles também afirmam que não há uma definição formal do estilo arquitetural de microsserviços, mas é possível descrever quais características são comuns. Dessa forma, nem todas as arquiteturas de microsserviços possuem todas as características implementadas, mas se espera que a maioria das arquiteturas de microsserviços possuam a maioria destas características.

COMPARAÇÃO DE ARQUITETURAS

Para entender melhor a arquitetura de microsserviços, vamos comparar com a arquitetura monolítica, analisando suas vantagens e desvantagens.

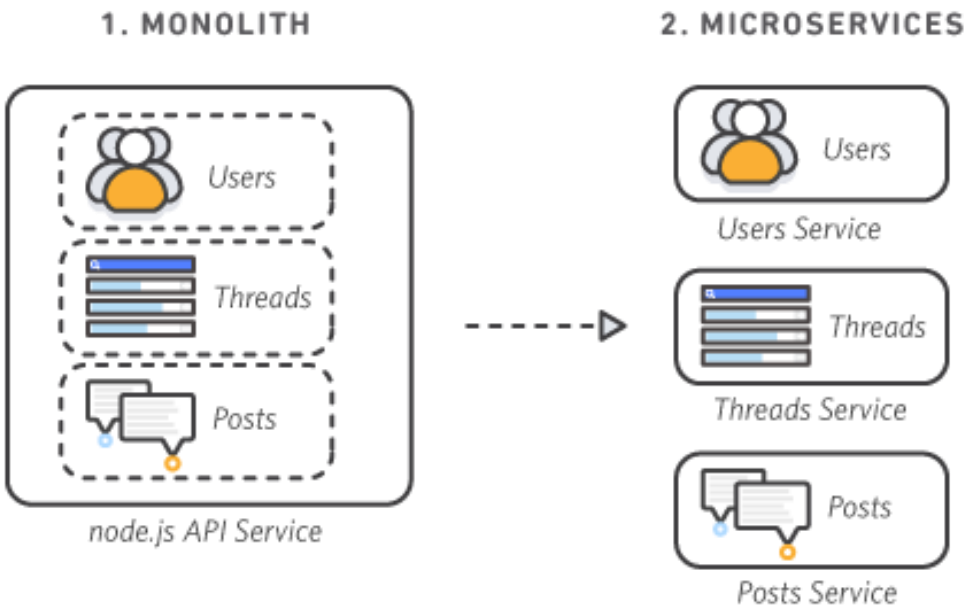

Figura 24 *- Arquitetura Monolítica versus Microsserviços*

https://d1.awsstatic.com/Developer%20Marketing/containers/monolith_1-monolith-microservices.70b547e30e30b013051d58a93a6e35e77408a2a8.png

Arquitetura Monolítica

Conforme Penna (13/05/2020), "arquitetura monolítica é um sistema único, não dividido, que roda em um único processo, uma aplicação de software em que diferentes componentes estão ligados a um único programa dentro de uma única plataforma".

"Com as arquiteturas monolíticas, todos os processos são altamente acoplados e executam como um único serviço. Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada." (AMAZON AWS, 2021)

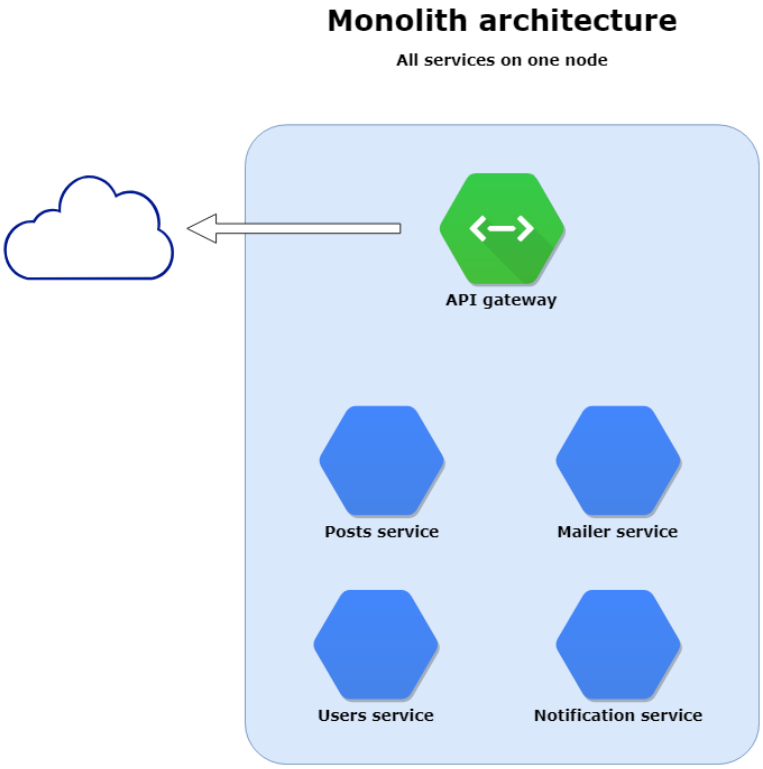

Figura 25 *- Arquitetura Monolítica*

https://moleculer.services/docs/0.14/clustering.html

Vantagens da Arquitetura Monolítica

De acordo com Chris Richardson, inicialmente, a arquitetura monolítica pode apresentar as seguintes vantagens:

- Simplicidade no desenvolvimento - *IDE's* e outras ferramentas de desenvolvimento estão empenhadas em desenvolver uma única aplicação.
- Facilidade para fazer mudanças radicais - É possível alterar o código, os esquemas de banco de dados, empacotar e implantar o sistema. Praticamente tudo em um único lugar.
- Facilidade para criar rotinas de testes - Os desenvolvedores podem criar testes integrados que inicia a aplicação e executa todo um processo do começo ao fim.
- Facilidade na implantação - Tudo que um desenvolvedor precisa fazer é transportar um único pacote para o ambiente produtivo.
- Facilidade para escalar - Nesta arquitetura, é possível escalar de forma vertical, criando novas instâncias de todo o sistema. (RICHARDSON, 2019, p. 4)

Desvantagens da Arquitetura Monolítica

Com o passar do tempo, a aplicação tende a ficar maior e mais complexa e podem surgir as seguintes desvantagens:

- Complexidade - Uma aplicação simples, com o passar do tempo, pode receber constantemente novas funcionalidades, o que torna o código-fonte cada vez mais amplo. O tamanho da equipe também tende a aumentar e é necessário um esforço maior para organização e gerenciamento.

Isso implica que a aplicação é muito grande e difícil de entender completamente. Cada novo ajuste necessário torna-se difícil e consume tempo. Para piorar a situação, toda essa complexidade pode tornar-se um ciclo vicioso, pois se o código é difícil de entender o desenvolvedor pode não aplicar os ajustes da melhor forma possível, aumenta a complexidade. O sistema gradualmente pode tornar-se uma monstruosa, incompreensível, grande bola de lama.

- Lentidão no desenvolvimento - Além de ter que lidar com a complexidade do sistema, os desenvolvedores enfrentam lentidão em tarefas do dia-a-dia. A aplicação gigante sobrecarrega a interface de desenvolvimento e o processo de *building* é bem mais demorado. Além disso, por ser tão grande, a aplicação também demora para iniciar, afetando toda a produtividade da equipe.
- O processo de implantação é longo e árduo - Com o crescimento desordenado da aplicação, cada alteração no sistema precisa passar por um processo longo, lento e doloroso para chegar no ambiente produtivo. A equipe precisa fazer *deploys* em intervalos maiores, tipicamente aos finais de semana para reduzir o impacto.
  Como muitos desenvolvedores estão trabalhando no mesmo código, frequentemente o sistema está com alterações em andamento, então o gerenciamento de mudanças torna-se mais complexo.
- As rotinas de testes também são mais longas, e a cada mudança o sistema demora para ter estabilidade.
- Dificuldade para escalar - Os diversos módulos de um sistema podem ter requisitos bem diferentes de memória, processamento e armazenamento. Como todos estão implantados no mesmo servidor, a configuração do servidor precisa atender a todos. Isso pode causar desperdícios de recursos para tarefas que não precisam e falta de recursos para atividades que requerem mais poder do servidor.
- Desafios de uma entrega confiável - Outra grande desvantagem é a dificuldade para entregar confiança. A rotina de testes pode não cobrir todo o código e bugs podem acabar sendo descobertos em produção. A aplicação também carece de mecanismos de tolerância a falhas porque todos os módulos estão sendo executados no mesmo processo. Frequentemente, uma falha em um módulo específico pode comprometer todos os outros módulos.
- Finalmente, o sistema está fortemente acoplado à tecnologia escolhida inicialmente. A arquitetura monolítica torna mais difícil adotar novas linguagens de programação ou diferentes frameworks. É muito arriscado inovar, por isso, é comum trabalhar em ambientes extremamente obsoletos. (RICHARDSON, 2019, p. 4-7)

Arquitetura de Microsserviços

Conforme a IBM, microsserviços é um estilo de arquitetura, no qual grandes e complexos aplicativos de software são compostos por um ou mais serviços. Os microsserviços podem ser implantados independentemente uns dos outros e estão fracamente acoplados. Cada um desses microsserviços se concentra em completar uma tarefa apenas e faz essa tarefa muito bem. Em todos os casos, essa tarefa representa uma pequena capacidade do negócio. (IBM REDBOOKS, 2015)

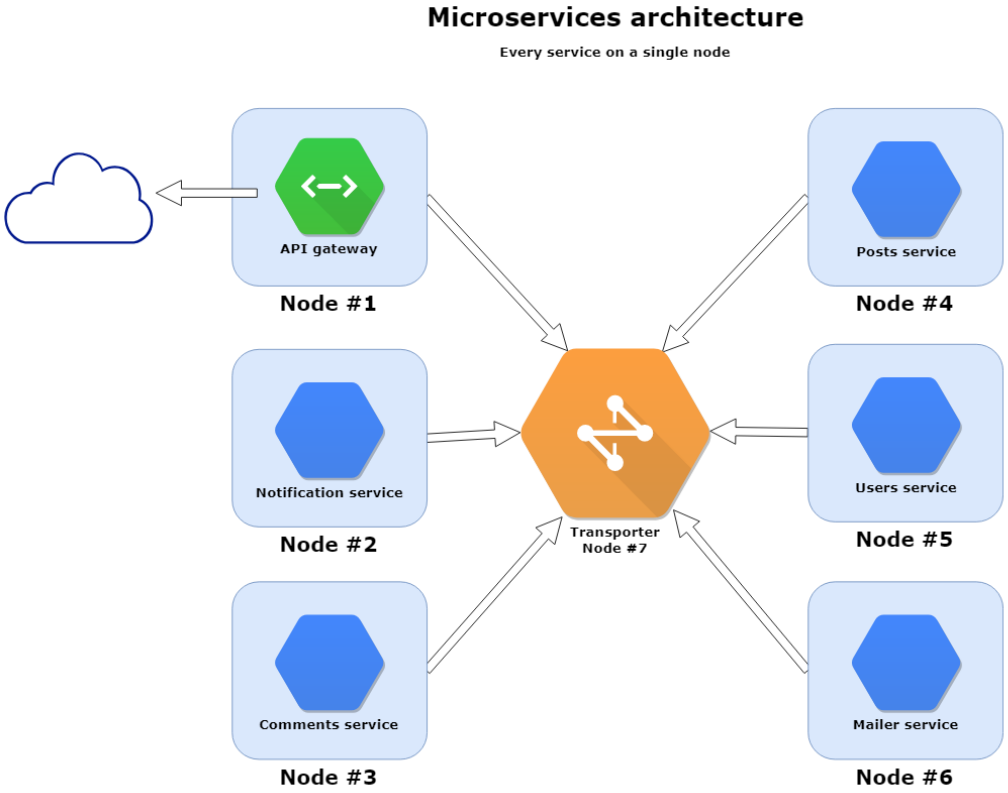

Figura 26 *- Arquitetura de Microsserviços*

https://moleculer.services/docs/0.14/clustering.html

Vantagens da Arquitetura de Microsserviços

Chris Richardson apresenta os seguintes benefícios da arquitetura de microsserviços:

- Permite a entrega e implantação contínuas de sistemas grandes e complexos - Entrega e implantação contínua faz parte do *DevOps*, um conjunto de práticas para a entrega rápida, frequente e confiável de software. Isso é facilitado pela arquitetura de microsserviços de três maneiras:
    - Automação de Testes - Como cada serviço é relativamente pequeno, testes automáticos são fáceis de implementar e executar.
    - *Deploy* - Cada serviço pode ser implantado de forma independente dos demais. Se uma alteração local for necessária em um serviço, a equipe responsável não precisa coordenar a implantação com as demais equipes. Dessa forma é muito mais fácil implantar mudanças em produção e de forma mais frequente.
    - Autonomia das equipes - As equipes podem ser organizadas em times menores, onde cada time pode desenvolver, implantar e escalar seus serviços de forma independente.
- Serviços são pequenos e de fácil manutenção - O código menor é mais fácil de ser entendido pelo desenvolvedor. A interface de desenvolvimento fica mais leve, e os serviços iniciam mais rapidamente, aumentando a produtividade.
- Serviços são escaláveis de forma independente - Cada serviço pode ser escalado de acordo com suas necessidades. Além disso, cada serviço pode ser implantado no hardware mais adequado. Isso resulta em redução de custos.

- Isolamento de falhas - A arquitetura de microsserviços possui um melhor isolamento de falhas, já que se um serviço estiver esgotando os recursos, isso deve afetar somente aquele serviço. E se algum serviço estiver fora do ar, os demais ainda podem funcionar.
- Adoção de novas tecnologias - Finalmente, a arquitetura de microsserviços permite experimentar novas tecnologias, já que a equipe pode escolher livremente a linguagem de programação e frameworks que melhor se ajustarem a necessidade. (RICHARDSON, 2019, p. 14-17)

Desvantagens da Arquitetura de Microsserviços

Implementar uma arquitetura de microsserviços possui vários desafios. Para a IBM, é importante analisar as seguintes situações:

- Não iniciar com microsserviços - Se a aplicação é pequena e a equipe não tem experiência com microsserviços, é melhor manter as coisas simples.
- Não planejar microsserviços sem DevOps - Microsserviços geram muitas partículas que precisam ter um fluxo sério de implantação, passando por testes automatizados e implantação e entrega contínua.
- Evitar administrar a própria infraestrutura - Microsserviços introduzem muitos conceitos com o que se preocupar, como banco de dados, serviços de mensagens, servidores que demandam cuidados. É melhor deixar essa parte com empresas que já fazem isso com excelência do que manter uma equipe própria preocupada com isso.
- Não iniciar com muito serviços - Cada serviço exige recursos. Em alguns casos é melhor iniciar um serviço maior e de acordo com o crescimento, dividi-lo, se for necessário.
- Problemas com latência - Criar serviços que dependam de outros pode introduzir problemas de latência. Ferramentas de monitoramento são essenciais para identificar problemas de comunicação entre os serviços. (IBM REDBOOKS, 2015, p. 10-12)

Arquitetura Orientada a Serviços (SOA)

Arquitetura orientada a serviços (SOA) é um tipo de design de software que torna os componentes reutilizáveis usando interfaces de serviços com uma linguagem de comunicação comum em uma rede. Em outras palavras, a arquitetura SOA integra os componentes de software que foram implantados e são mantidos separadamente, permitindo que eles se comuniquem e trabalhem juntos para formar aplicações de software que funcionam em sistemas diferentes. (RED HAT, 27/07/2020)

De uma forma geral, SOA e microsserviços são padrões de desenvolvimento que estruturam um sistema como um conjunto de serviços. Em um nível alto, existe similaridades. Mas olhando de perto, podemos encontrar diferenças importantes:

- Comunicação - SOA tipicamente usa tecnologias robustas e centralizadas para comunicação como SOAP e frequentemente fazem uso de um ESB (*enterprise service bus*) para integração dos serviços. Microsserviços costumam usar protocolos mais leves como REST ou gRPC.
- Dados - SOA geralmente tem um modelo global de dados com banco de dados compartilhados. Na arquitetura de microsserviços, cada serviço possui sua base de dados própria.
- Tamanho - SOA, geralmente é usado para integrar grandes e complexas aplicações. Mesmo que na abordagem de microsserviços nem sempre os serviços sejam extremamente pequenos, eles são quase sempre bem menores. Dessa forma, uma aplicação SOA geralmente consiste em alguns grandes serviços, enquanto uma aplicação baseada em microsserviços normalmente consiste em dezenas ou centenas de serviços menores. (RICHARDSON, 2019, p. 13-14)

Para concluir a comparação, observamos os objetivos. SOA tenta expor seus serviços a qualquer um que queira usá-los. Os microsserviços, alternativamente, são criados com um objetivo muito mais focado e limitado em mente, que é atuar como parte de um único sistema distribuído. (IBM REDBOOKS, 2015, p. 13)

ARQUITETURA DE MICROSSERVIÇOS - CONCEITOS ESSENCIAIS

Conforme explicado por Prates (2021), as empresas tech quando nascem começam a se desenvolver e criar softwares, que são suas soluções para o mercado, muitas vezes desenvolvidas em monólitos. Ao criá-las, ninguém sabe o tamanho exato que a empresa vai alcançar. Aos poucos, com novas soluções implementadas, a companhia começa a crescer, mudar e se adaptar. Surge então a necessidade de migrar o sistema legado para uma arquitetura mais dinâmica, orientada a microsserviços. Mas como fazer? Quebrar uma parte da aplicação e refazer uma parte específica ou recriar todo o sistema? (PRATES, 08/03/2021)

Strangler Pattern

Martin Fowler apresentou, em 2004, uma técnica de conversão que consiste em envolver a aplicação, substituindo seus módulos por microsserviços.

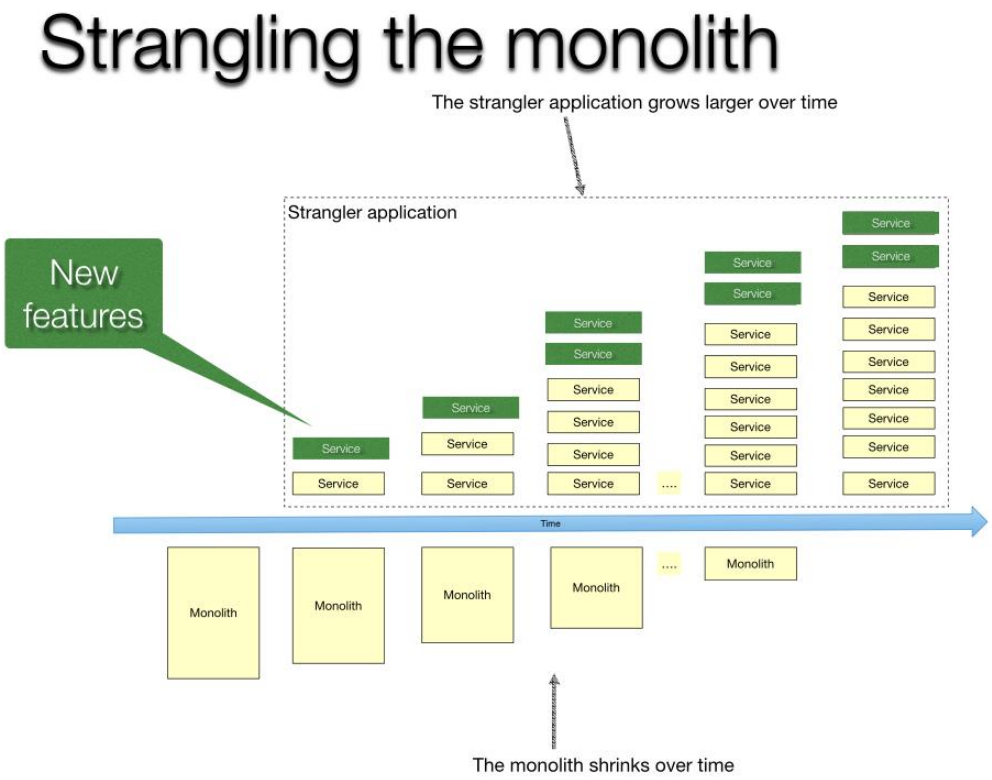

Figura 27 *- Strangler Pattern*

https://microservices.io/i/decompose-your-monolith-devnexus-feb-2020.001.jpeg

O padrão de desenvolvimento *Strangler application* permite modernizar uma aplicação de forma incremental, desenvolvendo uma nova aplicação em volta da aplicação legado, estrangulando aos poucos as funcionalidades. O aplicativo estrangulador consiste em dois tipos de serviços. Primeiro, existem serviços que implementam funcionalidades que residiam anteriormente no monólito. Em segundo lugar, existem serviços que implementam novos recursos. Gradualmente o tamanho do monólito diminui e a aplicação orientada a microsserviços aumenta (RICHARDSON, 2017).

Divisão de Responsabilidades

Com base em Gonçalves (2020), a implementação dos microsserviços consiste em um conjunto de padrões de design conceituais, divididos em categorias como aplicação, redes, infraestrutura, banco de dados, etc. Sendo cada unidade executando dentro de seu próprio processo, descrevendo características de serviços auto-contidos, autônomos e independentes.

A arquitetura de microsserviços propõe descentralizar a responsabilidade dos dados. Cada escopo de transação é tratado pelo microsserviço responsável dentro de seu contexto e limites transacionais adequados, devendo trabalhar com etapas de compensação das operações em mente do início ao fim do processo. A independência contribui para agilidade, trazendo maior liberdade para reagir rapidamente a mudanças e tomar decisões.

Ao desenvolvermos microsserviços, precisamos praticar o design de composição dos componentes com o domínio em mente. Modelos de domínio fazem mais ou menos sentido dependendo do contexto ao qual eles são observados, sendo assim, práticas de *domain-driven design* nos ajudam a compreender e construir modelos mais fiéis ao domínio de negócio.

Isolamento e Independência

Uma instância, que também pode ser chamada de nó, é um simples processo sendo executado numa rede local ou externa. Uma única instância de um nó pode hospedar um ou mais serviços. Dois (ou mais) serviços rodando em um único nó são considerados serviços locais. Eles compartilham recursos de hardware e usam o barramento local para se comunicar entre si, sem latência da rede (módulo de transporte não é usado). Serviços distribuídos através de vários "nós" são considerados remotos. Neste caso, a comunicação é feita através de um módulo de transporte (MOLECULERJS, 2021).

A computação distribuída trata de possibilitar maior disponibilidade e alto poder de escalabilidade quando necessário. Uma das necessidades que teríamos seria a de isolamento entre os componentes, pois ao executarmos cada microsserviço em seu próprio processo dentro de um servidor, adquirimos maior agilidade. Uma das razões para querermos o isolamento (independência) seria para lidar com falhas separadamente. (GONÇALVES, 01/06/2020)

Comunicação entre serviços

A arquitetura de microsserviços estrutura uma aplicação como um conjunto de serviços. Em muitos casos, esses serviços precisam colaborar entre si para responder uma requisição. Dessa forma, é necessário reservar tempo para analisar esse processo de comunicação entre os serviços.

Existem várias tecnologias que podem ser adotadas, como o padrão REST baseado no protocolo HTTP ou gRPC. Também é possível usar sistemas de mensagens assíncronas, baseadas em filas, com diversos formatos de mensagens. Os serviços podem usar formatos baseados em textos como JSON ou XML, ou também protocolos em formatos binários. (RICHARDSON, 2019, p. 65-67)

Estilos de comunicação

Há uma variedade de estilos de comunicação. Uma requisição pode ser processada exatamente por um único serviço ou por vários. E a comunicação também pode ser síncrona, assíncrona ou em sentido único.

- Síncrona: Um cliente de serviço faz uma solicitação a um serviço e espera por uma resposta. O cliente espera que a resposta chegue em tempo hábil. Pode haver um bloqueio de eventos enquanto espera. Este é um estilo de interação que geralmente resulta em serviços fortemente acoplados.
- Assíncrona: Um cliente de serviço envia uma solicitação a um serviço, que responde de forma assíncrona. O cliente não bloqueia enquanto espera, porque o serviço pode levar muito tempo para enviar a resposta.
- Sentido único: Um cliente de serviço envia uma solicitação a um serviço, mas nenhuma resposta é esperada ou enviada.

Ao estabelecer uma comunicação múltipla, o sistema de eventos é aplicado. Um cliente publica uma mensagem que é consumida por zero ou muitos serviços interessados naquele evento. (RICHARDSON, 2019, p. 67-68)

Usando REST

Conforme Silva (2017), REST é um conjunto de regras e padrões, enquanto *RESTful* é a implementação dessas regras em uma API. De acordo com Roy Fielding, um dos idealizadores do modelo arquitetural REST, para que uma API seja considerada *RESTful* esta deve obrigatoriamente seguir um conjunto de regras pré-definidas. Richardson (2019) propôs um modelo que define o nível de maturidade de uma API.

- Nível 0: Os clientes de um serviço de nível 0 invocam o serviço fazendo solicitações HTTP para seu *endpoint* único. No entanto, não existe um padrão.
- Nível 1: Um serviço de nível 1 introduz a ideia de recursos. Para realizar uma ação em um recurso, um cliente faz uma solicitação POST que especifica a ação a ser executada.
- Nível 2: Um serviço de nível 2 usa verbos HTTP para realizar ações: GET para recuperar, POST para criar e PUT para atualizar. Os parâmetros e o corpo da consulta da solicitação, se existir, especifica os parâmetros das ações. As respostas também contêm um código de status HTTP padrão.
- Nível 3: Conhecido como HATEOAS, a ideia básica é que a representação de um recurso retornado por uma solicitação GET contém links para executar ações naquele recurso. (RICHARDSON, 2019, p. 74)

Service registry

Para Peyrott (2015), o *service registry* é um catálogo de serviços preenchido com informações sobre como encaminhar solicitações para cada instância dos microsserviços.

A maioria das arquiteturas baseadas em microsserviços está em constante evolução. Os serviços aumentam e diminuem conforme as equipes de desenvolvimento se dividem, aprimoram, descontinuam e fazem seu trabalho. Sempre que um *endpoint* de serviço muda, o registro precisa saber sobre a mudança. É disso que se trata o registro: é o responsável por publicar ou atualizar as informações de como chegar a cada serviço.

Service discovery

O *service discovery* é a contrapartida do s*ervice registry* do ponto de vista dos clientes. Quando um cliente deseja acessar um serviço, ele deve descobrir onde o serviço está localizado e outras informações relevantes para realizar a solicitação.

Quando este serviço está no lado do cliente ele pode forçar o mesmo a consultar o *service discovery* antes de executar a requisição final.

Quando este serviço está do lado do servidor, o serviço de API Gateway é o responsável por descobrir o *endpoint* exato para cada requisição. (PEYROTT, 02/10/2015)

PADRÕES DE DESENVOLVIMENTO

Uma plataforma de aplicações engloba todas as ferramentas que são independentes de um microsserviço. Essas ferramentas devem ser construídas e dispostas de forma que o time de desenvolvimento não tenha que se preocupar com nada além da lógica da própria aplicação. (OPUS SOFTWARE, 31/05/2021)

Service Mesh

O padrão *service mesh* é uma maneira de controlar como diferentes componentes de uma aplicação compartilham dados entre si. (RED HAT)

Um *service mesh* gerencia toda a comunicação "serviço a serviço" em um sistema de software distribuído. Um *service mesh* fornece a descoberta dinâmica de serviços e gerenciamento de tráfego. Um *service mesh* também suporta a implementação e imposição de requisitos transversais, como segurança e confiabilidade (limitação de taxa, quebra de circuito). Como o *service mesh* está no caminho crítico para todas as solicitações tratadas no sistema, também pode fornecer "observabilidade" adicional, como rastreamento distribuído de uma solicitação, frequência de códigos de erro HTTP, latência global e serviço a serviço. (BRYANT, 30/07/2020)

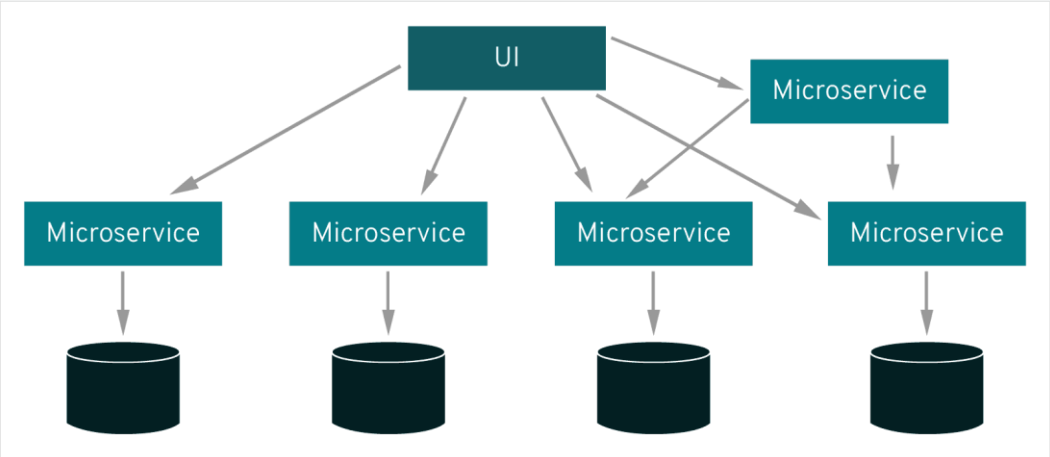

Figura 28 *- Comunicação entre microsserviços*

https://www.redhat.com/cms/managed-files/microservices-1680.png

Tolerância a falhas

Uma consequência do uso de serviços como componentes é que os aplicativos precisam ser projetados para que possam tolerar a falha desses serviços. Como os serviços podem falhar a qualquer momento, é importante ser capaz de detectar as falhas rapidamente e, se possível, restaurar o serviço automaticamente. As equipes de microsserviços esperam ver configurações sofisticadas de monitoramento e registro para cada serviço individual, como painéis que mostram o status ativo ou inativo e uma variedade de métricas operacionais e de negócios relevantes. (LEWIS e FOWLER, 25/03/2014)

Retry

Conforme Prashant (2021), o padrão *retry* possui uma ideia bastante simples. Se quem chama uma ação recebe uma resposta inesperada para uma requisição, o serviço que está chamando envia novamente a requisição para o serviço que está sendo requisitado. Se a requisição falhar devido a problemas ocasionais de rede, problemas de conexão, esse padrão pode ser bastante útil.

No entanto, solicitar novamente de forma indefinida, até receber uma resposta adequada, pode não ser uma boa ideia. Por isso é interessante estabelecer um número limite de re-tentativas. Uma outra técnica é estabelecer um tempo de atraso entre uma retentativa e outra, para dar a chance do sistema se restabelecer. Também é interessante determinar que somente ações que possuem uma chance de responder corretamente numa re-tentativa possam usar este mecanismo.

É importante desenhar processos que possam detectar duplicidade de chamadas e não causar inconsistência de dados.

Circuit Breaker

Para Fowler (2014), a ideia básica por trás do *circuit breaker* é muito simples. Você envolve uma chamada de função protegida em um objeto disjuntor, que monitora as falhas. Uma vez que as falhas atingem um certo limite, o disjuntor desarma e todas as chamadas adicionais para o disjuntor retornam com um erro, sem que a chamada protegida seja feita.

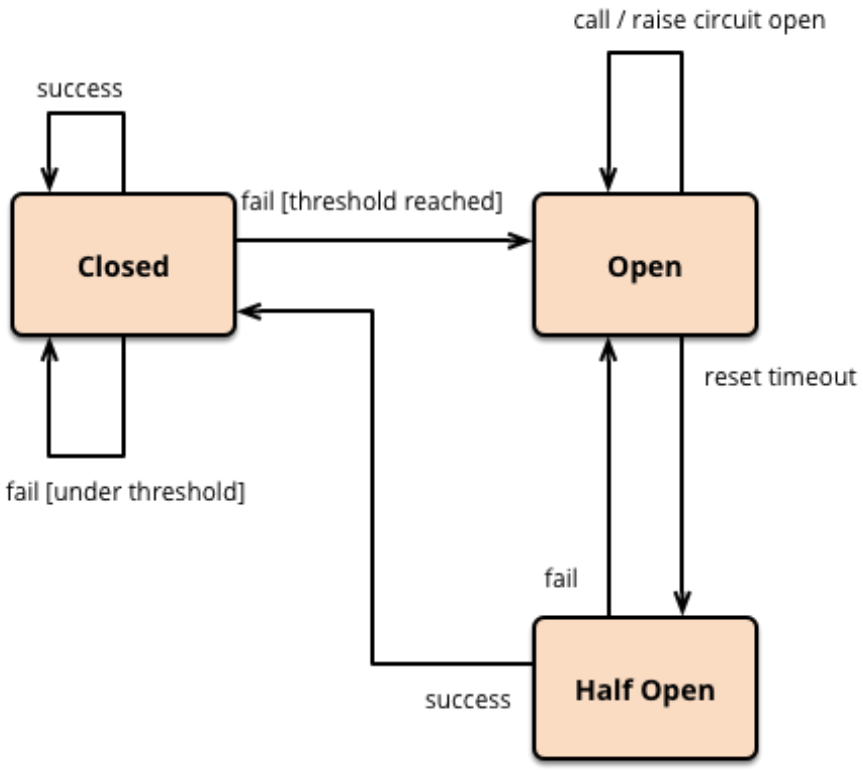

Figura 29 - Circuit Breaker

https://martinfowler.com/bliki/images/circuitBreaker/state.png

Este simples disjuntor evita fazer a chamada protegida quando o circuito está aberto, mas precisaria de uma intervenção externa para reiniciá-lo quando tudo estiver bem novamente. É possível implementar um comportamento de reinicialização automática tentando a chamada protegida novamente, após um intervalo adequado e reiniciando o disjuntor, caso seja bem-sucedido. Criar este tipo de disjuntor significa adicionar um limite para tentar o reset e configurar uma variável para conter a hora do último erro. Também existe um terceiro estado presente - meio aberto - o que significa que o circuito está pronto para fazer uma chamada real, com o intuito de verificar se o problema foi resolvido. No estado semi-aberto, uma chamada de teste é feita, e se for bem sucedida, fechará o disjuntor, se não, resetará o tempo de espera (FOWLER, 06/03/2014).

Bulkhead

Conforme escreveu Selvaraj (2020), um navio é dividido em pequenos compartimentos múltiplos usando anteparos. Anteparos são usados para selar partes do navio para evitar que todo o navio afunde em caso de inundação. Da mesma forma, falhas devem ser esperadas quando projetamos software. O aplicativo deve ser dividido em vários componentes e os recursos devem ser isolados de forma que a falha de um componente não afete o outro.

Usando o padrão *bulkhead*, alocamos um limite para os recursos de um componente específico, evitando consumir todos os recursos do aplicativo desnecessariamente. Nosso aplicativo permanece funcional mesmo sob carga inesperada.

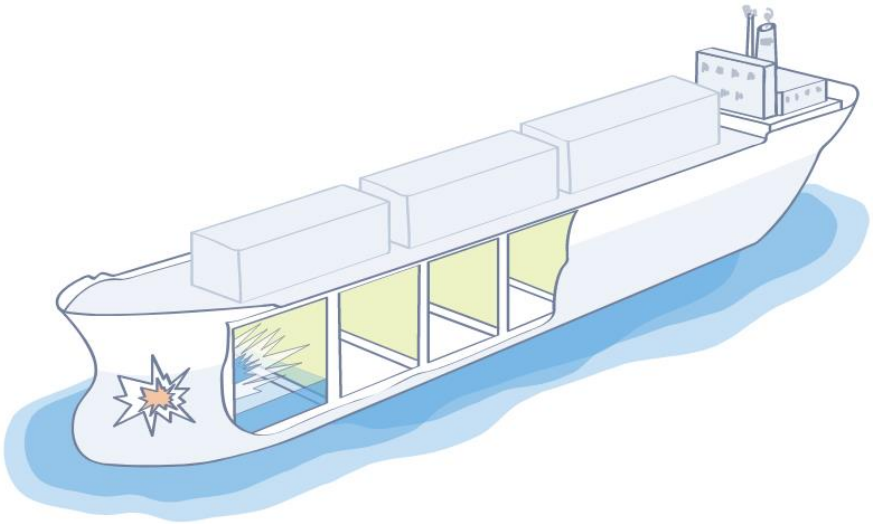

Figura 30 *- Bulkhead*

https://openliberty.io/guides/iguide-bulkhead/html/images/with-bulkhead.svg

Fallback

Para Seeley (2018), o padrão *Fallback* consiste em detectar um problema e, em seguida, executar um caminho de código alternativo. Esse padrão é usado quando o caminho do código original falha e fornece um mecanismo que permitirá ao cliente do serviço responder por meios alternativos. Outros caminhos podem incluir respostas estáticas, respostas em cache ou até mesmo serviços alternativos que fornecem informações

semelhantes. Depois que uma falha for detectada, talvez por meio de um dos outros padrões de resiliência, o sistema pode fazer fallback.

Timeout

De acordo com Selvaraj (2020), o padrão Timeout é uma das técnicas mais simples. Nós experimentamos lentidão intermitente de aplicativos de vez em quando, sem motivos óbvios. Isso pode acontecer com qualquer aplicativo, inclusive em grandes plataformas. Como não é um problema incomum, é melhor levar esse problema de indisponibilidade de serviço em consideração ao projetar os microsserviços.

Dessa forma, temos os seguintes benefícios:

- Fazer os serviços principais funcionarem conforme o esperado, mesmo quando os serviços dependentes não estiverem disponíveis.
- Não esperar uma resposta por tempo indeterminado
- Não bloquear o processamento
- Lidar com problemas de rede oferecendo uma resposta em cache.

Registro de Logs

Segundo Richardson (2019), logs são parte importante quando queremos saber o que há de errado com nossa aplicação. Mas é importante padronizar os registros para que possam ser consultados posteriormente. Existe o padrão *Log Agregation* que mantém todos os registros de todos os serviços em um banco de dados centralizado, que suporta consulta e alertas automáticos. Existem estruturas prontas em várias linguagens que lidam com armazenamento de logs, como por exemplo o combo ELK:

- Elasticsearch - Um banco de dados NoSQL orientado a pesquisa de textos, que é usado para armazenamento
- Logstash - Uma pilha de execução que organiza os logs e os envia para o Elasticsearch
- Kibana - Uma ferramenta de visualização para o Elasticsearch. (RICHARDSON, 2019, p. 369-370)

Tracing

Tracing consiste em atribuir a cada solicitação externa um ID exclusivo, bem como, um registro de como ele flui pelo sistema de um serviço para o outro, em um servidor centralizado, que fornece visualização e análise. Um registro típico contém o nome da aplicação, o ID do rastreamento, tempo de início e fim. (RICHARDSON, 2019, p. 371)

Conforme Ribenzaft (2019), *tracing* é uma forma de identificar e monitorar eventos em aplicativos. Com as informações certas, um rastreamento pode revelar o desempenho de operações críticas. O rastreamento distribuído é uma nova forma de rastreamento que se adaptou melhor a aplicativos baseados em microsserviços. Ele permite que os engenheiros vejam os rastros de ponta a ponta, localizem falhas e melhorem o desempenho geral. Em vez de rastrear o caminho em um único domínio de aplicativo, o rastreio distribuído segue uma solicitação do início ao fim. Existem ferramentas de código aberto que facilitam essa tarefa, como o projeto OpenTelemetry.

Métricas

De acordo com a organização Prometheus, métricas são medidas numéricas, e as séries temporais significam as mudanças que são registradas ao longo do tempo. O que os usuários desejam medir difere de

aplicativo para aplicativo. Para um servidor web, pode ser o número de solicitações, para um banco de dados pode ser o número de conexões ativas ou o número de consultas ativas, etc.

As métricas desempenham um papel importante na compreensão de por que seu aplicativo está funcionando de determinada maneira. Vamos supor que você esteja executando um aplicativo da web e descubra que ele está lento. Precisaremos de algumas informações para descobrir o que está acontecendo com seu aplicativo. Por exemplo, o aplicativo pode ficar lento quando o número de solicitações é alto. Se você tiver a métrica de contagem de requisições, poderá identificar o motivo e aumentar o número de servidores para lidar com a carga.

API GATEWAY

Uma abordagem para o *design* de uma API é os clientes invocarem os serviços diretamente. A princípio, isso parece adequado, mas essa abordagem raramente é usada em uma arquitetura de microsserviços devido às seguintes desvantagens:

- Os serviços são especialistas e podem exigir que os clientes façam várias solicitações para recuperar os dados de que precisam, o que é ineficiente e pode resultar em uma experiência de usuário insatisfatória.
- A falta de encapsulamento causada pelo conhecimento dos clientes sobre cada serviço e seus endpoints torna difícil mudar a arquitetura e a estrutura interna de cada serviço.
- Os serviços podem usar mecanismos de comunicação que não são convenientes ou práticos para os clientes usarem, especialmente aqueles clientes externos.
- Clientes diferentes podem ter largura de banda reduzida, como é o caso dos celulares e, portanto, deveriam ter uma resposta mais personalizada, para otimizar o desempenho. (RICHARDSON, 2019, p. 254-255)

O padrão de desenvolvimento *API Gateway* trata de implementar um serviço que é uma porta de entrada para os clientes externos. O *Api Gateway* lida com requisições simplesmente roteando para o serviço apropriado ou então distribui a chamada para vários serviços. Ao invés de fornecer um estilo de API único, o *API Gateway* pode expor uma API diferente para cada cliente.

Implementar um *API Gateway* oferece os seguintes benefícios:

- Isola os clientes de como o aplicativo é particionado em microsserviços;
- Isola os clientes do problema de determinar a localização das instâncias de serviço;
- Fornece a API ideal para cada cliente;
- Reduz o número de requisições de ida e volta. Por exemplo, o API Gateway permite que os clientes recuperem dados de vários serviços com uma única requisição externa. Menos requisições também significa menos sobrecarga e melhora a experiência do usuário. Um API Gateway é essencial para aplicativos móveis;
- Simplifica o cliente transferindo a responsabilidade de chamar vários serviços para o API Gateway;
- Traduz em um protocolo de API público amigável para a web para quaisquer protocolos usados internamente.

O padrão *API Gateway* tem algumas desvantagens:

- Maior complexidade - o API Gateway é mais uma parte que deve ser desenvolvida, implantada e gerenciada.
- Aumento do tempo de resposta devido a passagem adicional através do API Gateway - no entanto, para a maioria dos aplicativos, o custo dessa passagem é insignificante. (RICHARDSON, 2017)

Para a Red Hat, um *API gateway* faz parte do sistema de gerenciamento da API. Ele intercepta todas as solicitações de entrada e as envia por meio desse sistema, que processa diversas funções necessárias. Sendo esta porta de entrada, a função exata do *gateway* varia de uma implementação para outra. Algumas funções comuns incluem, roteamento, limitação de taxa, faturamento, monitoramento, análise, políticas, alertas e segurança.

Funções essenciais de um gateway

O gateway é responsável pelo roteamento da requisição, composição de API e tradução de protocolo. (RICHARDSON, 2019, p. 260)

Roteamento

Para Richardson (2019), uma das principais funções de um A*PI gateway* é o roteamento de requisições. O *gateway* implementa algumas operações de API roteando requisições para o serviço correspondente. Ao receber uma solicitação, o *gateway* consulta um mapa de roteamento que especifica para qual serviço rotear a solicitação.

Composição de API

Conforme Siahaan (2019), um *API gateway* também fornece o recurso de composição de API, que permite que determinado cliente recupere dados de forma eficiente usando uma única solicitação de API.

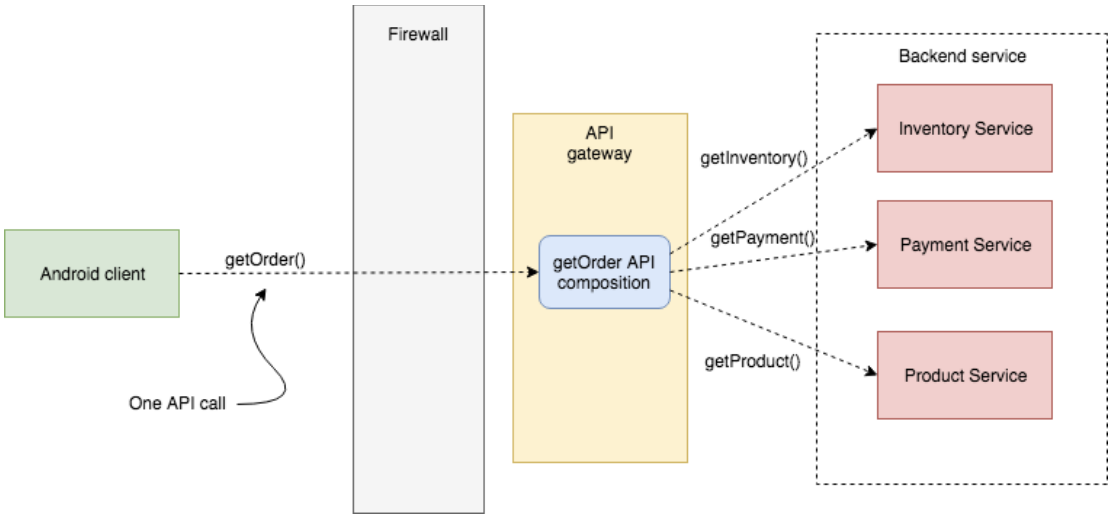

Figura 31 *- Composição de API*

https://miro.medium.com/max/766/1*xco_AbzG10GLrSSqDSqSFg.png

Tradução de protocolo

Um *gateway* também pode realizar a tradução de protocolo. Pode fornecer uma API RESTful para clientes externos, embora os serviços de aplicativo usem uma mistura de protocolos internamente, incluindo REST e gRPC. Quando necessário, a implementação de algumas operações da API traduz entre a API RESTful externa e a API interna baseada em gRPC. (RICHARDSON, 2019, p. 262)

Funções extras de um gateway

Um gateway também pode implementar funções relacionadas com a interceptação das requisições.

- Autenticação: Verifica a identidade do cliente em cada requisição.
- Autorização: Verifica se o cliente está autorizado a executar determinada operação.
- Limite de requisição: Limita o número de requisições por período de um cliente específico ou de todos os clientes.
- Cache: Mantém respostas em cache para reduzir o número de requisições feitas aos serviços.
- Captura de métricas: Colete métricas sobre o uso da API para fins de análise e de faturamento.
- Log de requisições: Registra o log das requisições. (RICHARDSON, 2019, p. 262)

Balanceamento de carga

Os balanceadores de carga são os responsáveis por rotear as requisições que vêm dos clientes para as instâncias do microsserviço requisitado, garantindo que nenhum servidor seja sobrecarregado, maximizando a velocidade e a capacidade de execução.

Se a instância de um determinado microsserviço cair, o balanceador de carga para de rotear requisições de clientes para esta instância. Da mesma forma, quando uma nova instância estiver disponível, o balanceador começa a repassar requisições para ela automaticamente. (OPUS SOFTWARE, 31/05/2021).

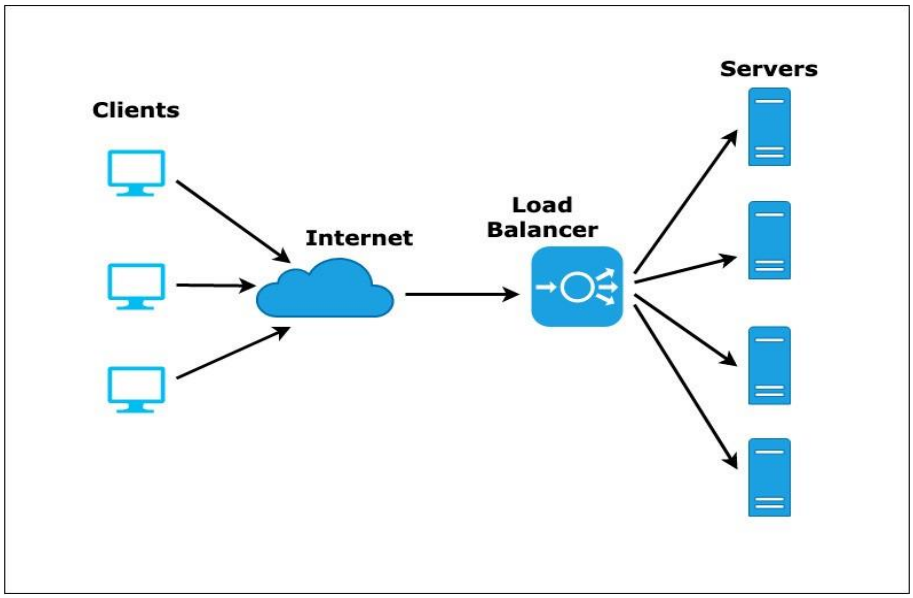

Figura 32 *- Load Balancer*

https://miro.medium.com/max/1400/1*tEaZGz-p1-E2ytNjl5RPJg.jpeg

APLICAÇÃO

Para demonstrar vários conceitos apresentados neste trabalho, um sistema foi desenvolvido usando o ambiente NodeJs e disponibilizado via API REST.

Visão geral do sistema

Este é um sistema fictício responsável pelo controle de fluxo de veículos em determinados locais cadastrados na aplicação. Os seguintes serviços foram implementados:

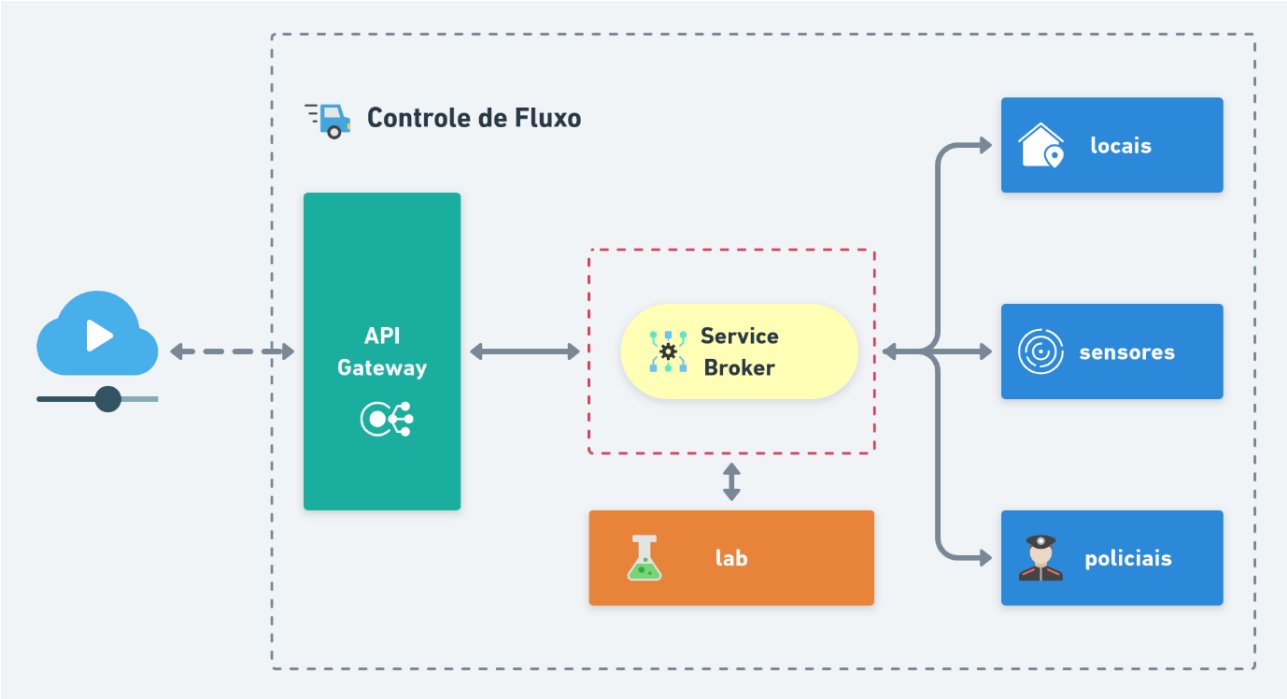

Figura 33 *- Controle de Fluxo*

Gerada pelo autor

- API: API *Gateway*, responsável por receber todas as requisições e direcionar para os serviços internos.
- Lab: Dashboard dos serviços, contendo métricas, logs e *tracings*.
- Locais: Responsável por cadastrar locais onde serão instalados os sensores de fluxo.
- Sensores: Responsável por cadastrar a passagem de um veículo, registrando a placa, o valor da tarifa e a velocidade. Se a velocidade for maior que 40 Km/h, um evento assíncrono "velocidade.alta" será emitido, informando os dados capturados pelo sensor.
- Policiais: Responsável por escutar se algum evento "velocidade.alta" for emitido e tomar as devidas providências.

Implementação local

Para reproduzir o cenário localmente, é necessário ter o ambiente NodeJs instalado. Em seguida o repositório git deve ser clonado, usando no terminal o seguinte comando:

git clone https://github.com/brasiqui/fluxo.git

Dentro da pasta criada, executar o comando npm install para instalar as dependências. Após a instalação, o comando npm run dev pode ser usado para iniciar a aplicação, e usando a url base http://localhost:3000/api, acessar os endpoints desenvolvidos.

Implementação em plataforma Cloud

Usando a tecnologia de containers, um cluster foi criado com *Docker Swarm*, que possibilita, através de arquivos de configuração, facilidade no *deploy*, escalabilidade e balanceamento de carga. O serviço

de sensores foi escalado com três instâncias e os demais com uma instância para cada um. Para a comunicação direta entre os serviços foi criada uma instância do *redis*, atuando como *service broker*. Portanto, a estrutura em *cloud* ficou dessa forma:

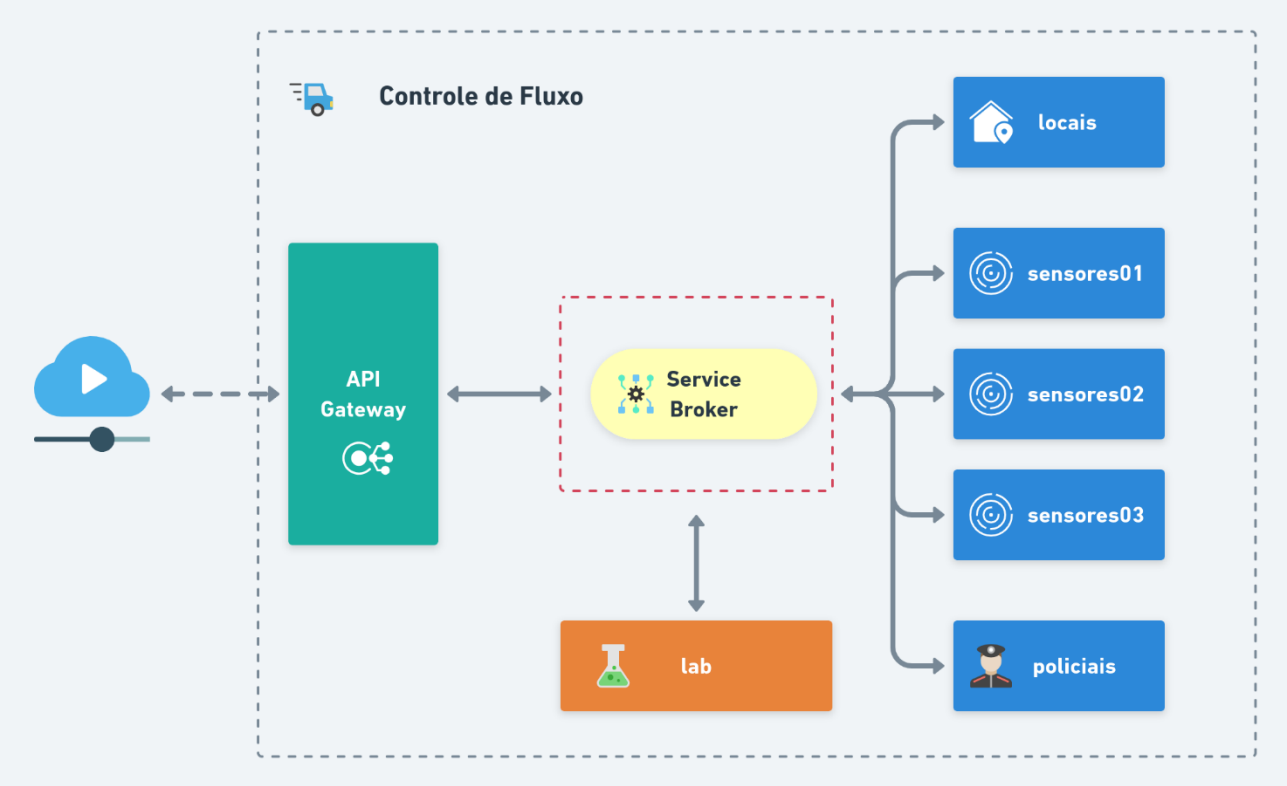

Figura 34 - Controle de Fluxo - Cloud

Gerada pelo autor

Demonstração

Cada serviço pode ter ações que são chamadas internamente ou expostas através de *endpoints*. Nos próximos tópicos, será exibido um exemplo das principais ações disponíveis no sistema.

Locais

Este serviço é responsável por cadastrar os locais e possui os seguintes *endpoints*:

- *Create* - POST /api/locais

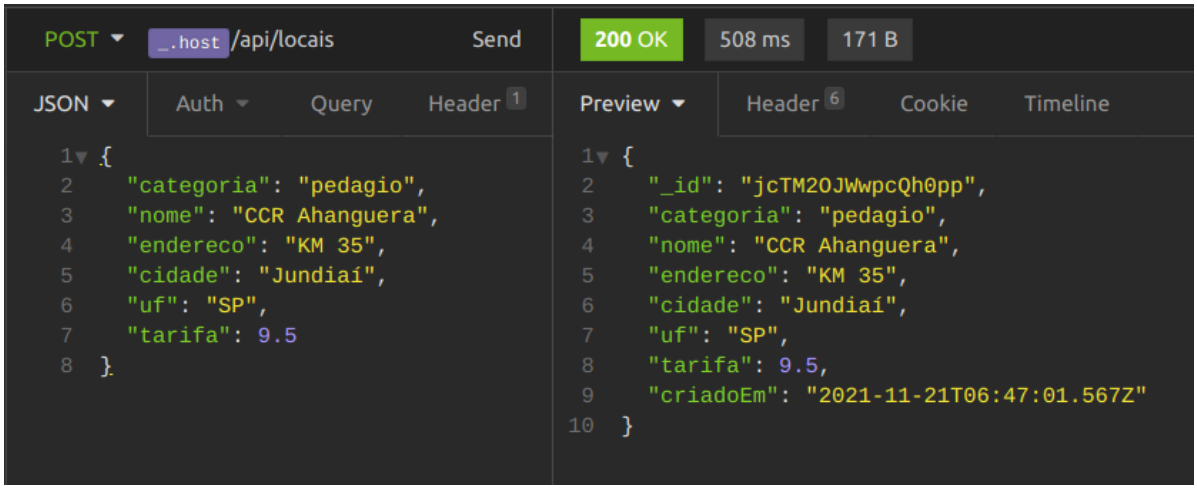

Figura 35 *- POST /api/locais*

Gerada pelo autor

- *Update* - PUT /api/locais/:id

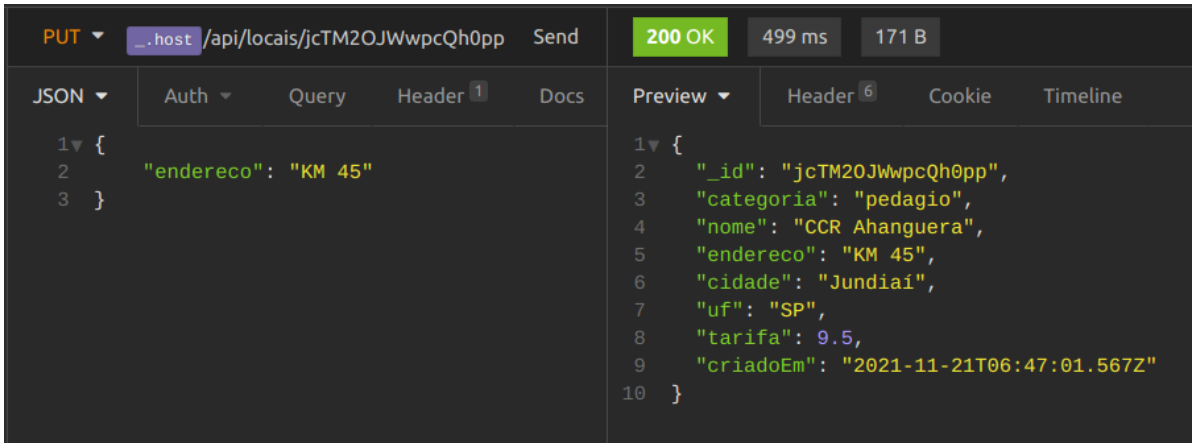

Figura 36 *- PUT /api/locais/:id*

Gerada pelo autor

- *Get* - GET /api/locais/:id

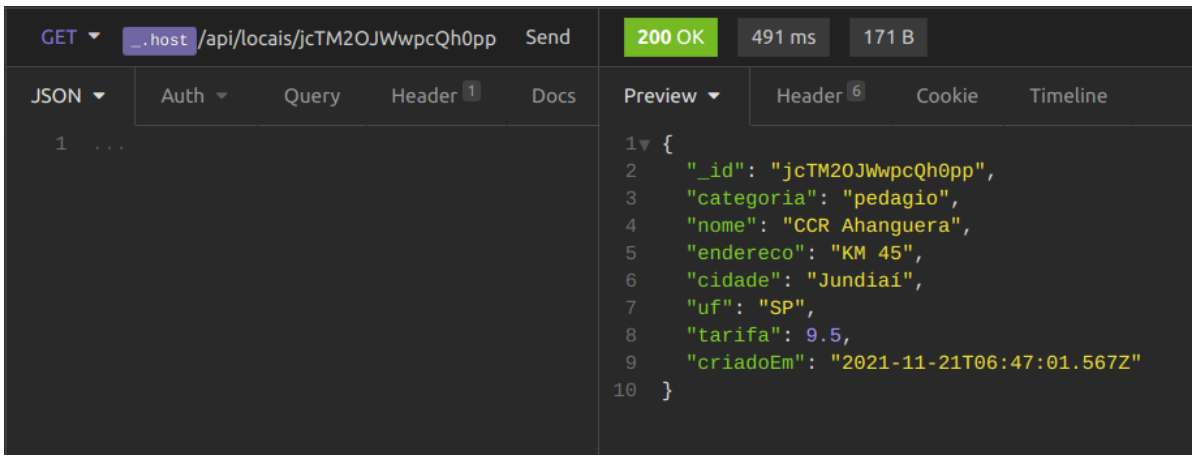

Figura 37 - GET /api/locais/:id

Gerada pelo autor

- *List* - GET /api/locais

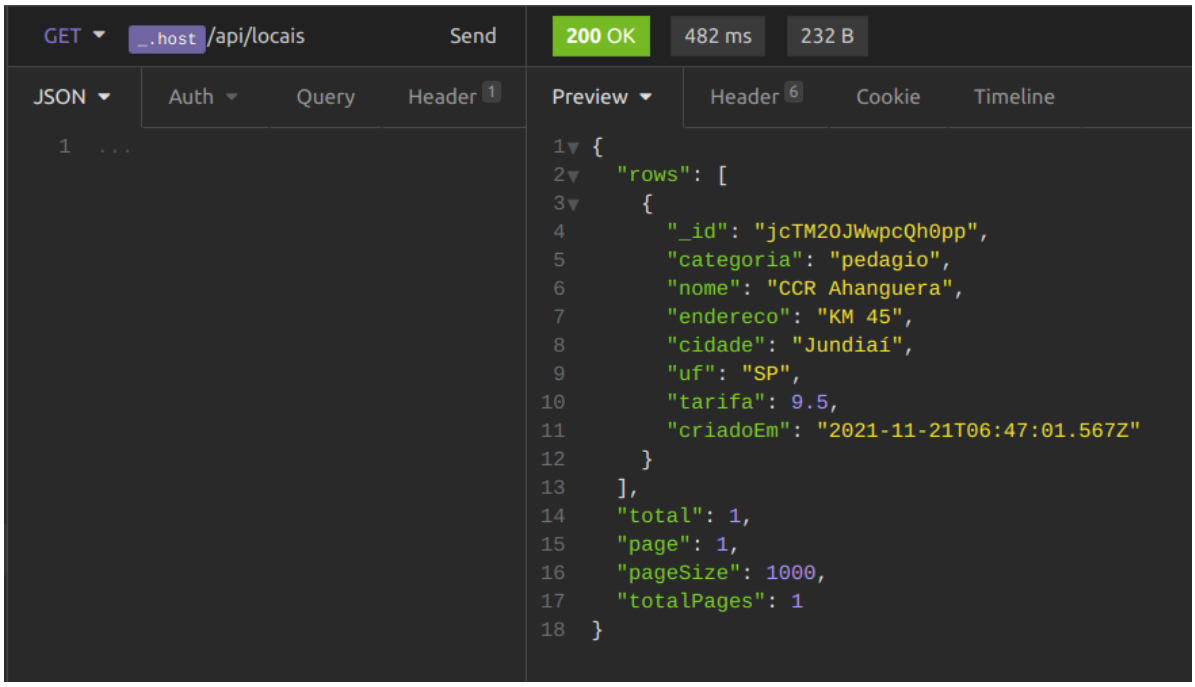

Figura 38 - GET /api/locais

Gerada pelo autor

- *Remove* - DELETE /api/locais/:id

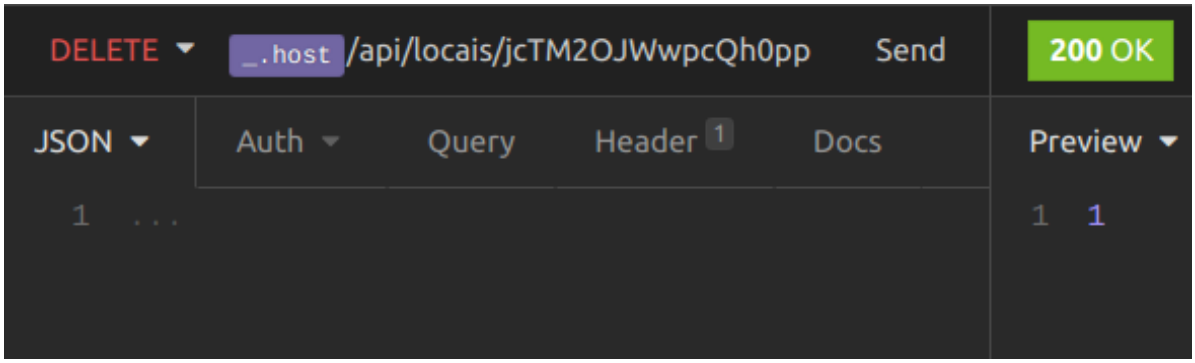

Figura 39 *- DELETE /api/locais/:id*

Gerada pelo autor

Sensores

Este serviço é responsável por cadastrar a passagem de um veículo e possui os seguintes endpoints, com formato similar ao anterior:

- *Create* - POST /api/sensores
- *Update* - PUT /api/sensores/:id
- *Get* - GET /api/sensores/:id
- *List* - GET /api/sensores
- *Remove* - DELETE /api/sensores/:id

Para testar o balanceamento de carga entre as três instâncias de sensores, uma ação fará o cadastro automático de uma passagem de veículo, utilizando dados aleatórios gerados pelo próprio sistema. Na resposta, virá o id do nó que respondeu a essa requisição. É possível notar que a cada execução, um nó diferente é acionado.

- *Seed* - POST /api/sensores/seed

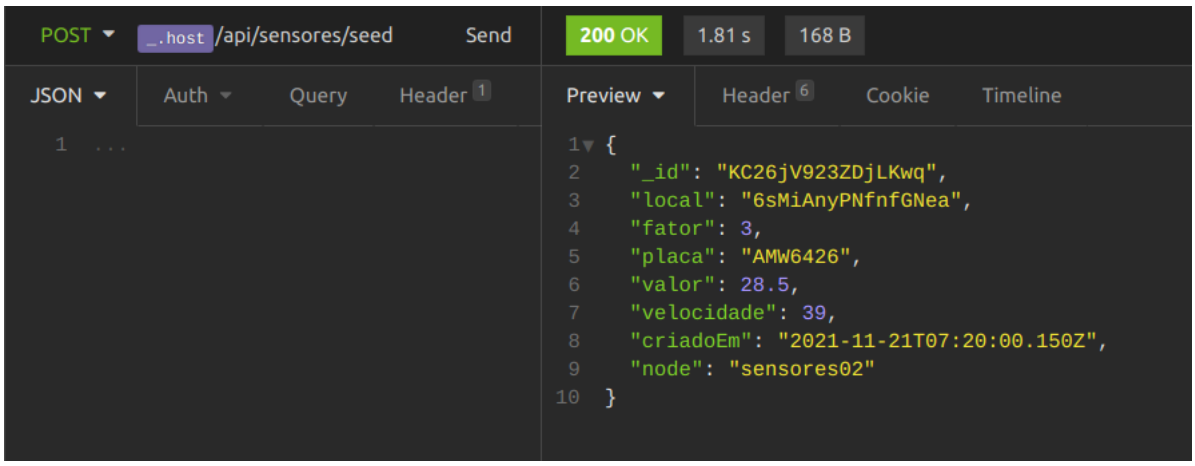

Figura 40 *– POST /api/sensores/seed*

Gerada pelo autor

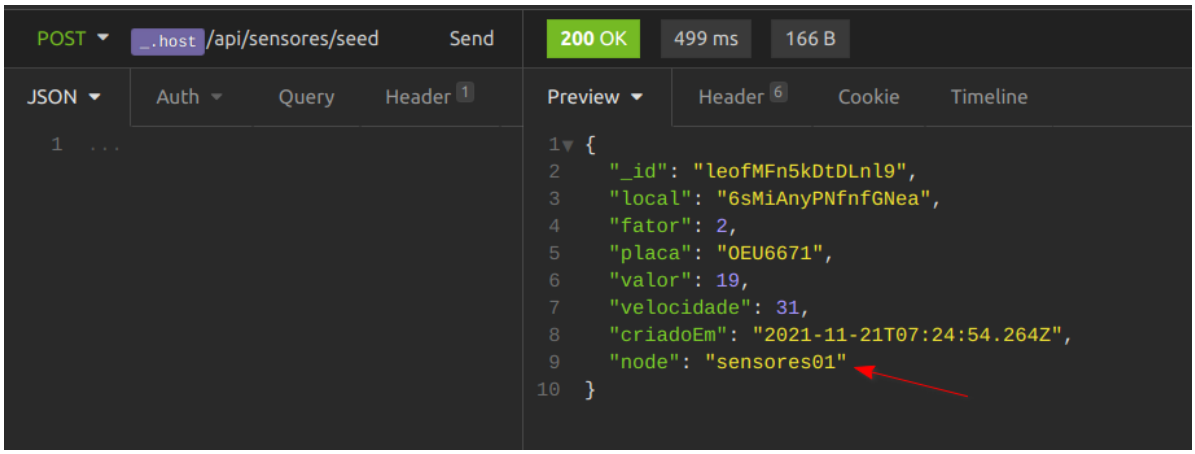

Figura 41 *- Balanceamento de sensores*

Gerada pelo autor

Policiais

Este serviço é responsável por escutar se algum evento "*velocidade.alta*" for emitido e tomar as devidas providências. No nosso caso, logo após cadastrar uma passagem de veículo acima de 40 Km/h, o serviço de sensores emite um evento que é recebido de forma assíncrona pelo serviço de policiais. Esta interação pode ser observada através dos logs do container.

```
[2021-11-21T07:26:25.389Z] WARN  policiais-19/POLICIAIS: Velocidade alta registrada.
[2021-11-21T07:26:25.389Z] WARN  policiais-19/POLICIAIS: Placa: FML9193
[2021-11-21T07:26:25.389Z] WARN  policiais-19/POLICIAIS: Velocidade: 49
[2021-11-21T07:26:25.395Z] INFO  policiais-19/TRACER: ┌──────────────────────────────────────
[2021-11-21T07:26:25.396Z] INFO  policiais-19/TRACER: │ ID: 714e9c82-dc04-475b-a964-f2cdf9f409eb
[2021-11-21T07:26:25.396Z] INFO  policiais-19/TRACER: ├──────────────────────────────────────
[2021-11-21T07:26:25.398Z] INFO  policiais-19/TRACER: │ Polícia Rodoviária
[2021-11-21T07:26:25.399Z] INFO  policiais-19/TRACER: └──────────────────────────────────────
```

Figura 42 *- Evento assíncrono*

Gerada pelo autor

Lab (Dashboard)

Este serviço é responsável por exportar informações do sistema de modo que possam ser lidas por outras aplicações. Assim, é possível visualizar métricas e informações do sistema em tempo real. Cada requisição executada grava logs e um *tracing* completo é registrado.

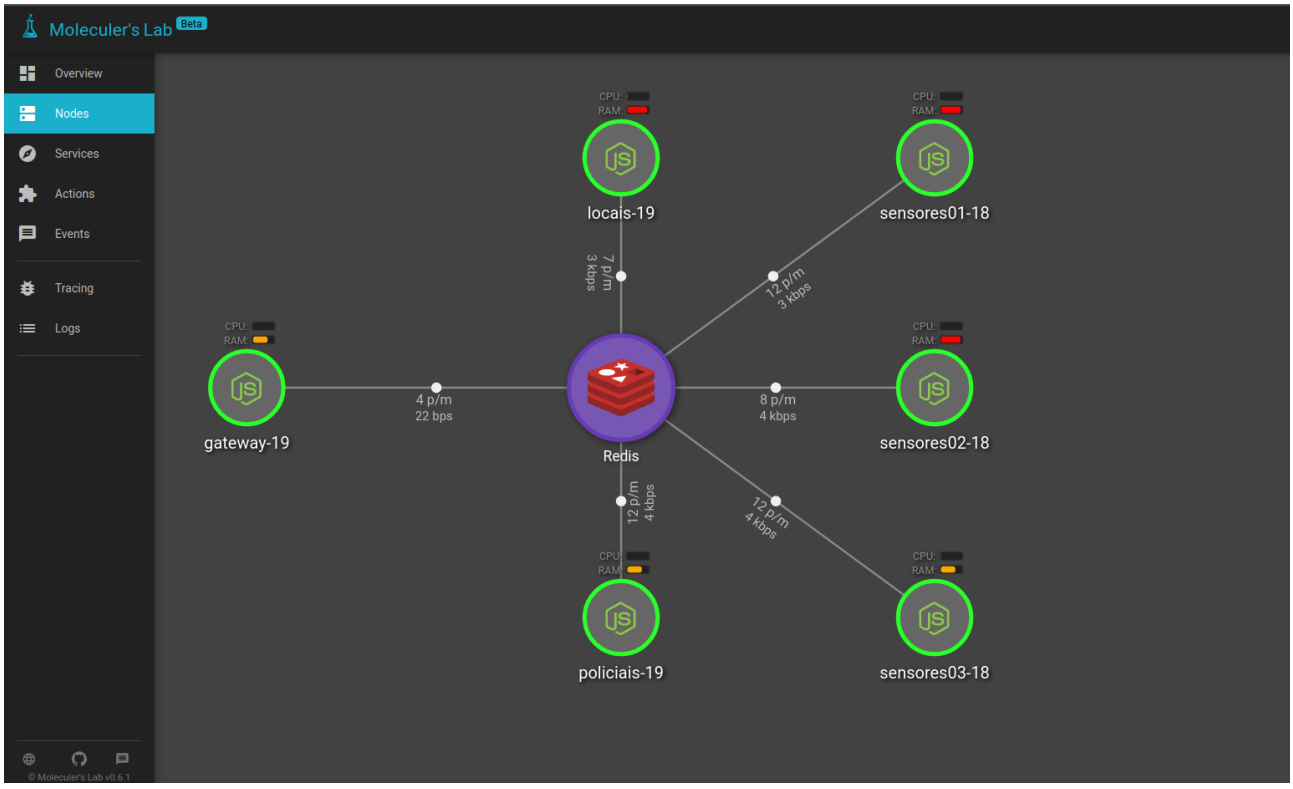

Figura 43 *- Dashboard*

Gerada pelo autor

CONCLUSÕES

Embora não exista uma definição formal para a arquitetura baseada em microsserviços, é possível identificar padrões de desenvolvimento essenciais, como divisão de responsabilidades, isolamento de independência, elementos de comunicação, tolerância a falhas e observabilidade.

Durante a implementação desses conceitos, foi possível notar que para implantar um sistema baseado em microsserviços, é necessário ter um conjunto de recursos que gerenciem essa plataforma de desenvolvimento, dando suporte à entrega e melhorias contínuas.

A adoção da arquitetura de microsserviços não é recomendada para todos os projetos, pois exige uma maturidade no desenvolvimento de aplicações e infraestruturas. Em muitos casos será interessante começar de forma mais simples e quando for necessário, aplicar de forma gradual os conceitos aqui apresentados. Em sistemas grandes e complexos, a arquitetura de microsserviços permite construir sistemas distribuídos altamente escaláveis, simplificando o desenvolvimento, homologação e implantações automatizadas, dando flexibilidade para adoção de novas tecnologias e entregando produtos mais confiáveis.

REFERÊNCIAS

AMAZON AWS. **Microsserviços**, 2021. Disponivel em: <https://aws.amazon.com/pt/microservices/>. Acesso em: 28 set. 2021.

PRASHANT, A.. **Retry pattern in microservices**, 26/01/2021. Disponivel em: <https://engineering.mercari.com/en/blog/entry/20210126-retry-pattern-in-microservices/>. Acesso em: 17 out. 2021.

BRYANT, D. **Service Mesh guia final: Gerenciando as comunicações serviço a serviço na era dos microservices**, 30/07/2020. Disponivel em: <https://www.infoq.com/br/articles/service-mesh-ultimate-guide/>. Acesso em: 17 out. 2021.

FOWLER, M. **CircuitBreaker**, 06/03/2014. Disponivel em: <https://martinfowler.com/bliki/CircuitBreaker.html>. Acesso em: 17 out. 2021.

FOWLER, M. **StranglerFigApplication**, 2004. Disponivel em: <https://martinfowler.com/bliki/StranglerFigApplication.html>. Acesso em: 17 out. 2021.

GONÇALVES, M. M. **Arquitetura de Microsserviços**, 01/06/2020. Disponivel em: <https://medium.com/@marcelomg21/arquitetura-de-microsservi%C3%A7os-bc38d03fbf64>. Acesso em: 17 out. 2021.

IBM REDBOOKS. **Microservices from Theory to Practice**. [S.l.]: IBM Redbooks, 2015. ISBN 0738440817.

LEWIS, J.; FOWLER, M. **Microservices - a definition of this new architectural term**, 25/03/2014. Disponivel em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 19 set. 2021.

MOLECULERJS. **Core Concepts**, 2021. Disponivel em: <https://moleculer.services/docs/0.14/concepts.html>. Acesso em: 17 out. 2021.

OPUS SOFTWARE. **Como criar o ecossistema de ideal para a arquitetura de microsserviços**, 31/05/2021. Disponivel em: <https://www.opus-software.com.br/arquitetura-de-microsservicos/>. Acesso em: 18 out. 2021.

PENNA, W. **Arquitetura Monolítica e Microsserviços**, 13/05/2020. Disponivel em: <https://www.zappts.com/blog/arquitetura-monolitica-e-microsservicos/>. Acesso em: 28 set. 2021.

PERRY, D. E.; WOLF, A. L. Foundations for the Study of Software Architecture. **SOFTWARE ENGINEERING NOTES**, 17, 1992. 40-53.

PEYROTT, S. **An Introduction to Microservices, Part 3: The Service Registry**, 02/10/2015. Disponivel em: <https://auth0.com/blog/an-introduction-to-microservices-part-3-the-service-registry/>. Acesso em: 18 out. 2021.

PRATES, B. G. **Strangler Pattern: como migrar um monólito para microsserviços**, 08/03/2021. Disponivel em: <https://imasters.com.br/apis-microsservicos/strangler-pattern-migrar-monolito-para-microsservicos>. Acesso em: 17 out. 2021.

PROMETHEUS. **Overview - Prometheus**. Disponivel em: <https://prometheus.io/docs/introduction/overview/>. Acesso em: 18 out. 2021.

RED HAT. **O que é arquitetura orientada a serviços (SOA)?**, 27/07/2020. Disponivel em: <https://www.redhat.com/pt-br/topics/cloud-native-apps/what-is-service-oriented-architecture>. Acesso em: 08 set. 2021.

RED HAT. **O que é service mesh?** Disponivel em: <https://www.redhat.com/pt-br/topics/microservices/what-is-a-service-mesh>. Acesso em: 17 out. 2021.

RED HAT. **O que são os microsserviços?** Disponivel em: <https://www.redhat.com/pt-br/topics/microservices/what-are-microservices>. Acesso em: 19 set. 2021.

RED HAT. **Qual é a função de um gateway de API?** Disponivel em: <https://www.redhat.com/pt-br/topics/api/what-does-an-api-gateway-do>. Acesso em: 18 out. 2021.

RIBENZAFT, R. **Microservices: Using Distributed Tracing for Monitoring & Troubleshooting**, 12/07/2019. Disponivel em: <https://cloudacademy.com/blog/microservices-using-distributed-tracing-monitoring-troubleshooting/>. Acesso em: 18 out. 2021.

RICHARDSON, C. **Pattern: API Gateway**, 2017. Disponivel em: <https://microservices.io/patterns/apigateway.html>. Acesso em: 18 out. 2021.

RICHARDSON, C. **Pattern: Strangler application**, 2017. Disponivel em: <https://microservices.io/patterns/refactoring/strangler-application.html>. Acesso em: 17 out. 2021.

RICHARDSON, C. **Microservices Patterns**. 1. ed. Shelter Island, NY: Manning Publications, 2019. ISBN 9781617294549.

RICHARDSON, C. **Pattern: Monolithic Architecture**, 2019. Disponivel em: <https://microservices.io/patterns/monolithic.html>. Acesso em: 19 set. 2021.

SEELEY, L. **4 Resiliency Patterns Leveraged by Capital One's Mobile Edge Engineering Team**, 19/09/2018. Disponivel em: <https://medium.com/capital-one-tech/resiliency-patterns-at-the-edge-capital-one-a5b4d41d477e>. Acesso em: 17 out. 2021.

SELVARAJ, V. **Resilient Microservice Design – Bulkhead Pattern**, 04/11/2020. Disponivel em: <https://dzone.com/articles/resilient-microservices-pattern-bulkhead-pattern>. Acesso em: 17 out. 2021.

SELVARAJ, V. **Timeout Pattern — Resilient Microservice Design With Spring Boot**, 25/10/2020. Disponivel em: <https://vinsguru.medium.com/resilient-microservice-design-with-spring-boot-timeout-pattern-72b5f5174d2a>. Acesso em: 17 out. 2021.

SIAHAAN, J. N. **API Gateway Part 2**, 07/09/2019. Disponivel em: <https://medium.com/easyread/api-gateway-part-2-7264ee5be187>. Acesso em: 18 out. 2021.

SILVA, A. F. **Alcançando a excelência do REST com um Modelo de Maturidade eficiente**, 02/10/2017. Disponivel em: <https://mundoapi.com.br/destaques/alcancando-a-excelencia-do-rest-com-um-modelo-de-maturidade-eficiente/>. Acesso em: 17 out. 2021.

UM ESTUDO APLICADO A SEGURANÇA DE APLICAÇÕES WEB

A STUDY APPLIED TO WEB APPLICATION SECURITY

Felipe Delboni ORTEGA
fdelb.ortega@gmail.com
Aluno do Curso de Bacharelado em Ciência da Computação
Centro Universitário Padre Anchieta - Jundiaí, SP

Carlos Eduardo CÂMARA
dinhocamara@gmail.com
Pesquisador na Área de Ciência da Computação
Doutorado e Mestrado em Engenharia Elétrica - FEEC/UNICAMP – Campinas/SP

RESUMO

O objetivo desse trabalho, consiste em apresentar um estudo teórico orientado à segurança de aplicações Web, a partir da análise de alguns de seus principais pontos de vulnerabilidade, permitindo seu entendimento, a mitigação e uma visão compreensiva da importância de uma estratégia de desenvolvimento seguro, baseada em estudos analíticos e metodologias de mercado. De maneira específica, o projeto "*OWASP Top Ten 2021"* é a metodologia fundamental adotada neste trabalho, contemplando os pontos focais e cenários de fragilidade em comum entre a maioria das aplicações Web, em conjunto com demais experiências no mundo da tecnologia.

Palavras-Chave: Desenvolvimento seguro, Segurança da informação, Teste de intrusão, *Owasp Top 10 2021*.

ABSTRACT

This work consists in presenting theoretical research oriented to web application security through the analysis of some of its main points of vulnerability, allowing its understanding, mitigation and a comprehensive vision of the importance of a secure development strategy, based on analytical studies and market methodologies. In a specific way, the "OWASP Top Ten 2021" project is the fundamental methodology of this work, contemplating the focal points and common fragility scenarios among most Web Applications, together with other experiences in the technology world.

Keywords: Secure Development, Information Security, *pentest*, *Owasp Top 10 2021*.

1. INTRODUÇÃO

A tecnologia está em constante evolução e seu avanço traz consigo uma enorme dependência das capacidades computacionais, acesso à Internet e acesso à informação. E com a tendencial utilização dos dispositivos e das transações digitais, os riscos de perda, vazamento e roubo de informações, ou até mesmo a possibilidade de indisponibilidade de serviços aumentam significativamente.

As vulnerabilidades intrínsecas a aplicações e sistemas da informação, que, por vezes não são controladas, devido a incapacidade de compreensão da extensão das estruturas de comunicação complexas de hoje, podem causar enormes prejuízos financeiros as organizações. Ademais, a condição de risco passa a ser ainda maior quando as aplicações estão direcionadas para o contexto web, já que este tipo de aplicação vem numa crescente, respondendo por 67% de todos os ataques em 2021, segundo o *"Relatório de Inteligência de Ameaças Globais 2021 (GTIR)",* lançado pela empresa *"Nippon Telegraph and Telephone Corporation (NTT)*".

A fim de reduzir possíveis vulnerabilidades de segurança voltadas a aplicações Web, estratégias de desenvolvimento seguro são extremamente necessárias, já que o cenário web há algum tempo é considerado popular nas organizações, baseado em sua facilidade de acesso a serviços e gerenciamento de sistemas centralizado.

2. CONCEITOS BÁSICOS

Esta seção busca apresentar as variáveis nomenclaturas e termos que serão abordados ao longo do trabalho, e seus respectivos significados, para melhor compreensão sobre os assuntos tratados.

2.1 CWE

Segundo a MITRE, CWE (Common Weakness Enumeration – Enumerações de fraquezas comuns), é uma lista desenvolvida pela comunidade, com o intuito de informar os tipos de fraquezas de software e hardware. A empresa (MITRE), afirma que esta lista serve como uma linguagem comum e fonte de medição, para a identificação de fraquezas, mitigação e esforços de prevenção.

A CWE é patrocinada pelo Departamento de Segurança Interna dos Estados Unidos (DHS – Department of Homeland Security), Agência de Segurança Cibernética e Infraestrutura (CISA - Cybersecurity and Infrastructure Security Agency) e gerenciada pelo Instituto de Engenharia e Desenvolvimento de Sistemas de Segurança Interna (HSSEDI - Homeland Security Systems Engineering and Development Institute), operado pela MITRE Corporation (MITRE).

2.2 CVE

Segundo a Red Hat (2020), CVE (do inglês Common Vulnerabilities and Exposures), é uma lista de falhas de segurança de computadores, divulgadas publicamente. Dessa forma, as referências CVE que serão citadas neste trabalho, significam uma falha de segurança específica que recebeu um número de identificação, ID CVE.

Ainda em conformidade com a Red Hat (2020), o programa CVE assim como o CWE supervisionado é mantido pela corporação MITRE (Corporation).

2.3 MÉTODOS DE SOLICITAÇÃO HTTP

Segundo MOZILLA (2021), HTTP define um conjunto de métodos de solicitação para indicar a ação desejada a ser executada para um determinado recurso que se encontra no servidor. Esses métodos são: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE e PATCH.

Embora também possam ser substantivos, esses métodos de solicitação às vezes são chamados de *verbos HTTP*.

2.4 JWT

Segundo a AUTH0 (2021), JSON web token (JWT), é um padrão aberto (RFC 7519) que define uma maneira compacta e independente para transmitir informações com segurança entre partes, como um objeto JSON. Um JWT pode ser enviado por meio de uma URL, por meio de um parâmetro POST ou dentro de um cabeçalho HTTP e é transmitido rapidamente.

Pode ser utilizado como forma de autenticação, quando um usuário efetua login com sucesso usando suas credenciais, Autorização, uma vez que um usuário é autenticado com sucesso ou também como troca de informações, em uma transmissão segura de dados entre as partes – por exemplo, aplicação cliente e servidor.

2.5 CORS

Segundo MOZILLA (2021), Cross-Origin Resource Sharing (CORS) é um mecanismo baseado em cabeçalho HTTP que permite a um servidor indicar qualquer origem (domínio, esquema ou porta) diferente da sua própria, a partir da qual um navegador deve permitir o carregamento de recursos.

Na prática, este conceito funciona de maneira que quando uma aplicação Front-end, JavaScript por exemplo, registrado com domínio https://dominio-frontend.com, tenta fazer uma solicitação à https://dominio-

backend.com. Por motivos de segurança, os navegadores restringiriam esta solicitação devido a uma requisição de origem cruzada. Dessa forma, a aplicação Front-end só conseguirá solicitar recursos à aplicação Back-end se os cabeçalhos CORS corretos forem configurados.

2.6 OAUTH

SOBERS (2018), diz que OAuth é um protocolo ou estrutura de autorização de padrão aberto que fornece aos aplicativos a capacidade de "acesso designado seguro", ou seja, é um protocolo de autenticação que permite com que um usuário seja aprovado em uma aplicação que está interagindo com outra, sem revelar sua senha, a partir de tokens de autorização.

2.7 NEGAÇÃO DE SERVIÇO (DoS) NEGAÇÃO DE SERVIÇO DISTRIBUÍDO (DDoS)

COSTA (2014), define DoS (do inglês, Denial of Service), como uma forma de ataque a sistemas computacionais. Consiste em uma tentativa de sobrecarga em um servidor ou computador, para que os recursos do sistema - pode-se citar como exemplo CPU, memória, banda, etc. - fiquem indisponíveis.

Já DDoS (do inglês, Distributed Denial of Service), segundo COSTA (2014), é um ataque similar ao de DoS, porém, ele atinge camadas extras, ou seja, atinge um ou mais computadores mestres, que gerenciam uma série de outros computadores, também chamados de "zumbis", causando um ataque distribuído. Dessa forma, com o acesso de vários computadores "mestres", e computadores "zumbis", o invasor os utiliza para aumentar a sobrecarga sobre determinado servidor.

2.8 PATCH

Segundo a FC BRASIL (2020), um patch - que traduzido ao português significa correção ou remendo - pode ser definido como uma solução rápida para atualizar ou corrigir um software. Patches são criados, quando há uma vulnerabilidade existente em um software, computador, dispositivos móveis ou outras máquinas em uma rede. Eles garantem que os hackers não usem a fragilidade para invadir redes corporativas.

Este termo será abordado neste trabalho em algumas menções de ataques cibernéticos ou em algumas formas de mitigação de riscos de segurança.

2.9 PHISING

PHISHING (2017), define o termo Phishing em si, como um crime cibernético em que um alvo é contatado por e-mail, telefone ou mensagem de texto por alguém que se passa por uma instituição legítima ou amigável, com o intuito de atrair indivíduos a fornecer dados confidenciais, como informações de identificação pessoal, dados bancários e senhas, resultando em roubo de identidade e perda financeira.

2.10 AUTENTICAÇÃO MULTIFATORES

HOSTMIDIA (2021), define Autenticação multifatores - ou também conhecida como autenticação de dois fatores – como o uso de dois ou mais fatores ou agentes de verificação de autenticidade. Isto é, a utilização de dois ou mais métodos para atestar a identidade de alguém para concessão de acesso a um sistema, documento ou informação.

Algumas formas de autenticação de multifatores podem ser o envio de um código numérico de verificação via Email ou SMS para uma conta/número já cadastrado previamente no sistema em questão, através de verificação biométrica como leituras de digitais ou reconhecimento facial, aplicativos de geração de códigos (como Google Authenticator), etc.

2.11 PIPELINES DE CI / CD

Por definição, o termo CI, define-se como C*ontinuous Integration*, e CD C*ontinuous Delivery*. Dessa forma, uma pipeline de CI / CD, consiste numa série de etapas que devem ser executadas para fornecer uma nova versão de software. (REDHAT, 2019)

Focada em melhorar a entrega de software, a partir de uma abordagem de DevOps, uma Pipeline CI / CD busca apresentar monitoramento e automação para melhorar o processo de desenvolvimento de aplicativos, especialmente nas fases de integração, teste, entrega e implantação. (REDHAT, 2019)

2.12 SUPPLY CHAIN ATTACK

GCSEC (2021) define um ataque de cadeia de suprimentos, como uma estratégia que visa afrontar empresas de software ou provedoras de serviços terceirizados, por meio de vulnerabilidades em sua cadeia de fornecimento, causando de certa forma, um efeito dominó.

MICROSOFT (2021) define Supply Chain Attack - ou Ataque a cadeia de suprimentos – como um tipo emergente de ameaça que tem como alvo os desenvolvedores e fornecedores de software, cujo objetivo é acessar códigos-fonte, processos de construção ou mecanismos de atualização, infectando aplicativos legítimos para distribuir malware em sua cadeia de fornecimento, ou seja, espalhá-lo para os clientes da empresa fornecedora do software atingido.

2.13 ATAQUE MITM

Malenkovich (2013) diz que um ataque MITM, ou Ataque Man-in-the-Middle, consiste em um ataque cujo invasor se posiciona entre duas partes que tentam comunicar-se, intercepta mensagens enviadas e depois se passa por uma das partes envolvidas. Dessa forma, este agente malicioso pode colocar suas armadilhas entre a vítima e sites relevantes, como sites de bancos e contas de e-mail.

A variante do ataque MITM mais utilizada é a partir de uma situação cujo agressor configura seu dispositivo wireless para atuar como ponto de WiFi, nomeando-o com um título comum em redes públicas, na qual se bem sucedido, o invasor poderá roubar todas as credenciais transacionadas. Estes ataques são

extremamente eficientes e difíceis de detectar, especialmente por usuários inexperientes ou desavisados. (Malenkovich**,** 2013)

2.14 SERIALIZAÇÃO E DESSERIALIZAÇÃO

Segundo DEVOPEDIA (2020), “desserialização” define-se no processo de transformar algum objeto serializado - por exemplo no formato JSON, ou XML - em um formato estruturado de dados - por exemplo baseado em uma classe no conceito de orientação a objetos. Podendo assim ser armazenado em banco de dados, em arquivos ou ser utilizado como parte das comunicações entre as funções do sistema.

DEVOPEDIA (2020) menciona serialização no processo inverso, isto é, coletar este objeto serializado e estruturado para aquela aplicação específica - por exemplo um objeto tipado em uma aplicação JAVA ou C# - e reconstruí-lo em um objeto compreensível – JSON, ou XML - para que outra aplicação possa realizar sua leitura.

2.15 LOG

Segundo MACHADO (2012), o registro de logs define-se no registro de dados e informações cruciais em arquivos de texto, para verificação posterior, que tem como objetivo o monitoramento e a detecção de ações impróprias nos sistemas de informação.

2.16 RANSOMWARE

KASPERSKY (2021) define Ransomware, a partir da divisão da palavra, "ransom", que significa resgate. Dessa forma, Ransomware é um software de extorsão que tem o intuito de bloquear um computador, como quaisquer arquivos presentes nele, e depois exigir um resgate para desbloqueá-lo, onde normalmente este resgate é solicitado em dinheiro. Este bloqueio pode ser realizado a partir da criptografia do sistema operacional ou apenas para arquivos individuais.

2.17 APT

APT (do inglês, Advanced Persistent Threats), como o nome "avançado" sugere, um ataque persistente avançado, utiliza técnicas de invasão contínuas, clandestinas e sofisticadas para obter acesso a um sistema e permanecer dentro dele por um período prolongado, com consequências potencialmente destrutivas.

2.18 DEVOPS E DEVSECOPS

MXM SISTEMAS (2020) define o conceito de DEVOPS como:

> “*O DevSecOps — desenvolvimento, segurança e operação — pode ser conhecido como a evolução do DevOps — desenvolvimento e operação. Embora o DevOps também envolva*

> *segurança, o foco do conceito está em agregar as práticas mais eficazes de desenvolvimento de software.*"

De acordo com a definição de DEVOPS apresentada, é possível concluir que embora equipes de DevOps estejam por vezes encarregadas de atividades de segurança, seu verdadeiro foco é no auxílio e suporte das equipes de desenvolvimento de software.

Com base nisso, MXM SISTEMAS (2020) define DEVSECOPS conforme abaixo:

> *"O DevOps faz parte de uma cultura de engenharia de software que tem como objetivo principal unificar a operação e o desenvolvimento e, por isso, tem sido cada dia mais adotada. No entanto, a segurança ainda é uma medida secundária nesse modelo. Logo, o DevSecOps é a opção que visa resolver essa questão."*

2.19 TESTE DE INTRUSÃO

Teste de intrusão ou muito conhecido pelo termo *pentest*, é o método de simulação de ataques cibernéticos em uma aplicação, com o objetivo de diagnosticar suas fragilidades de segurança, possibilitando a solidificação de melhores estratégias de defesa (BLOG DA CCM, 2019). Para LIMA (2020, p. 20), sua execução, em contextos específicos, pode ser benéfica na análise dos requisitos não-funcionais de um sistema, como desempenho, usabilidade e segurança.

SALGADO (2014) enaltece que este método de diagnóstico fortalece o ambiente tecnológico, pois é executado a partir do ponto de vista do atacante e não se compara às auditorias de segurança baseadas na norma ISO/IEC 27001, uma vez que estas são passivas, poucos intrusivas e aplicadas por meio de entrevistas aos funcionários. (apud CINTO, 2015, p. 21).

O posicionamento de SALGADO traz uma visão crítica, sobre às auditorias de segurança baseadas nas normas e diretrizes ISO/IEC 27001:2006, em comparação aos testes de penetração. Em vista disso, partindo da perspectiva da relação dos *pentests* com o padrão de segurança ISO/IEC 27001:2013, de acordo com SEGOVIA (2016), apenas a realização de análise de vulnerabilidade já mantém a conformidade com a própria ISO 27001:2013, porém não garantirá discernir o quão vulnerável a aplicação está. Este resultado só poderá ser obtido através da exploração detalhada pelo teste aplicado.

Para a sua execução, SEGOVIA (2016) recomenda direcionar os testes de intrusão em algumas fases (Planejamento, Coleta de Informações, Modelagem de Ameaças, Análise de vulnerabilidade, Exploração, Pós-exploração e Relatórios), na qual pode-se relacioná-las com o que diz a metodologia (do inglês Penetration Testing Execution Standard) PTES.

O padrão de execução do teste de penetração (PTES, do inglês Penetration Testing Execution Standard) (PTES, 2017), é uma metodologia de testes de intrusão. Ela é baseada em uma divisão de 7 seções principais, com objetivo de realizar a cobertura total necessária de um pentest.

O fluxo da Figura 1 abaixo simboliza as 7 etapas da metodologia PTES, com suas devidas descrições em seguida (LIMA, 2020, p. 22-23):

Figura 1. Fluxo das etapas da metodologia PTES

Fonte: LIMA, 2020, p. 22.

1. **Pré-engajamento** (pre-engagement interactions): etapa para definir o escopo e objetivo do teste de intrusão;

2. **Levantamento de informações** (intelligence gathering): etapa para coletar informações da AST, como usuário e senha, por exemplo;

3. **Modelagem de ameaças** (threat modeling): etapa para desenvolver estratégias para minimizar riscos associados à presença de vulnerabilidades na AST;

4. **Análise de vulnerabilidades** (vulnerability analysis): etapa para identificar vulnerabilidades que podem ser exploradas pelo testador na AST;

5. **Exploração** (exploitation): etapa para o testador estabelecer acesso à AST a partir de vulnerabilidades existentes;

6. **Pós-exploração** (post exploitation): etapa para manter acesso à AST e esconder possíveis evidências da invasão;

7. **Relatar resultados** (reporting): etapa para produzir um relatório com os resultados obtidos nos testes.

A sigla AST conforme LIMA (2020) é definida como Aplicação Sob Teste.

Segundo Sanches (2018), em um *pentest* pode-se utilizar diversas metodologias de mercado, que já foram aplicadas e testadas por outras empresas, consolidadas em uma espécie de script, de forma que o pentester não fique preso somente em uma metodologia, mas que a utilize como referência para investigação dos riscos.

Ainda de acordo com Sanches, atualmente, as principais metodologias de mercado para a realização de testes de intrusão em aplicações Web são: PTES (Penetration Testing Execution Standard), OSSTMM (Open Source Security Testing Methodology Manual), NIST 800-115 (National Institute of Standards and Technology), e OWASP Top 10 (Open Web Application Security Project), que será a metodologia referência a ser utilizada neste trabalho. (Sanches, 2018)

3. OWASP

Partindo ao foco deste trabalho, a apresentação da metodologia base será enfatizada permitindo que o leitor tenha maior compreensão e interpretação metodológica, acerca das definições técnicas a serem expostas nas próximas seções.

3.1 SOBRE A FUNDAÇÃO

Fundada em 1º de dezembro de 2001, a OWASP (Open Web Application Security Project - Projeto Aberto de Segurança em Aplicações Web) é uma fundação sem fins lucrativos que trabalha em prol da segurança da informação. Uma metodologia de mercado, formada por desenvolvedores, pesquisadores e especialistas em segurança da informação, que disponibilizam conteúdos educacionais através de conferências, treinamentos, artigos técnicos, e projetos de software de código aberto, liderados pela comunidade. (OWASP, 2021)

A entidade produz uma variedade de materiais de forma colaborativa, transparente e aberta, para empresas e profissionais da área manterem e implementarem aplicações confiáveis, além de fornecer um dos principais projetos de pesquisa utilizados para testes de intrusão e desenvolvimento seguro, o já citado OWASP TOP 10. (OWASP, 2017)

3.2 O PROJETO TOP 10

Segundo a própria OWASP (OWASP, 2021), o projeto Top 10 é um documento padrão de conscientização de segurança de aplicativos da web para desenvolvedores. Ele representa um amplo consenso sobre os riscos de segurança mais críticos para aplicativos da web.

Em seu último lançamento de 2021 (OWASP, 2021), a fundação busca estimular as empresas para que adotem este documento, a fim de garantir a minimização dos riscos abordados, e complementa que a utilização do OWASP Top 10 seja talvez o primeiro passo mais eficaz para mudar a cultura de desenvolvimento de software em uma organização, para que códigos mais seguros sejam produzidos.

Esta metodologia, na maioria dos contextos, é referência útil para orientar os desenvolvedores, a partir de problemas comuns que tornam um código inseguro. Conforme as aplicações são moldadas, baseadas nas detecções e abordagens dos riscos, é certo que haverá o aumento da resistência às ameaças cibernéticas.

A construção do projeto é realizada de maneira híbrida. De acordo com o OWASP, esta abordagem mista é adotada porque os dados estatísticos refletem apenas fatos conhecidos do passado, mas podem não cobrir tendências recentes, uma área que complementa os resultados da pesquisa.

A definição dos 10 riscos da lista, é compilada a partir de dois métodos, conforme será citado abaixo:

1. Coleta de dados

O processo de coleta de dados é o que define oito dos dez riscos da lista, que consiste no período de coleta de dados estatísticos sobre vulnerabilidades de aplicações web, encontradas em vários processos. Os dados são fornecidos por organizações parceiras, que realizam serviços de testes em aplicações de outras instituições ou contribuem com dados de teste internos e, doravante, os enviam para a OWASP a partir de uma convocatória de dados.

Segundo a OWASP (OWASP, 2021), a fim de tornar este o maior e mais abrangente conjunto de dados de segurança de aplicativos, foram fornecidos dados de mais de 500 mil aplicativos. Ainda em conformidade com a OWASP (OWASP, 2021), as empresas doadoras dos dados para a edição atual são:

AppSec Labs; Cobalt.io; GitLab; *HackerOne; HCL Technologies; Micro Focus; PenTest-Tools; Probely; Sqreen; Veracode; WhiteHat (NTT);*

2. Pesquisa da comunidade Top 10

A pesquisa da comunidade têm intuito de questionar à especialistas em segurança e desenvolvimento na linha de frente, suas visões e previsões a respeito de fraquezas essenciais que podem não ser refletidas apenas em dados estatísticos. Seus resultados implicam na determinação de duas das dez categorias do projeto.

Esta metodologia de pesquisa foi implementada pela fundação na última edição lançada em 2017, visto que segundo a OWASP (OWASP, 2021), esta é uma forma valiosa de permitir que os indivíduos da comunidade identifiquem riscos importantes que podem ter passado despercebidos na coleta dos dados pelas organizações.

A pesquisa segue o seguinte fluxo no qual inicialmente são mapeados CWEs iminentes de entrar na lista do Top 10, ou apresentados em eventos significativos. Estes CWEs são retornados para a comunidade para avaliação, para que em seguida seja realizada a tabulação dos resultados. Esta tabulação consiste em comparar os resultados de pesquisa com a análise dos dados, onde a partir disso, são selecionados os dois riscos que receberem a maioria dos votos dos especialistas, e ainda não estão presentes nos dados analisados.

4. ANÁLISE DE RISCOS

Desde 2017, quando a última lista com as dez maiores vulnerabilidades foi lançada, analisando a categorização de riscos realizada pela OWASP há 4 anos atrás, comparada com o recente lançamento de 2021, é possível mensurar algumas das novas tendências de fragilidades de segurança, no ciclo de desenvolvimento de aplicações Web.

A figura 2, apresenta um comparativo das vulnerabilidades da última edição com o lançamento de 2021.

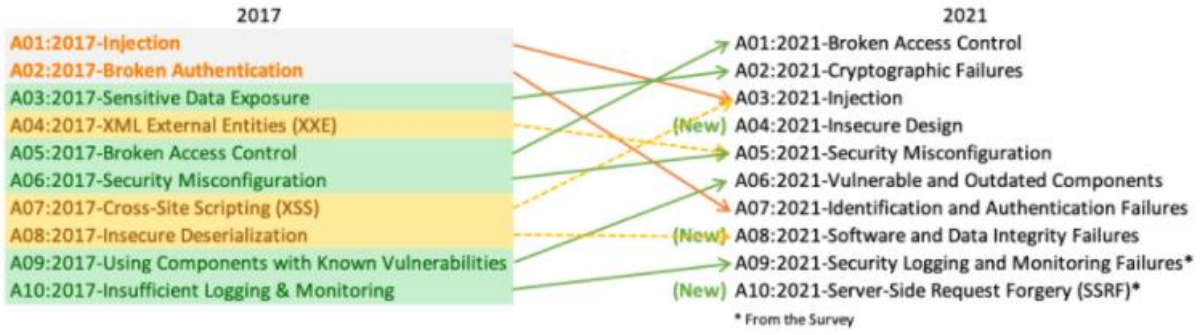

Figura 2. Comparativo OWASP TOP 10 do ano 2017 a 2021

Fonte: OWASP, 2021.

Em uma análise comparativa a partir da Figura 2, é possível perceber que todas as categorias foram mantidas de 2017 para 2021, porém com transformações em suas classificações. De maneira geral, em alguns

casos houve a consolidação de mais de uma categoria da última edição, sendo agrupada em apenas uma. De acordo com a OWASP (2021) estas alterações ocorreram devido a necessidade de concentrar na causa raiz em vez do sintoma.

Houve também a inclusão de três novas categorias, que são: A04:2021-Insecure Design, focada nos riscos relacionados a falhas de design e arquitetura de aplicações; *A08:202-Software and Data Integrity Failures* com o intuito de realizar suposições relacionadas a atualizações de software e A10:2021-Server-Side Request Forgery (SSRF), que foi a primeira das duas categorias adicionadas pelo processo de pesquisa da comunidade da OWASP (OWASP, 2021).

Segundo uma análise realizada pela HKCERT (2021), foram resumidos alguns pontos a respeito das alterações:

- A Exposição de Dados Sensíveis (Sensitive Data Exposure) passou a ser reclassificada como Falhas Criptográficas, devido a este tipo de falha ser, na verdade, a causa raiz do sintoma de exposição de dados confidenciais.
- Cross-Site Scripting (XSS) foi incluso na categoria de Injeção (Injection), o que tecnicamente, é uma injeção de script direcionada a outros usuários que acessaram o site afetado. Este é um ataque diferente dos métodos de injeções que acontecem do lado do servidor, como SQL, comandos do sistema operacional e injeções LDAP.
- A vulnerabilidade XML External Entities (XXE) passou a ser recategorizada à "Configuração Incorreta de Segurança" (Security Misconfiguration), já que muitas vezes é um erro cometido pelo desenvolvedor ao configurar incorretamente os analisadores XML em seu código.
- A desserialização insegura (Insecure Deserialization) é colocada em "Falhas de Integridade de software e de dados". Este, por ser um problema de proteção insuficiente no tratamento de objetos serializados, leva à adulteração de dados e à transmissão desses à aplicação, a partir de uma fonte não confiável. Um número significativo de desserializações inseguras é atribuído à vulnerabilidade de código aberto, que deve ser gerenciada ativamente e não deve ser negligenciada.

4.1 BROKEN ACCESS CONTROL

O controle de acesso impõe a política de modo que os usuários não possam agir fora de suas permissões pretendidas. Dessa forma, a quebra do controle de acesso, normalmente leva à divulgação, modificação ou destruição de dados, informações ou ao desempenho de uma funcionalidade comercial do sistema, que se encontra fora dos limites do usuário. (OWASP, 2021)

OWASP (2021), diz que, partindo da quinta posição, para líder do ranking, 94% dos aplicativos foram testados para alguma forma de controle de acesso quebrado, com uma taxa de incidência média de 3,81% e tem o maior número de ocorrências no conjunto de dados, contribuído com mais de 318k.

SIEMBA (2021) diz que muitos gigantes do negócio já foram vítimas deste ataque, devido à falta de implementação de um mecanismo de controle de acesso adequado, levando a uma autenticação fraca, falta de controle de acesso de nível funcional e gerenciamento de sessão.

Segundo Goldman (2016) dados fiscais como impostos e salários de mais de 431.000 funcionários da rede varejista foram vazados, tornando-se acessíveis através do site W-2 eXpress da empresa Equifax, uma agência de crédito que segundo Krebs (2016) torna os formulários W-2 eletrônicos acessíveis para download para muitas empresas, incluindo a Kroger.

> *"According to a letter Kroger sent to employees dated May 5, thieves were able to access W-2 data merely by entering at Equifax's portal the employee's default PIN code, which was nothing more than the last four digits of the employee's Social Security number and their four-digit birth year. "(*Krebs, 2016)

IDX (2016) resume a relação entre ambas as empresas (Kroger e Equifax Inc.) e o ataque ocorrido, de forma que, a Kroger contratou a Equifax para fornecer seu conveniente sistema eletrônico W-2 aos próprios funcionários da Kroger. Porém, em seu e-mail para funcionários, a gigante do varejo (Kroger) reconheceu que o site W-2 eXpress da Equifax, usa informações de login padrão com base em SSNs (Social Security Number - número de identificação pessoal nos Estados Unidos) e datas de nascimento. Conforme já mencionado por Krebs (2016), já que apenas o ano de nascimento de quatro dígitos era necessário, este foi um ataque relativamente fácil para os criminosos.

Formas de exploração

Abaixo, consta-se algumas vulnerabilidades comuns de controle de acesso OWASP (2021):

- Violação do princípio de privilégio mínimo ou negação por padrão, onde o acesso deve ser concedido apenas para determinados recursos, funções ou usuários, mas por falta de tratamento, se encontra disponível para qualquer pessoa.
- Omissão de verificações de controle de acesso, a partir de modificações diretas na URL (como adulteração de parâmetros), no estado interno da aplicação, na página HTML ou utilizando uma ferramenta de ataque que modifica as solicitações que vão para a API.
- Permissão da visualização ou edição da conta de outro usuário, fornecendo seu identificador exclusivo.
- Acesso à API sem controles de acesso de métodos de solicitação HTTP POST, PUT e DELETE.

- Elevação de privilégio. Agir como um usuário sem estar logado ou agir como um administrador quando logado como um usuário. Exemplo: Forçar a navegação para determinadas páginas, autenticado como um usuário não autenticado, ou navegar para páginas privilegiadas como um usuário padrão.
- Manipulação de metadados, como reproduzir ou adulterar um token de controle de acesso JSON Web Token (JWT), um cookie ou um campo oculto manipulado para elevar privilégios ou abusar da invalidação de JWT.
- A configuração incorreta do CORS permite o acesso à API de origens não autorizadas / não confiáveis.

Mitigação

Segundo a HACKSPLANNING, não existe uma solução única para implementar corretamente o controle de acesso, basta que sua estratégia de controle de acesso abranja três aspectos:

- **Autenticação**: Identificar corretamente um usuário quando ele retorna ao aplicativo.
- **Autorização**: Decidir quais ações um usuário deve e não deve ser capaz de executar depois de ter sido autenticado.
- **Verificação de permissão**: Avaliar a autorização no momento em que um usuário tenta executar uma ação.

Como forma de prevenção, a OWASP (2021) sugere que os tratamentos de controles de acessos devem ser aplicados do lado do servidor confiável, de forma que o invasor não consiga realizar modificações na verificação de controle de acesso. A fundação, lista os seguintes pontos como forma de mitigação:

- Exceto para recursos públicos, negar por padrão.
- Implementação e reutilização de mecanismos de controle de acesso em todo o aplicativo, incluindo a minimização do uso de Cross-Origin' Resource Sharing (CORS).
- Os controles de acesso ao modelo devem impor a propriedade do registro, em vez de aceitar que o usuário possa criar, ler, atualizar ou excluir qualquer registro.
- Os requisitos de limite de negócios de aplicativos exclusivos devem ser impostos por modelos de domínio.

- Desativar a lista de diretórios do servidor da web e certificar de que os metadados do arquivo - por exemplo o arquivo *.git* – arquivo do sistema GIT de controle de versão de arquivos do projeto - e os arquivos de backup não estejam presentes nas raízes da web.
- Registro de falhas de controle de acesso e notificação aos administradores da aplicação, quando apropriado (por exemplo, falhas repetidas).
- Limite de taxa ao acesso da API e do controlador, afim de minimizar os danos de um possível conjunto de ferramentas de ataque automatizado.
- Os identificadores de sessão com estado devem ser invalidados no servidor após o logout. Os tokens JWT sem estado devem ter vida curta para que a janela de oportunidade para um invasor seja minimizada. Para JWTs de longa duração, é altamente recomendável seguir os padrões OAUTH para revogar o acesso.

4.2 CRYPTOGRAPHIC FAILURES:

Anteriormente conhecida como Exposição de dados confidenciais (*Sensitive Data Exposure*) como mostra na Figura 2 da seção 3, segundo a OWASP (2021) este é mais um sintoma amplo do que uma causa raiz, onde esse ajuste de categoria está focado nas falhas relacionadas à criptografia (ou falta dela). O que muitas vezes leva à exposição de dados confidenciais.

Segundo dados disponibilizados pela OWASP (2021), 79,33% das aplicações testadas detectaram falhas criptográficas, apontando uma taxa de incidência máxima de 46,44%, e uma taxa de incidência média de 4,49%, com um total de 233.788 ocorrências.

Um grande caso a ser citado, conforme a IMMUNIWEB (2021), é a violação massiva contra a Equifax, em 29 de julho de 2017, que causou o roubo de dados altamente sensíveis de 143 milhões de usuários nos Estados Unidos – cerca de 44% da população. De acordo com Ivanova (2017), o ex-CEO da empresa repassou a um comitê do congresso que a empresa utiliza muitas técnicas de proteção de dados, porém esses dados específicos não foram criptografados em repouso.

IMMUNIWEB (2021) apresenta um raciocínio em que, tecnicamente, uma exposição de dados confidenciais pode ser causada por meio de vários outros riscos, como Injeções, quebra de controle de acesso, etc. Porém, apesar das vulnerabilidades que precedem o roubo de dados e custos subsequentes após o roubo de dados, existe fundamentalmente apenas um risco: a exposição real desses dados. Dessa forma, IMMUNIWEB (2021) conclui que se os dados forem realmente mantidos em segurança, e devidamente criptografados, as demais vulnerabilidades passariam a contar menos, e as consequências seriam muito menores.

IMMUNIWEB (2021) diz que a criptografia pode ser dividida em algumas vertentes, onde primeiramente, a criptografia dispõe de dois estados: em repouso (ou seja, em armazenamento); e em trânsito (ou seja, sendo transportados de um local para outro). Pode-se completar conforme diz a OWASP (2021), em que, determinar as necessidades de proteção dos dados em trânsito e em repouso deve ser o primeiro passo da implementação de criptografia de dados, principalmente se esses dados se enquadrarem nas leis de privacidade, por exemplo, Regulamento geral de proteção de dados (GDPR). Pode-se citar também a própria Lei geral de proteção de dados (LGPD) no Brasil.

Formas de exploração

De acordo com uma análise realizada por HOLMWOOD (2021), sobre as várias formas de ataque descritas pela OWASP (2021), foram divididas três categorias, conforme abaixo:

1. **Operacional**:

    - Chaves criptográficas fracas, falta de gerenciamento ou falta de rotação das mesmas, levando a um acesso não autorizado, como legítimo.
    - Downgrades de protocolos. Segundo Educalingo na computação, o termo Downgrade refere-se ao retorno de um software para uma versão anterior, ou seja, é o oposto da atualização. Estes podem ser rebaixados para remover recursos não utilizados ou com erros, e aumentar a velocidade e/ou a facilidade de uso. Partindo para uma definição focada em Downgrades de protocolos, segundo Naziridis (2021), downgrades podem ser considerados também como ataques de rede, que forçam os computadores a abandonar um tipo seguro de conexão, como uma conexão criptografada, recorrendo a versões mais antigas e vulneráveis.
    - Mensagens de erro que podem levar a inferências sobre os dados sendo transportados.
    - Transmissão de dados em texto não criptografado.

2. **Uso incorreto de criptografia:**

    - Aleatoriedade insuficiente, o que pode tornar o texto cifrado previsível.
    - Uso de algoritmo ou protocolo criptográfico antigo ou fraco.

3. **Má criptografia:**

    - Uso de funções hash obsoletas, como MD5 ou SHA1, ou funções hash não criptográficas usadas quando funções hash criptográficas são necessárias.

Mitigação

A mitigação de falhas criptográficas pode ser realizada de diversas maneiras, dependendo de seu escopo e da análise realizada sobre os dados do sistema. Segundo a CIPHERSTASH (2021), os destaques são:

- Classificação dos dados processados, armazenados ou transmitidos por um aplicativo, com o objetivo de entender quais dados serão necessários, e para defender e identificar os controles apropriados.
- Criptografia de todos os dados classificados como "confidenciais".
- Utilização de uma criptografia que forneça sigilo de encaminhamento, para garantir que os dados criptografados no passado não possam ser descriptografados caso as chaves de sessão sejam expostas.
- Verificação da eficácia das configurações de criptografia de forma independente.

4.3 INJECTION

WALLARM (2017) diz que um ataque de injeção ou *Injection*, é quando usuários com intenções maliciosas injetam sua própria entrada maliciosa através de quaisquer entradas de dados possíveis, que sejam controláveis pelo próprio usuário, incluindo variáveis de ambiente, parâmetros, serviços da web externos e internos, funcionalidade de importação ou qualquer outra possibilidade para os usuários inserirem seus próprios vetores de ataque. WALLARM (2017), complementa que estas entradas maliciosas vindas do usuário levam a execução remota do código, que pode ocasionar a perda de dados, ações não autorizadas, corrupção de dados e muito mais.

Este risco pode ser explorado de diversas formas, a partir de quaisquer processos que recebam a entrada do usuário. O ataque de injeção SQL ou *SQL Injection*, é uma das mais comuns maneiras. (WALLARM, 2017)

PINTO e col. (2019) dizem que a injeção SQL é um tipo ataque que ocorre quando usuários com intenções maliciosas, injetam um código malicioso a partir de formulários de entrada de dados, sem validação. Conforme PINTO e col. (2019), pode-se citar alguns exemplos de entradas de dados de usuários, como, formulários de cadastro como nome, data de nascimento, endereço, também formulários de login ou campos de pesquisa presentes em websites.

De acordo com uma pesquisa de Akamai, provedora de serviços em nuvem com sede nos Estados Unidos:

> (...) *"SQL injection and Local Filer Inclusion attacks comprised over 85 percent of the attack vectors recorded. SQL injection attacks accounted for 65 percent of web-based attack vectors from November 2017 to March 2019. "*(TECHMONITOR, 2019)

Em uma análise comparativa, a partir da Figura 2, seção 3, além de sua queda no posicionamento de 2017 a 2021, observa-se que foram fundidas duas categorias principais em uma só, unificando INJECTION e CROSS-SITE SCRIPTING. O que é coerente, já que o CROSS-SITE SCRIPTING embora seja diferente de outros métodos de injeção do lado do servidor, é afinal, um tipo de ataque de injeção. (HKCERT, 2021)

A respeito do CROSS-SITE SCRIPTING, KirstenS (2021) diz que:

> *"Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user."* (KirstenS, 2021)

Em conformidade com KirstenS (2021), após uma brecha de execução de script maliciosa ser devidamente explorada por um atacante, o navegador do usuário, por achar que o script veio de uma fonte confiável, irá permitir que o script possa acessar qualquer cookie, token de sessão ou outras informações confidenciais retidas pelo navegador e usadas naquele site.

Considerado estatisticamente o maior e mais frequente risco, a injeção se manteve em primeiro lugar no ranking OWASP Top 10 em 2017. Em uma análise comparativa, no ano de 2021, a ex-líder dos rankings de 2017 deslizou para a terceira posição no ranking, na qual, segundo a OWASP, 94% das aplicações testadas apontaram alguma forma de injeção com uma taxa de incidência máxima de 19%, uma taxa de incidência média de 3% e 274k ocorrências. (OWASP, 2021)

IMMUNIWEB (2021) aponta que o impacto desse tipo de ataque pode variar a partir das funcionalidades da aplicação e do tipo da injeção que está sendo explorada, porém um ataque bem sucedido pode ser o causador de perda de dados, ações não autorizadas, corrupção de dados e ganho de acesso a dados sensíveis de usuários, como ocorreu com a violação de Heartland em 2008.

MESSMER (2009) diz que o ataque a Heartland Payment Systems comprometeu os dados dos cartões que cruzaram a rede varejista, juntamente com detalhes pessoais de muitos clientes. Segundo a IMMUNIWEB (2021), apesar dos dados exatos não terem sido divulgados, este foi por muitos anos apontado como a maior violação de dados de todos os tempos.

Formas de exploração

Segundo a OWASP (2021), uma aplicação é vulnerável a ataques de injeção quando:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

- Consultas dinâmicas ou chamadas não parametrizadas sem escape ciente do contexto são usadas diretamente no interpretador.
- Dados hostis são usados nos parâmetros de pesquisa de mapeamento relacional de objeto (ORM - Object Relational Mapping Tools) para extrair registros confidenciais adicionais.
- Dados hostis são usados diretamente ou concatenados. O SQL ou comando contém a estrutura e os dados maliciosos em consultas dinâmicas, comandos ou procedimentos armazenados.

Mitigação

OWASP (2021) aponta que para prevenir a injeção, deve-se manter os dados separados dos comandos e consultas, onde logo abaixo elencamos algumas maneiras de mitigar este problema, de acordo com a própria OWASP:

- A utilização de uma API segura, evitando o uso do interpretador SQL inteiramente, fornecendo uma interface parametrizada com Object Relational Mapping Tools (ORMs). ORMs, segundo Cadu (2011), é uma técnica de mapeamento de objeto relacional que permite fazer uma relação dos objetos de classes mapeadas com os dados que os mesmos representam em seu devido banco de dados.
- Para quaisquer consultas dinâmicas, realizar tratativa de escape para caracteres especiais, usando a sintaxe de escape específica para esse interpretador. No entanto, esta forma de prevenção pode não cobrir totalmente um ataque de Injection, já que estruturas SQL, como nomes de tabelas, nomes de colunas e assim por diante, não podem ter escape. Portanto, nomes de estruturas fornecidos pelo usuário são perigosos, e mesmo um tratamento como este, não seria capaz de cobrir esta falha. Este é um problema comum em softwares de elaboração de relatórios.
- Uso do LIMIT e outros controles SQL em consultas para evitar a divulgação em massa de registros no caso de injeção de SQL. GALLAGHER (2020) diz que o LIMIT é uma clausula SQL, que restringe quantos registros serão retornadas a partir de uma consulta SQL.

4.4 INSECURE DESIGN

Segundo a OWASP (2021), o Design Inseguro (Insecure Design) é uma nova categoria para 2021, que se concentra nos riscos relacionados a falhas de design e arquitetura, onde busca alertar os desenvolvedores para que façam mais uso de modelagem de ameaças, padrões de design seguros e arquiteturas de referência, a fim de antecipar ataques e aplicar medidas preventivas para proteger adequadamente as aplicações Web.

Com uma cobertura de 40 CWEs mapeados e um total de 262.407 ocorrências, a partir dos dados de testes disponibilizados pela OWASP (2021), segundo a própria fundação, o design inseguro não deve ser considerado como a fonte de todas as outras 10 categorias de riscos principais, pois há uma diferença entre design inseguro e implementação insegura.

HKCERT (2021) menciona que a lógica por traz de ferramentas automatizadas, se não devidamente controladas podem causar riscos de segurança para as empresas que as utilizam. O incidente de vazamento de dados no Facebook no início de 2021, é um exemplo disso. Segundo RIGUES (2021), o vazamento de dados de 533 milhões de usuários foi causada por atores maliciosos que abusaram de uma ferramenta de importação de contatos para obter uma quantidade limitada de dados sobre um perfil. RIGUES (2021), complementa:

> *"Segundo a empresa, os dados não são resultado de uma invasão aos seus sistemas, mas sim do uso de uma técnica conhecida como scraping (raspagem ou coleta de dados), onde ferramentas automatizadas são usadas para coletar dados em páginas e perfis publicamente disponíveis."*

Com isso pode-se concluir que o mau uso de uma ferramenta também pode ser um risco, e não somente ataques a partir de vulnerabilidades diretas. A principal causa dessa violação foi um uma arquitetura insegura aplicada no uso da ferramenta. Um design de aplicação mal elaborado pode levar a graves consequências para um negócio.

Design Seguro x Design Inseguro

Segundo descrito pela OWASP (2021), um design seguro pode ter defeitos de implementação que levam a vulnerabilidades que podem ser exploradas. Já um design inseguro não pode ser corrigido nem mesmo por uma implementação perfeita, pois, por definição, na arquitetação base do projeto, os controles de segurança necessários não foram criados para a defesa contra ataques específicos.

Mitigação

IMMUNIWEB (2021) lista algumas etapas que podem ser úteis na projeção de uma aplicação:

- Implementação SDLC para ajudar a avaliar e projetar controles relacionados à segurança e à privacidade.
- Implementação de uma biblioteca de padrões de design seguros para utilização na aplicação.
- Avaliação do processo de autenticação de aplicativos, controle de acesso, lógica de negócios e a utilização da modelagem de ameaças para identificação de possíveis vetores de ataque.
- Criação de testes de integração e verificações para validar se todos os fluxos críticos são resistentes ao modelo de ameaça projetado.
- Atenção aos recursos computacionais e ao limite do consumo pelo usuário do serviço

4.5 SECURITY MISCONFIGURATION

Segundo MANAGEENGINE, Configurações incorretas de segurança ou *Security Misconfiguration* são controles de segurança que são configurados incorretamente ou de forma insegura, colocando sistemas e dados em risco, como alterações de configuração mal documentadas ou a utilização de configurações padrão.

De acordo com dados disponibilizados pela OWASP (2021) 90% dos aplicativos foram testados para alguma forma de configuração incorreta, com uma taxa de incidência média de 4%, e mais de 208 mil ocorrências de Enumerações de Fraquezas Comuns (CWEs), nesta categoria de risco, dentre elas. A Fundação complementa que com muitas mudanças em softwares altamente configuráveis, não é surpreendente ver essa categoria subir.

Em uma análise comparativa a partir da Figura 2, na seção 3, é possível observar que a categoria XML External Entities (XXE), presente no 4º lugar da edição de 2017, foi contida em Security Misconfiguration. Segundo Vicente (2019), a essência dos riscos de injeção de XXE é uma configuração incorreta nos servidores que aceitam XML como entrada de aplicativos *clients* não confiáveis, na qual, invasores abusam desse aspecto para injetar cargas úteis de XMLs personalizados, para extrair dados, falsificar solicitações do lado do servidor (SSRF), ou conduzir tentativas de negação de serviço (DoS).

Segundo IMMUNIWEB (2021), um relatório de abril de 2018 da IBM apontou algumas mudanças interessantes nas tendências de segurança ao longo de 2017.

> *"IBM reports that breaches related to bad configuration jumped by 424% in 2018, accounting for nearly 70% of compromised records over the year."* (IMMUNIWEB, 2021)

Formas de exploração

OWASP (2021) informa que um aplicativo pode ser vulnerável nessa categoria se contemplar alguns critérios, onde pode-se citar os destaques como:

- Falta de proteção de segurança apropriada ou permissões configuradas incorretamente em serviços em nuvem.
- Recursos desnecessários são ativados ou instalados.
- Contas e senhas padrão ainda ativas e inalteradas.
- O tratamento e mensagens de erros revelam erros excessivamente informativos aos usuários.
- Para sistemas atualizados, recursos mais recentes de segurança desabilitados ou não configurados com segurança.
- As configurações de segurança de Frameworks (Spring, ASP.NET) e de bibliotecas não definidas para proteger os valores da aplicação.

- Software desatualizado ou vulnerável (Consulte a seção 3.6 – VULNERABLE AND OUTDATED COMPONENTS).

Mitigação

IMMUNIWEB (2021) busca acalmar esta situação da seguinte maneira:

> *"Fortunately, most security misconfiguration risks are known and documented. This means automated testing resources are especially useful for detecting this kind of problem, especially if the testing tools are well-integrated into".*

Porém, mesmo com as mais atualizadas documentações de ferramentas e tecnologias, este tipo de falha sempre pode vir a acontecer. Dessa maneira, abaixo a OWASP (2021) busca apresentar algumas formas de prevenção:

- Um processo de proteção repetível torna mais rápido e fácil implantar outro ambiente que esteja devidamente bloqueado. Os ambientes de desenvolvimento, controle de qualidade e produção devem ser todos configurados de forma idêntica, com credenciais diferentes usadas em cada ambiente. Este processo deve ser automatizado para minimizar o esforço necessário para configurar um novo ambiente seguro.
- Uma plataforma mínima sem recursos, componentes, documentação e amostras desnecessários torna-se inviável e de risco.
- Remoção de recursos e estruturas não utilizados.
- Uma tarefa para revisar e atualizar as configurações apropriadas para todas as notas de segurança, atualizações e patches como parte do processo de gerenciamento de patches (Consulte a seção 3.6 – VULNERABLE AND OUTDATED COMPONENTS).
- Revisão de permissões de armazenamento em nuvem (por exemplo, uma aplicação que utiliza a ferramenta AWS S3, verificar as permissões dos buckes S3).
- Uma arquitetura de aplicativo segmentada fornece separação eficaz e segura entre componentes ou locatários, com segmentação, conteinerização ou grupos de segurança em nuvem (ACLs). Segundo a AWS (2021) Network ACL, é uma lista de controle de acesso à rede, que consiste numa camada de segurança opcional para uma VPC (Amazon Virtual Private Cloud), que funciona como um firewall para controlar o tráfego de entrada e saída de uma ou mais sub-redes.

- Envio de diretivas de segurança para aplicações clients, por exemplo, cabeçalhos de segurança. Por exemplo, como será citado no tópico de mitigação da seção 4.7 IDENTIFICATION AND AUTHENTICATION FAILURES, onde com uma diretiva de segurança, é possível definir que se um usuário errar a senha três vezes, dentro de um período de meia hora, a sua conta deve ficar bloqueada até que um Administrador desbloqueie.
- Um processo automatizado para verificar a eficácia das configurações em todos os ambientes.

4.6 VULNERABLE AND OUTDATED COMPONENTS

Na última edição OWASP TOP 10 2017 (OWASP, 2017), nomeado como Uso de Componentes com vulnerabilidades conhecidas *(Using Components with Known Vulnerabilities)*, esta categoria passou da 9ª posição em 2017 para a 6ª posição em 2021, no qual consiste em um problema conhecido que de acordo com a OWASP (2021), houve dificuldades em testar e avaliar os riscos para esta categoria. Além de ser a única categoria que não tem nenhuma Vulnerabilidade e Exposições Comuns (CVEs Common Vulnerabilities and Exposures) mapeada para os CWEs (Common Weakness Enumeration) incluídos.

Segundo a OWASP (2021), esta categoria foi a segunda colocada na pesquisa da comunidade Top 10, porém apesar de sua alta relevância a partir da pesquisa designada pelos especialistas que colaboraram para com a sua análise, a mesma já havia dados suficientes para se enquadrar no índice Top 10.

De acordo com Nayak (2021), essa mudança de status significativa reflete a importância crescente dessa vulnerabilidade no desenvolvimento de aplicações modernas e a preocupação crescente em como a comunidade de segurança passará a ver esse risco.

IMMUNIWEB (2021) diz que atualmente o número de aplicativos que fazem o uso de componentes pré-existentes em vez de serem totalmente codificados do zero é cada vez maior, onde este problema é cada vez mais grave, devido a tendência de desenvolvedores utilizarem cada vez mais componentes de código aberto expostos à comunidade. Sob a pressão da entrega, estes componentes, por vezes, não são suficientemente verificados.

Uma publicação realizada por UCHILL (2021) para a revisa SC Magazine, menciona que Chris Wysopal - fundador e diretor de tecnologia do testador de segurança de aplicativos automatizado Veracode - estima que 90% das aplicações modernas utilizam componentes de código aberto. UCHILL (2021) complementa que mesmo perante a estes riscos, o código fonte aberto é onipresente nas aplicações por vários motivos, como a economia de tempo e dinheiro durante o desenvolvimento de uma funcionalidade ou execução de testes.

Formas de exploração

De acordo com a OWASP (2021), uma aplicação é vulnerável à esta categoria a partir de alguns fatores, conforme será citado abaixo:

- Desconhecimento das versões dos componentes e dependências que são utilizadas na aplicação (tanto do lado do cliente quanto do lado do servidor).
- Software vulnerável, sem suporte ou desatualizado, podendo incluir também toda sua estrutura externa, como Sistema Operacional, Servidor Web, Sistema de Gerenciamento de Banco de Dados (DBMS), APIs e bibliotecas.
- Ausência da execução de varredura de vulnerabilidades regularmente.
- Ausência de correção da plataforma, estruturas e dependências subjacentes de maneira oportuna e baseada em riscos. Isso geralmente acontece em ambientes em que a correção é uma tarefa mensal ou trimestral sob controle de alterações, deixando as organizações abertas a dias ou meses de exposição desnecessária, a vulnerabilidades corrigidas.
- Ausência de testes de compatibilidade de bibliotecas atualizadas ou patches.
- Ausência de proteção das configurações dos componentes, dependências e ferramentas auxiliares (Categoria citada na seção 3.5 SECURITY MISCONFIGURATION).

Mitigação

IMMUNIWEB (2021) diz que a atenção a respeito de qualquer vulnerabilidade em componentes de código aberto continua sendo um grande problema e, complementa, que a conscientização é a melhor defesa de uma empresa contra riscos de vulnerabilidades conhecidas.

Segundo IMMUNIWEB (2021):

> *"A web application firewall (WAF) can also be employed since defence in depth is always a good principle. But WAF is no silver bullet, and researchers have repeatedly demonstrated they can frequently be bypassed by competent attackers"*

OWASP (2021), cita algumas formas de mitigação para este risco, como:

- Remoção de dependências não utilizadas, recursos, componentes, arquivos e documentações desnecessárias.
- Levantamento contínuo das versões dos componentes do lado do cliente e do lado do servidor (por exemplo: estruturas, bibliotecas) e suas dependências, usando ferramentas como OWASP Dependency-Check, retire.js, etc.

- Monitoramento de bibliotecas e componentes sem manutenção ou que não possuem lançamentos de patches de segurança para versões anteriores. Se não for possível obter patches pelo responsável pela biblioteca, por exemplo, considerar implantar um patch virtual para monitorar, detectar ou proteger contra o problema descoberto.
- Coletar componentes e dependências apenas de fontes oficiais, por meio de links seguros.

4.7 IDENTIFICATION AND AUTHENTICATION FAILURES

Segundo a IBM (2021), *Identificação* se resume à capacidade de constatar exclusivamente um usuário de um sistema ou um aplicativo em execução no sistema. *A autenticação* consiste na capacidade de provar que um usuário ou aplicativo é genuinamente quem essa pessoa ou aplicativo alega ser. Esta categoria trata-se das vulnerabilidades intrínsecas a estes dois componentes.

De acordo com a Figura 2, na seção 3, esta categoria decaiu da 2ª posição (anteriormente nomeada como *Broken Authentication* – Autenticação Quebrada no ranking 2017) para a 7ª posição no ranking 2021, ficando atrás apenas dos ataques de injeção. Sua alteração na nomenclatura diz respeito ao mapeamento de novas CWES relacionadas à falha de identificação, conforme mencionado pela OWASP (2021).

Formas de exploração

IMMUNIWEB (2021) realiza uma análise baseada na divisão dos três padrões de ataque que exploram a vulnerabilidade de Autenticação Fraca, segundo a OWASP (2021), que são: *Credential Stuffing*, *Brute Force Access* e *Session hijacking*.

1. **Credential Stuffing**: Consiste no uso de ferramentas automatizadas para testar uma lista de nomes de usuários e senhas válidos, roubados de uma aplicação, contra outra aplicação. Normalmente estas listas são obtidas também a partir da "Dark Web", que são nada menos que frutos de vazamentos de credenciais de usuários de outras empresas no passado, que consequentemente são utilizados para realização de novos ataques em novos alvos. IMMUNIWEB (2021) menciona que:

   > *"In 2017, researchers discovered a file on the dark web containing 1.4 billion compromised username and password combinations, in plain text format. These were compiled from numerous earlier breaches and made available for anyone to use."*

2. **Brute Force Access**: Ataques de força bruta podem ser definidos técnicamente como o procedimento de tentativa de todas as possibilidades de senhas e usuários diferentes até que ambos sejam encontrados. Em alguns casos, este método pode ser desnecessário, já que os invasores utilizam primeiramente listas de senhas mais comuns, para tentativa de acesso.

3. **Session hijacking**: Aplica-se no cenário onde há a exploração de uma sessão de um usuário legítimo autenticado. Após o usuário se autenticar, é gerado um ID ou token de sessão, onde este é armazenado em algum local na máquina do usuário (como cookie de sessão, Local Storage – armazenamento local de dados de um navegador Web - ou até mesmo anexado a URL do navegador). Posteriormente, quando este usuário efetuar o logout da sessão, o token ou ID de sessão deve ser removido também. Caso o mesmo não seja removido, este pode ser aproveitado por um invasor, permitindo-o "sequestrar" a sessão do usuário legítimo e executar qualquer ação permitida a este usuário.

Mitigação

Com base nas formas de prevenção sugeridas pela OWASP, é possível destacar os seguintes pontos:

- Implementação de autenticação multifatores, para evitar o autopreenchimento de credenciais, ataques de força bruta e ataques de reutilização de credenciais roubadas.
- Não utilização de credenciais padrão, especialmente para usuários administradores.
- Implementação de verificações de senha fracas, como também validar senhas baseadas em históricos do usuário, ou em relação a lista de senhas inseguras, como cita a GATEFY (2019) no ranking das 100 piores senhas de 2019.
- Mapeamento de complexidade e políticas de rotação de senha com as diretrizes do Instituto Nacional de Padrões e Tecnologia (NIST) 800-63b, cuja a seção 5.1 das Diretrizes de Identidade Digital publicada pela NIST (Atualizada em 2020) apresenta políticas de senha modernas, baseadas em evidências e regras de aceite. NIST (2019)
- Garantir que os PATHs das páginas de registro, recuperação de credencial e API sejam protegidos contra ataques de enumeração de contas usando as mesmas mensagens para todos os resultados. Segundo MICROSOFT (2021), um ataque de enumeração de contas consiste em um cenário cujo um invasor usa um dicionário com milhares de nomes de usuário, ou ferramentas para tentar adivinhar nomes de usuário no domínio. Este tipo de ataque pode ser ainda mais preciso no cenário conforme diz KASPERSKY (2021), onde, caso o hacker consiga verificar se determinado usuário está cadastrado na aplicação, bastará configurar um ataque de força bruta para encontrar a senha correspondente, economizando tempo e esforço (apud TIINSIDE, 2021).
- Limitar e atrasar cada vez mais as tentativas de login malsucedidas, porém é necessário cuidado para não criar um cenário de negação de serviço.
- Registro de todas as falhas e alerta aos administradores quando o enchimento de credenciais, força bruta ou outros ataques forem detectados.
- Utilização de um gerenciador de sessão integrado e seguro do lado do servidor que gera um novo token de sessão aleatório com alta entropia após o login. O identificador de sessão não deve estar no URL, e ser armazenado com segurança e invalidado após o logout.

4.8 SOFTWARE AND DATA INTEGRITY FAILURES

Com base na Figura 2, da seção 3, a categoria de desserialização insegura, apresentada na 8ª posição da última edição TOP 10, foi re-enquadrada nesta nova categoria e, segundo a OWASP (2021), esta trata-se de uma nova categoria, focada em realizar suposições relacionadas à falta de verificação de integridade durante atualizações de software, dados críticos e pipelines de CI / CD.

Segundo HKCERT (2021), este risco relaciona-se ao código ou infraestrutura não protegida contra violações de integridade, e pode ser considerado um dos mais impactantes. HKCERT (2021) complementa que um exemplo disso, é a violação da SolarWinds Orion, ocorrida em dezembro de 2020.

SolarWinds é uma empresa que produz softwares de soluções de gerenciamento de rede, sistemas, banco de dados e segurança de TI (SOLARWINDS, 2021), na qual segundo Turner (2020), um de seus produtos de gerenciamento de rede, chamado Orion, sofreu um ataque, cujo os agentes maliciosos incorporaram código malicioso ao software, adulterando-o antes que uma atualização fosse lançada para todos os seus usuários, o que resultou na implantação de malware para os clientes da SolarWinds.

Segundo AIQON (2020), os responsáveis por este ataque foram Hackers russos, pertencentes a um grupo de ameaça APT29 conhecido como Cozy Bear, no qual utilizaram o tipo de ataque conhecido como "supply chain attack", ou Ataque de cadeia de suprimentos.

Pode-se citar por exemplo o produto Orion da empresa SolarWinds. Com isso, após um malware instalado, os atacantes por vezes conseguem obter controle total sobre as redes do cliente do fornecedor.

Com base nisso, conclui-se que este ataque veio a acontecer devido à falta de um processo eficaz de verificação de integridade do código. HCKERT (2021) diz que este código malicioso foi entregue a mais de 18.000 organizações da SolarWinds, causando um dos ataques cibernéticos mais sérios até hoje.

Formas de exploração

Conforme citado, uma das principais formas de ataque dessa categoria pode ser nas atualizações de software automáticas. Segundo IMMUNIWEB (2021), atualmente muitas empresas aderem para suas aplicações, essa funcionalidade de atualização automática de software, o que pode levantar certas preocupações sobre a integridade dos dados durante o processo de atualização. IMMUNIWEB (2021) cita que um ataque MitM também pode ser uma forma de ataque, no qual se bem sucedido, o invasor pode enviar um código malicioso para o aplicativo durante o processo de atualização do software.

CWE-494 (2021), cita uma vertente desta forma de ataque descrita no último parágrafo, conforme abaixo:

> *"The product downloads source code or an executable from a remote location and executes the code without sufficiently verifying the origin and integrity of the code."*

Além do vetor de ataque citado acima, pode-se citar também o problema na desserialização de dados não confiáveis. CWE-502 (2021) cita esta vulnerabilidade como:

> *"The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid. "*

Com base na citação acima, para fonte de informação, a desserialização e serialização, são mencionadas na seção 2.14 SERIALIZAÇÃO E DESSERIALIZAÇÃO, em conceitos básicos.

Dessa forma, a partir dos ataques de desserialização de dados não confiáveis, CWE-502 (2021), define algumas consequências para a integridade da aplicação, onde os invasores podem modificar objetos ou dados inesperados que foram considerados protegidos contra modificação, e também complementa sobre consequências para a disponibilidade da aplicação, como um cenário de uma função da aplicação que faz uma suposição sobre quando terminar, baseada no conteúdo de um atributo do tipo String. Esta função poderia facilmente nunca terminar, podendo causar graves problemas no desempenho da aplicação.

Mitigação

OWASP (2021) sugere algumas formas de mitigação para este risco:

- Utilização de assinaturas digitais ou mecanismos semelhantes para verificar se o software ou os dados são da fonte esperada e não foram alterados.
- Certificar-se das bibliotecas e dependências, como npm ou Maven. O Maven é uma ferramenta de construção e resolução de dependências popular para a linguagem Java, assim como o NPM é para a linguagem Javascript.
- Utilização de uma ferramenta de segurança da cadeia de suprimentos de software, como OWASP Dependency Check ou OWASP CycloneDX, seja usada para verificar se os componentes não contêm vulnerabilidades conhecidas.
- Revisão do código-fonte da aplicação, para evitar alterações de configuração indesejadas, ou introdução de código malicioso no software.
- Certificar-se que a pipeline de CI/CD tenha segregação, configuração e controle de acesso adequados, a fim de garantir a integridade do código que flui pelos processos de construção e implantação.
- Certificar-se que nenhum dado serializado ou não criptografado não seja enviado a clientes não confiáveis sem alguma forma de verificação de integridade ou assinatura digital para detectar adulteração ou repetição dos dados serializados.

4.9 SECURITY LOGGING AND MONITORING FAILURES

A 9ª posição do ranking OWASP TOP 10 (2021), a partir da Figura 2, seção 3, foi considerada a 5ª categoria da pesquisa da comunidade da última edição de 2017, anteriormente classificada como Insuficient Logging & Monitoring e agora denominada Security Logging and Monitoring Failures.

Conforme a OWASP (2021), a categoria foi expandida para incluir mais tipos de falhas, incluindo falhas que podem afetar diretamente a visibilidade, o alerta de incidentes e a perícia.

Segundo os dados disponibilizados pela OWASP (2021), esta categoria apresentou um total de 53.615 ocorrências para 53,67% das aplicações testadas, com 4 CWEs mapeadas, onde, com base na amplitude dos dados e fontes mapeadas, esta categoria visa ajudar a detectar, escalar e responder às violações ativas. Sem registro e monitoramento, as violações não podem ser detectadas. OWASP (2021)

O relatório sobre custos de violação de dados realizado pelo Instituto Ponemon e pela IBM (2021), aponta que o tempo médio para identificar e conter uma violação de dados é de aproximadamente 287 dias e complementa que, quanto mais tempo demora a identificação, mais cara será as consequências para a organização.

Este relatório citado no parágrafo acima, de acordo com a IBM, oferece percepções de 537 violações reais entre 17 países e regiões, e 17 empresas que providenciam taxas e médias globais, afim de ajudar a compreender os riscos cibernéticos em um mundo de mudança.

IMMUNIWEB (2021) aponta a extensão deste problema, de forma que se uma aplicação web e os incidentes do servidor forem monitorados incorretamente, atividades suspeitas podem facilmente passar despercebidas. E complementa que, um método de registro bem implementado, com alertas sempre que surgirem anomalias e um monitoramento diligente, permite que ações sejam mais rapidamente tomadas contra a exploração de vulnerabilidades.

Formas de exploração

OWASP (2021) descreve alguns dos principais pontos atrelados a esta categoria:

- Ausência de registros em eventos auditáveis, logins com falhas e transações de alto valor, ou se existentes, armazenados apenas localmente.
- Avisos e erros geram mensagens de log inexistentes, inadequadas ou pouco claras.
- Logs de aplicativos e APIs não são monitorados para atividades suspeitas.
- Analisadores de vulnerabilidades e varreduras por ferramentas de teste de segurança de aplicativo dinâmico não acionam alertas.

- Não detecção ou alerta para ataques ativos em tempo real ou quase em tempo real.

Mitigação

OWASP (2021) orienta que os desenvolvedores implementem alguns ou todos os controles a seguir, a depender do risco do aplicativo:

- Todas as falhas de login, controle de acesso e validações de entrada do lado do servidor, devem ser registradas para identificação de contas suspeitas ou maliciosas e retidas por tempo suficiente para permitir análise posterior.
- Geração de logs em um formato flexível, para que as soluções de gerenciamento dos logs possam ser acessadas e consumidas facilmente. IMMUNIWEB (2021) diz que o melhor uso dos logs é a revisão pós-evento e descoberta das áreas da aplicação que foram comprometidas.
- Os dados de registro devem ser codificados corretamente para evitar injeções ou ataques nos sistemas de registro ou monitoramento.
- As transações de alto valor devem possuir uma trilha de auditoria com controles de integridade, para evitar adulteração ou exclusão, como o uso de tabelas de banco de dados limitadas somente à inserção de dados.
- Estabelecimento de monitoramento e alertas eficazes para que atividades suspeitas sejam detectadas e respondidas rapidamente. Normalmente estas ações são realizadas pelas equipes de operações de segurança da informação.
- Estabelecimento e adoção de planos de resposta e recuperação de incidentes, como são apontados pelo o Instituto Nacional de Padrões e Tecnologia (NIST) 800-61r2 ou similares. (NIST, 2012)

Além dos pontos mencionados pela OWASP (2021), IMMUNIWEB (2021), apresenta outras abordagens, na qual sugere que uma das boas práticas de testar riscos de registros inadequados é a partir da utilização de um Pentest, que investigará e tentará violar a aplicação, de forma simulada – como já descrito na seção 2.1.

A respeito dos registros de logs, IMMUNIWEB (2021), cita que sua utilização com criptografia seria a melhor alternativa, porém pode ser um processo caro em termos de desempenho e desenvolvimento.

4.10 SERVER-SIDE REQUEST FORGERY (SSRF)

Na última posição do ranking TOP 10, de acordo com a Figura 2 na seção 3, encontra-se a categoria “Server-Side Request Forgery (SSRF)” ou Falsificação de solicitação do lado do servidor. Conforme a

OWASP (2021), esta é uma nova categoria adicionada a partir da pesquisa da comunidade Top 10, ocupando o primeiro lugar no levantamento realizado pelos especialistas.

De acordo com os dados disponibilizados pela OWASP (2021), de 75% das aplicações testadas para este risco, foi apresentada uma taxa máxima de 2%. Dessa forma, é perceptível a sua baixa taxa de incidência, na qual, embora não esteja ilustrado nos dados, a OWASP (2021) diz que esta categoria representa um cenário indicado pela comunidade de segurança, por ser um risco importante a ser abordado.

Para PORTSWIGGER (2021) um ataque SSRF bem-sucedido pode muitas vezes resultar em ações não autorizadas, acesso a dados dentro da organização ou quaisquer ataques mal-intencionados que levariam os analistas a entender que o problema se origina da organização que hospeda o aplicativo vulnerável. Seja na própria aplicação vulnerável ou em outros sistemas Back-end com os quais a aplicação atacada pode se comunicar.

Segundo IMMUNIWEB (2021), as falsificações de solicitações a partir do servidor vieram a se tornar uma das vulnerabilidades mais discutidas em 2021, devido a grandes danos causados por agentes APT, e por ransomwares.

Pode-se citar um caso de ataque cibernético semelhante que ocorreu com a Microsoft, em março de 2021. Segundo Mackie (2021), os servidores Exchange da empresa – um produto Microsoft, que fornece servidores de e-mail e calendário a pequenas e médias empresas SPRINGPEOPLE (2018) – teriam sofrido um ataque de um ransomware, afetando um total de 400.000 servidores conectados à Internet.

Segundo a SBARAGLIA (2021), após as falhas terem sido descobertas, a Microsoft implementou rapidamente patches de segurança, mas até que as atualizações fossem instaladas, muitas empresas ainda se mantiveram vulneráveis.

Uma análise realizada pela UNIT 42 (2021) apresenta uma visão geral de forma ilustrada sobre a exploração das vulnerabilidades do Microsoft Exchange Server. Segundo informa a empresa, foram mapeadas um total de 4 vulnerabilidades encadeadas, que são:

- CVE-2021-26855
- CVE-2021-26857
- CVE-2021-26858
- CVE-2021-27065

UNIT 42 (2021), descreve o fluxo seguido pelos agentes maliciosos baseados nas CVE's citadas acima, da seguinte forma:

> *"For this attack to be successful, the adversary would first need to identify an on-premises Microsoft Exchange Server that is able to receive untrusted connections from an external*

source on port 443. If an adversary is successful in securing a connection, they can then exploit CVE-2021-26855 to authenticate themselves as a Microsoft Exchange server. This can be followed by the exploitation of CVE-2021-26857, CVE-2021-26858 and CVE-2021-27065 post-authentication, allowing the adversary to gain remote access."

A análise fornecida pela empresa, complementa que se o acesso for obtido, a execução de comandos remotos, ou o upload de um Web Shell – por exemplo Chopper China - que permitiria ao invasor roubar dados e executar ações maliciosas, como baixar arquivos remotos.

A Figura 3 a seguir, apresenta de maneira ilustrada os atores de ameaça responsáveis pela violação e suas atividades descritas:

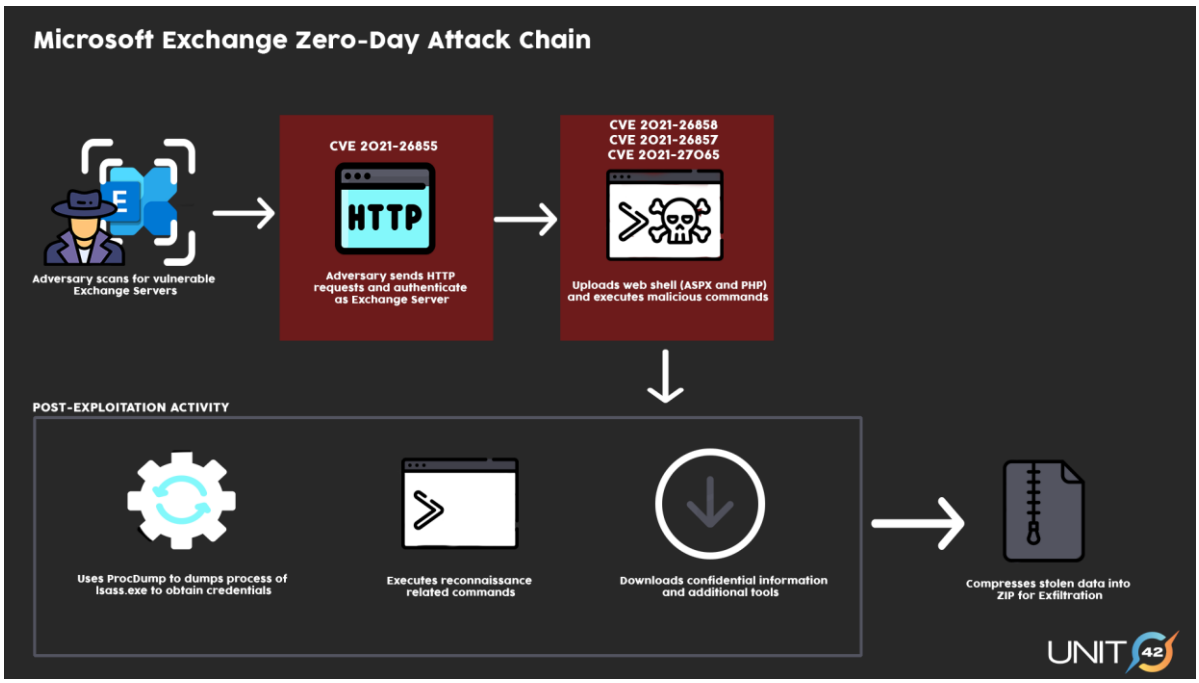

Figura 3. "*Threat actors can chain together four zero-day vulnerabilities to gain unauthorized access to Microsoft Exchange Servers."*

Fonte: UNIT 42 (2021)

Em complemento com a análise fornecida pela UNIT 42 (2021), analisando a Figura 3, o quadro que apresenta *POST-EXPLORATION ACTIVITY*, consiste em apresentar algumas das atividades pós-exploratórias a serem realizadas (também as mais comuns). Conforme apresenta abaixo do ícone de engrenagem, um *Procdump*, teria a função de despejar a memória do processo LSASS (Local Security Authority Subsystem Service - Serviço de Subsistema de Autoridade de Segurança Local), para obter credenciais. Dessa forma, após a obtenção das credenciais, segundo a UNIT 42 (2021), os invasores costumam comprimir esses arquivos de

dados sensíveis em utilitários de compactação (por exemplo: 7zip ou WinRar), para posteriormente exfiltrar estes dados roubados.

A partir disso, é possível apontar que, a primeira e talvez principal vulnerabilidade explorada na violação contra os servidores da Microsoft Exchange, mencionada nos parágrafos acima, foi a CVE-2021-26855, que segundo a GRUNZWEIG e col. (2021), a mesma pode ser descrita como uma vulnerabilidade de falsificação de solicitação do lado do servidor ou *SERVER-SIDE REQUEST FORGERY (SSRF)*, que pode ser executada remotamente, sem nenhuma tipo de autenticação.

Formas de Exploração:

- Comprometimento de serviços internos, no qual o invasor pode abusar dos serviços internos para conduzir outros ataques, como execução remota de código ou negação de serviço (DoS). OWASP (2021);
- Filtragem ausente ou insuficiente da entrada fornecida por um invasor, que posteriormente é usada pelo aplicativo para iniciar uma conexão com uma aplicação terceira. (IMMUNIWEB, 2018);
- Segundo POSTSWIGGER, algumas aplicações transmitem dados em formatos cuja especificação permite a inclusão de URLs, que podem ser solicitados pelo analisador de dados para o formato. Um exemplo disso é a partir da transmissão de dados estruturados do cliente para o servidor no formato de dados estruturados XML. Dessa forma, conforme abordada na seção 3.5 (SECURITY MISSCONFIGURATION), quando uma aplicação aceita e realiza a leitura de dados no formato XML, pode ser vulnerável à injeção XXE (XML External Entity attack) que, por sua vez, passa a ser vulnerável à SSRF via XXE;
- A partir de ataques de Phising, o SSRF pode ser usado em conjunto com redirecionamentos HTTP para redirecionar a vítima a uma página da web maliciosa. Isso pode ser usado para servir malware, afim de colher credenciais (IMMUNIWEB, 2018).

Mitigação

OWASP (2021), sugere algumas formas de prevenção de SSRF, implementado alguns ou todos os controles de defesa que serão citados, conforme abaixo:

- **Da camada de rede:**

1. Segmentação de funcionalidades de acesso a recursos remotos em redes separadas para reduzir o impacto de SSRF.

2. Implementação de políticas de firewall, como “negar por padrão” ou regras de controle de acesso à rede para bloquear todo o tráfego da intranet, exceto o essencial. Segundo Dutran, Chrispim e Ferreira (2019), *negar por Padrão* é uma política conhecida por bloquear todo tipo de dado, seja ele chegando ou saindo da rede, que não está explicitamente permitido pelas regras. A mesma reduz drasticamente a chance de haver ataques, e é uma política considerada mais segura do que permitir que todo tráfego que não está explicitamente proibido passe pelo Firewall.
3. Estabelecimento de uma propriedade e um ciclo de vida para regras de firewall, baseadas em aplicativos.
4. Registrar todos os fluxos de rede aceitos e bloqueados em firewalls (Conforme citado na seção 3.9 SECURITY LOGGING AND MONITORING FAILURES).

- **Da camada de aplicativo:**

1. Limpeza e validação de todos os dados de entrada fornecidos pelo cliente.
2. Implementação do esquema de URL, porta e destino, com uma lista de permissões positiva.
3. Não enviar respostas brutas aos clientes, sempre realizar o refinamento de seu conteúdo.
4. Desativar redirecionamentos HTTP.

5. CONSIDERAÇÕES FINAIS

A partir da análise apresentada, é notável os cenários de vulnerabilidades interligadas entre si, ou mesmo o agrupamento de mais de uma vulnerabilidade sendo explorada simultaneamente ou não. Quanto mais brechas encontradas em uma aplicação, maiores podem ser os impactos e mais difícil será sua mitigação. Dessa forma, este trabalho buscou conduzir cada risco citado, apresentando inicialmente fatores históricos característicos de cada um, e em seguida, partindo para o ponto teórico de exploração e mitigação.

Dentre os riscos expostos, sua abrangência consiste numa abordagem preventiva, porém buscando uma condução ampla de cenários de ciberataques, para que além das tratativas diretas de fragilidades sejam realizadas, as mais viáveis maneiras de investigação e diagnóstico de problemas de segurança sejam implementadas.

O assunto exposto neste trabalho é de grande valia para qualquer profissional de TI, sem ressalvas, e, portanto, espera-se que os dados e as informações expostas contribuam para esclarecer as principais vulnerabilidades em aplicações Web, inteiradas em 2021 e baseadas em uma metodologia de reconhecimento internacional.

Para uma pesquisa futura, voltada ao tema abordado, podem ser exploradas na prática as diversas formas de ataque citadas para cada vulnerabilidade, a partir de ferramentas de testes automatizadas, a fim de relatar e relacionar as fragilidades encontradas, com as indicadas pelo projeto "*OWASP Top 10*".

6. REFERÊNCIAS BIBLIOGRÁFICAS

AIQON. **Ataque à cadeia de suprimentos da SolarWinds**. 2020. Disponível em: < https://aiqon.com.br/blog/ataque-a-cadeia-de-suprimentos-da-solarwinds/>. Acesso em: 13 de nov. 2021.

AUTH0. **JSON Web Tokens**. 2021. Disponível em: <https://auth0.com/docs/security/tokens/json-web-tokens>. Acesso em: 24 de nov. 2021.

AWS. **Network ACLs**. 2021. Disponível em: < https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. Acesso em: 25 de nov. 2021.

BATTISTI, Julio. **Windows 7: Lição 156 - Capítulo 06 - Diretivas Locais de Segurança - Conceito**. 2020. Disponível em: <https://www.juliobattisti.com.br/artigos/windows7/capitulo06/cap06_19.asp>. Acesso em: 25 de nov. 2021.

BLOG DA CCM. **Teste de intrusão: o que é, a importância e como utilizar**. 2019. Disponível em: <https://blog.ccmtecnologia.com.br/post/teste-de-intrusao-o-que-e-a-importancia-e-como-utilizar >. Acesso em: 16 de out. 2021.

Cadu. **ORM: Object Relational Mapper**. DEVMEDIA. 2011. Disponível em: < https://www.devmedia.com.br/orm-object-relational-mapper/19056 >. Acesso em: 05 de nov. 2021.

CINTO, Nícholas Antunes. **Teste de Vulnerabilidades em Aplicações Web**. TCC - Instituto Municipal Educacional do Município de Assis – IMESA e à Fundação Educacional do Município de Assis – FEMA. Assis, p. 63. 2015.

COSTA, Matheus Bigogno. **O que é DoS e DDoS?** CANALTECH. 2014. Disponível em: <https://canaltech.com.br/produtos/o-que-e-dos-e-ddos/>. Acesso em: 25 de nov. 2021.aut

CWE-494. **CWE-494: Download of Code Without Integrity Check**. 2021. Disponível em: <https://cwe.mitre.org/data/definitions/494.html/>. Acesso em: 14 de nov. 2021.

CWE-502. **CWE-502: Deserialization of Untrusted Data**. 2021. Disponível em: < https://cwe.mitre.org/data/definitions/502.html/>. Acesso em: 14 de nov. 2021.

DEVOPEDIA. **Data Serialization**. 2020. Disponível em: <https://devopedia.org/data-serialization/>. Acesso em: 25 de nov. 2021.aut

DUTRA, Renan; NATHALIA, Chrispim; WESLLEY, Ferreira. DEL POLI UFRJ. 2021. **Firewall: Tecnologias**. Disponível em: < https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/firewall/tecnologias.html/>. Acesso em: 20 de nov. 2021

EDUCALINGO. **Downgrade**. 2021. Disponível em: < https://educalingo.com/pt/dic-en/downgrade>. Acesso em: 01 de nov. 2021.

FC BRASIL. **O que é patch e para que serve esse programa?** 2020. Disponível em: < https://www.fcbrasil.com.br/blog-fcb/133-o-que-e-gerenciamento-de-patch-e-como-ele-funciona>. Acesso em: 25 de nov. 2021.

GALLAGHER, James. **SQL Limit: A Beginner's Guide**. CARRER KARMA. 2020. Disponível em: <https://careerkarma.com/blog/sql-limit//>. Acesso em: 05 de nov. 2021.

GATEFY. **Confira a lista das piores senhas de 2019**. 2019. Disponível em: < https://gatefy.com/pt-br/blog/confira-lista-piores-senhas-2019/>. Acesso em: 09 de nov. 2021.

GCSEC. **Ataques à cadeia de fornecimento: o que são?** 2021. Disponível em: < https://gcsec.com.br/ataques-a-cadeia-de-fornecimento-o-que-sao/index.html/>. Acesso em: 14 de nov. 2021.

GOLDMAN, Jeff. Kroger. **Wendy's, Kiddicare Suffer Data Breaches**. eSecurity Planet. 2016. Disponível em: <https://www.esecurityplanet.com/networks/kroger-wendys-kiddicare-suffer-data-breaches/>. Acesso em: 27 de out. 2021.

GRUNZWEIG, Josh; MATTHEW, Meltzer; SEAN. Koessel, Steven Adair, Thomas Lancaster. VOLEXITY. 2021. **Operation Exchange Marauder: Active Exploitation of Multiple Zero-Day Microsoft Exchange Vulnerabilities**. Disponível em: < https://www.volexity.com/blog/2021/03/02/active-exploitation-of-microsoft-exchange-zero-day-vulnerabilities/>. Acesso em: 20 de nov. 2021

HACKSPLANNING. **Ensuring Proper Access Control**. 2021. Disponível em: < https://www.hacksplaining.com/prevention/broken-access-control>. Acesso em: 28 de out. 2021.

HKCERT. **OWASP Top 10-2021 is Now Released.** 2021. Disponível em: < https://www.hkcert.org/blog/owasp-top-10-2021-is-now-released>. Acesso em: 24 de out. 2021.

HOLMWOOD, Lindsay. **Cryptographic Failures is now #2 on the OWASP Top 10**. CIPHERSTASH. Disponível em < https://cipherstash.com/blog/2021-10-13-owasp-top-10-number-2-cryptographic-failures/>. Acesso em: 01 de nov. 2021.

HOSTMIDIA. **O que é MFA?** Por que é importante? 2021. Disponível em: <https://www.hostmidia.com.br/blog/o-que-e-mfa/>. Acesso em: 26 de nov. 2021.aut

IBM. **Security concepts and mechanisms: identification and authentication**. 2021. Disponível em: < https://www.ibm.com/docs/en/ibm-mq/9.0?topic=mechanisms-identification-authentication/>. Acesso em: 09 de nov. 2021.

IBM; PONEMON. **How much does a data breach cost? Cost of a Data Breach Report 2021 explores ways to help mitigate risk**. 2021. Disponível em: <https://www.ibm.com/security/data-breach/>. Acesso em: 15 de nov. 2021

IDX. **The Kroger/Equifax W-2 Breach: What Can We Learn from It?** 2016. Disponível em: < https://www.idx.us/knowledge-center/the-kroger-equifax-w-2-breach-what-can-we-learn-from-it/>. Acesso em: 28 de out. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Cryptographic Failures Practical Overview**. 2021. Disponível em: <https://www.immuniweb.com/blog/OWASP-cryptographic-failures.html/>. Acesso em: 01 de nov. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Identification and**

Authentication Failures Practical Overview. 2021. Disponível em: < https://www.immuniweb.com/blog/OWASP-identification-and-authentication-failures.html/>. Acesso em: 09 de nov. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Injection Practical Overview**. 2021. Disponível em: <https://www.immuniweb.com/blog/OWASP-injection.html/>. Acesso em: 05 de nov. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Insecure Design Practical Overview**. 2021. Disponível em: <https://www.immuniweb.com/blog/OWASP-insecure-design.html/>. Acesso em: 05 de nov. 2021.

IMMUNIWEB. **Monitoring Failures Practical Overview**. 2021. Disponível em: <https://www.immuniweb.com/blog/OWASP-security-logging-and-monitoring-failures.html/>. Acesso em: 15 de nov. 2021

IMMUNIWEB. **OWASP Top 10 in 2021: Security Logging and**

Monitoring Failures Practical Overview. 2021. Disponível em: <https://www.immuniweb.com/blog/OWASP-security-logging-and-monitoring-failures.html/>. Acesso em: 15 de nov. 2021

IMMUNIWEB. **OWASP Top 10 in 2021: Security Misconfiguration Practical Overview**. 2021. Disponível em: < https://www.immuniweb.com/blog/OWASP-security-misconfiguration.html/>. Acesso em: 06 de nov. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Server-Side Request Forgery**

(SSRF) Practical Overview. 2021. Disponível em: < https://www.immuniweb.com/blog/OWASP-server-side-request-forgery-ssrf.html/>. Acesso em: 17 de nov. 2021

IMMUNIWEB. **OWASP Top 10 in 2021: Software and Data Integrity**

Failures Practical Overview. 2021. Disponível em: < https://www.immuniweb.com/blog/OWASP-software-and-data-integrity-failures.html/>. Acesso em: 14 de nov. 2021.

IMMUNIWEB. **OWASP Top 10 in 2021: Vulnerable and Outdated**

Components Practical Overview. 2021. Disponível em: < https://www.immuniweb.com/blog/OWASP-vulnerable-and-outdated-components.html/>. Acesso em: 08 de nov. 2021.

IMMUNIWEB. **Server-Side Request Forgery [CWE-918].** 2018. Disponível em: <https://www.immuniweb.com/vulnerability/ssrf.html/>. Acesso em: 17 de nov. 2021

IVANOVA, Irina. **Equifax ex-CEO: Hacked data wasn't encrypted**. CBS NEWS. 2017. Disponível em <https://www.cbsnews.com/news/equifax-ex-ceo-hacked-data-wasnt-encrypted/>. Acesso em: 01 de nov. 2021.

KASPERSKY. **O que é uma ameaça persistente avançada (APT)?** 2021.Disponível em: <https://www.kaspersky.com.br/resource-center/definitions/advanced-persistent-threats>. Acesso em: 26 de nov. 2021.

KASPERSKY. **Ransomware: definição, prevenção e remoção**. 2021.Disponível em: <https://www.kaspersky.com.br/resource-center/threats/ransomware>. Acesso em: 26 de nov. 2021.

KirstenS. **Cross Site Scripting (XSS).** OWASP. 2021. Disponível em: < https://owasp.org/www-community/attacks/xss/>. Acesso em: 03 de nov. 2021.

KREBS, Brian. **Crooks Grab W-2s from Credit Bureau Equifax**. KrebsonSecurity. 2016. Disponível em: <https://krebsonsecurity.com/2016/05/crooks-grab-w-2s-from-credit-bureau-equifax/>. Acesso em: 27 de out. 2021.

LIMA, Luis Felipe de. **Teste de intrusão para aplicações web: um método com planejamento em inteligência artificial**. Tese (Mestrado em Informática no Programa de PósGraduação em Informática, Setor de Ciências Exatas) - Universidade Federal do Paraná. Curitiba, p. 111. 2020.

MACHADO, Marcel Jacques. **Segurança da Informação: uma Visão Geral sobre as Soluções Adotadas em Ambientes Organizacionais**. Curitiba: UFPR, 2012. Trabalho de Graduação – Bacharelado em Ciência da Computação, Universidade Federal do Paraná, Curitiba, 2012.

MACKIE, Kurt. Microsoft Releases Out-of-Band Security Patches for Exchange Server. Redmond. 2021. Disponível em: < https://redmondmag.com/articles/2021/03/02/exchange-server-zero-day-patches.aspx/>. Acesso em: 18 de nov. 2021

MALENKOVICH, Serge. **O que é um Ataque Man-in-the-Middle?** Kaspersky. 2013. Disponível em: <https://www.kaspersky.com.br/blog/what-is-a-man-in-the-middle-attack/462/>. Acesso em: 26 de nov. 2021.

MESSMER, Ellen. **Heartland: 'Largest Data Breach Ever'**. CSO Online. 2009. Disponível em: <https://www.csoonline.com/article/2123599/heartland---largest-data-breach-ever-.html/>. Acesso em: 05 de nov. 2021.

MICROSOFT. **Supply chain attacks**. 2021. Disponível em: <https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/supply-chain-malware>. Acesso em: 26 de nov. 2021.

MICROSOFT. **Tutorial: Alertas de reconhecimento**. 2021. Disponível em: < https://docs.microsoft.com/pt-br/defender-for-identity/reconnaissance-alerts/>. Acesso em: 10 de nov. 2021.

MITRE. **Common Weakness Enumeration**. 2021. Disponível em: <https://cwe.mitre.org/index.html>. Acesso em: 24 de nov. 2021.

MOZILLA. **Cross-Origin Resource Sharing (CORS)**. 2021. Disponível em: < https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. Acesso em: 24 de nov. 2021.

MOZILLA. **HTTP request methods**. 2021. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>. Acesso em: 24 de nov. 2021.

MXM SISTEMAS. **DevSecOps e DevOps: entenda as diferenças**. 2020. Disponível em: <https://www.mxm.com.br/blog/devsecops-devops-diferenca/>. Acesso em: 26 de nov. 2021.

NAYAK, Amukta. **OWASP Top 10 Deep Dive: Getting a Clear View on Vulnerable and Outdated Components**. Rapid7. 2021. Disponível em: < https://www.rapid7.com/blog/post/2021/11/08/owasp-top-10-deep-dive-getting-a-clear-view-on-vulnerable-and-outdated-components/>. Acesso em: 08 de nov. 2021.

NAZIRIDIS, Nick. **Perguntas frequentes: ataques de rede e problemas de segurança**. SSL.com. 2021. Disponível em < https://www.ssl.com/pt/faqs/ataques-de-rede-e-problemas-de-seguran%C3%A7a/>. Acesso em: 01 de nov. 2021.

NIST. **Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology**. 2012. Disponível em: <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf/>. Acesso em: 16 de nov. 2021.

NIST. **Digital Identity Guidelines Authentication and Lifecycle Management: 5.1.1 Memorized Secrets**. 2019. Disponível em: < https://pages.nist.gov/800-63-3/sp800-63b.html#memsecret/>. Acesso em: 09 de nov. 2021.

NTT. **Global Threat Intelligence Report Executive Guide**. 2021. Disponível em < https://hello.global.ntt/en-us/insights/2021-global-threat-intelligence-report>. Acesso em 14 set. 2021.

OWASP. **A01:2021 – Broken Access Control**. 2021. Disponível em: < https://owasp.org/Top10/A01_2021-Broken_Access_Control/> Acesso em: 27 de out. 2021.

OWASP. **A02:2021 – Cryptographic Failures**. 2021. Disponível em: < https://owasp.org/Top10/A02_2021-Cryptographic_Failures//> Acesso em: 01 de nov. 2021.

OWASP. A03:2021 – Injeção. 2021. Disponível em: < https://owasp.org/Top10/pt_BR/A03_2021-Injection/>. Acesso em: 03 de nov. 2021.

OWASP. **A04:2021 – Insecure Design**. 2021. Disponível em: < https://owasp.org/Top10/A04_2021-Insecure_Design/>. Acesso em: 05 de nov. 2021.

OWASP. **A05:2021 – Security Misconfiguration**. 2021. Disponível em: < https://owasp.org/Top10/A05_2021-Security_Misconfiguration/>. Acesso em: 06 de nov. 2021.

OWASP. **A06:2021 – Vulnerable and Outdated Components**. 2021. Disponível em: < https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/>. Acesso em: 08 de nov. 2021.

OWASP. **A07:2021 – Identification and Authentication Failures**. 2021. Disponível em: < https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/>. Acesso em: 09 de nov. 2021.

OWASP. **A08:2021 – Software and Data Integrity Failures**. 2021. Disponível em: < https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures//>. Acesso em: 13 de nov. 2021.

OWASP. **A09:2021 – Security Logging and Monitoring Failures**. 2021. Disponível em: < https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/>. Acesso em: 15 de nov. 2021.

OWASP. **A10:2021 – Server-Side Request Forgery (SSRF)**. 2021. Disponível em: <https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/>. Acesso em: 16 de nov. 2021.

OWASP. **A9:2017-Using Components with Known Vulnerabilities**. 2017. Disponível em: <https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities/>. Acesso em: 08 de nov. 2021.

OWASP. **About the OWASP Foundation. 2021**. Disponível em: < https://owasp.org/about/>. Acesso em: 24 de out. 2021.

OWASP. **OWASP Top 10 – 2017: The Ten Most Critical Web Application Security Risks**. 2017. Disponível em: < https://wiki.owasp.org/images/0/06/OWASP_Top_10-2017-pt_pt.pdf>. Acesso em: 24 de out. 2021.

OWASP. **OWASP Top Ten. 2021**. Disponível em: <https://owasp.org/www-project-top-ten/>. Acesso em: 24 de out. 2021.

OWASP. **Top Ten Data Driven (partially)**. 2021. Disponível em: < https://www.owasptopten.org/thedata/>. Acesso em: 24 de out. 2021.

OWASP. **Top Ten Survey. 2021**. Disponível em: < https://www.owasptopten.org/thesurvey/>. Acesso em: 24 de out. 2021.

PHISHING. **What Is Phishing?** [2017]. Disponível em: < https://www.phishing.org/what-is-phishing/>. Acesso em: 26 de nov. 2021.aut

PINTO, Juliane Borsato Beckedorff; COSTA, Lucas da Silva; GODOY, Leonardo Buck. **Segurança em Aplicações Web: Um estudo do SQL Injection**. TCC - Faculdade de Tecnologia de Americana. Americana, p. 16. 2019.

PODJARNY, Guy. **Which of the OWASP Top 10 Caused the World's Biggest Data Breaches?** Snyk.io. 2017. Disponível em: <https://snyk.io/blog/owasp-top-10-breaches/>. Acesso em: 27 de out. 2021.

PORTSWIGGER. **Server-side request forgery (SSRF). 2021**. Disponível em: < https://portswigger.net/web-security/ssrf/>. Acesso em: 16 de nov. 2021.

PTES. **Main Page: High Level Organization of the Standard**. 2014. Disponível em: <http://www.pentest-standard.org/index.php/Main_Page>. Acesso em: 20 de out. 2021.

RED HAT. **What is a CVE?** 2020. Disponível em: < https://www.redhat.com/en/topics/security/what-is-cve>. Acesso em: 24 de nov. 2021.

REDHAT. **What is a CI/CD pipeline?** 2019. Disponível em: <https://www.redhat.com/en/topics/devops/what-cicd-pipeline>. Acesso em: 26 de nov. 2021.

RIGUES, Rafael. **Facebook atribui recente vazamento de dados de usuários a "scraping". Olhar Digital**. 2021. Disponível em: <https://olhardigital.com.br/2021/04/07/seguranca/facebook-atribui-recente-vazamento-de-dados-de-usuarios-a-scraping/>. Acesso em: 06 de nov. 2021.

SANCHES, Alan. Metodologias de Testes de Intrusao Pentest. Esecurity. 2019. Disponível em: < https://esecurity.com.br/metodologias-de-testes-de-intrusao-pentest/ >. Acesso em: 20 de out. 2021.

SBARAGLIA, Giorgio. **The main cyber attacks of the first semester of 2021 targeting companies and organizations. Flashstart**. 2021. Disponível em: < https://flashstart.com/the-main-cyber-attacks-of-the-first-semester-of-2021-targeting-companies-and-organisations/>. Acesso em: 19 de nov. 2021

SEGOVIA, Antonio Jose. **How to use penetration testing for ISO 27001 A.12.6.1**. Advisera. 2016. Disponível em: <https://advisera.com/27001academy/blog/2016/01/18/how-to-use-penetration-testing-for-iso-27001-a-12-6-1/ >. Acesso em: 17 de out. 2021.

SIEMBA. **OWASP Top 10: Broken Access Control. 2021**. Disponível em: < https://www.siemba.io/post/owasp-top-10-broken-access-control>. Acesso em: 27 de out. 2021.

SOBERS, Rob. **What is OAuth? Definition and How it Works**. VARONIS. 2018. Disponível em: <https://www.varonis.com/blog/what-is-oauth/>. Acesso em: 25 de nov. 2021.

SOLARWINDS. Solar Winds. 2021. Disponível em: https://www.solarwinds.com/pt/>. Acesso em: 13 de nov. 2021.

SPRINGPEOPLE. **Know What Is An Exchange Server And How It Works**. 2018. Disponível em: < https://www.springpeople.com/blog/what-is-an-exchange-server-and-how-it-works/>. Acesso em: 17 de nov. 2021

TECHMONITOR. **SQL Injection Attacks on the Rise, As Gaming Industry Under Attack from Credential Stuffing**. 2019. Disponível em: <https://techmonitor.ai/techonology/cybersecurity/sql-injection-attacks/>. Acesso em: 03 de nov. 2021.

TIINSIDE. **Um terço das empresas sofreram com ataques por causa de senhas e políticas fracas de segurança, diz Kaspersky**. 2021. Disponível em: < https://tiinside.com.br/16/08/2021/um-terco-das-empresas-sofreram-com-ataques-por-causa-de-senhas-e-politicas-fracas-de-seguranca-diz-kaspersky/>. Acesso em: 10 de nov. 2021.

TURNER, Jeremy. **December 2020 SolarWinds breach: What you need to know**. Coalitioninc. 2020. Disponível em: < https://www.coalitioninc.com/blog/december-2020-solarwinds-breach-what-you-need-to-know/>. Acesso em: 13 de nov. 2021

UCHILL, Joe. **Google pitches security standards for 'critical' open-source projects**. SC Media. 2021. Disponível em: < https://www.scmagazine.com/news/security-news/google-pitches-security-standards-for-critical-open-source-projects/>. Acesso em: 09 de nov. 2021

UNIT 42. **Palo Alto Networks. Threat Assessment: Active Exploitation of Four Zero-Day Vulnerabilities in Microsoft Exchange Server.** 2021. Disponível em: < https://unit42.paloaltonetworks.com/microsoft-exchange-server-vulnerabilities//>. Acesso em: 19 de nov. 2021

WALLARM. **A1: Injection 2017 OWASP**. 2021. Disponível em: <https://www.wallarm.com/what/a1-injection-2017-owasp>. Acesso em: 03 de nov. 2021.

7. AGRADECIMENTOS

Primeiramente a Deus, que assim como em todos momentos desafiadores da vida, foi refúgio e esperança. A toda minha família por todo o apoio, incentivo e base fundamental para todos os meus objetivos de vida. A todos os professores do curso de Ciência da Computação, por todo o conteúdo lecionado, juntamente com todo o conhecimento de voz experiente transmitido, em especial ao prof. Dr. Carlos Eduardo Câmara, pelas aulas e pela orientação para a execução deste trabalho. À meus amigos de sala, pela parceria de todos esses anos e ao Centro Universitário Padre Anchieta, pela estrutura fornecida.

XÔ, DENGUE! UM APLICATIVO DE APOIO AO COMBATE DO MOSQUITO *AEDES AEGYPTIS*

XÔ, DENGUE! A SUPPORT APPLICATION TO COMBAT THE AEDES AEGYPTIS MOSQUITO

Marco Aurélio Fontes VENÂNCIO
marcosym@live.com
Ciência da Computação, Universidade Cruzeiro do Sul

Fernando TAMATAYA
ftamataya@outlook.com
Ciência da Computação, Universidade Cruzeiro do Sul

Lucas Fermiano TEIXEIRA
lucas.nota10@hotmail.com
Ciência da Computação, Universidade Cruzeiro do Sul

Juliano SCHIMIGUEL
schimiguel@gmail.com
Centro Universitário Anchieta, e Universidade Cruzeiro do Sul

Carlos Adriano MARTINS
ead.adriano@gmail.com
Universidade Cidade de São Paulo, e Faculdade Fernão Dias

Resumo

Este trabalho tem como objetivo descrever um aplicativo para dispositivos móveis, com apoio ao ambiente WEB, visando ao combate do mosquito Aedes Aegypti e suas doenças. O aplicativo foi construído utilizando-se do framework Android Java, com funcionalidade compatível com dispositivos móveis e também em ambiente Web, Java e tecnologias diversas, tendo múltiplas funções. Como resultado preliminar, durante o seu desenvolvimento, o aplicativo foi avaliado e caracterizado como um aplicativo com tema relevante e de fácil uso, constituindo iniciativa adequada para apoio à comunidade. Espera-se atingir o objetivo de conscientizar a população acerca dos riscos do mosquito, bem como utilizar a tecnologia móvel para incrementar o trabalho de ajuda coletiva.

Palavras-Chave

Aplicativo, Android, Dispositivos Móveis, Aedes Aegypti, Conscientização.

Abstract

This study aims to describe a software to mobile application, with the support to the WEB environment, focusing on the combat against the mosquito Aedes Aegypti and its diseases. The application was developed on an Android Java framework, supporting mobile devices and Web environments, utilizing Java and multiple technologies, having multiple functions. As preliminary results, during its development, the application was evaluated and characterized as a relevant application and easy to use, an adequate idea to help the community. It is expected to achieve the goal of raising awareness population about of the risks of mosquito and its diseases, and how to make use of mobile technology to increase the collective work of help.

Keywords

Software Application, Android, Aedes Aegypti, Awareness.

INTRODUÇÃO

O Zika Vírus, o Flavivírus, é uma doença que possui sintomas similares à dengue comum, sendo transmitida pelo mosquito Aedes Aegypti, e que foi primeiramente registrada em 1947, em um caso isolado na floresta Zika, localizada na Uganda, continente africano. Até 2013, a doença não teve muito destaque, permanecendo restrita apenas a locais específicos da África e Ásia. Seu primeiro registro fora desses continentes ocorreu na Ilha da Páscoa, há mais de 3.000 km. A doença entraria em foco no Brasil em meados de 2015, com denúncias de um surto em Camaçari, Bahia, de uma doença que se caracterizava por sintomas de febre, dores musculares, dores nas articulações e conjuntivite (CAMPOS et al., 2015).

Em 26 de março de 2015, amostras foram coletadas de 24 pacientes considerados infectados, no hospital de Santa Helena e então encaminhados à Universidade Federal da Bahia. Após estudos, foram constatados que 10 pacientes possuíam casos confirmados do Zika, dando início a um surto da doença que ocorreria no país. Apesar dos sintomas do Zika Vírus, a doença em si não parecia apresentar riscos tão grandes para a população, pois o real risco seria identificado a partir de estudos e observações de pacientes infectados. Percebeu-se que o número de ocorrências da Microcefalia (condição neurológica rara em que a cabeça e o cérebro do recém-nascido são significativamente menores do que outras crianças da mesma idade com desenvolvimento típico) teve mais casos ocorridos após o surto do vírus da Zika, com mais de 8.000 casos registrados em um curto período de apenas 9 meses.

Relacionado à esta doença, o Centro de Controle de Doenças dos Estados Unidos (CDC) afirmou que a transmissão do Zika Virus pode, inclusive, se dar por meio de relações sexuais, alertando para a necessidade de conscientização da população (EXAME, 2016).

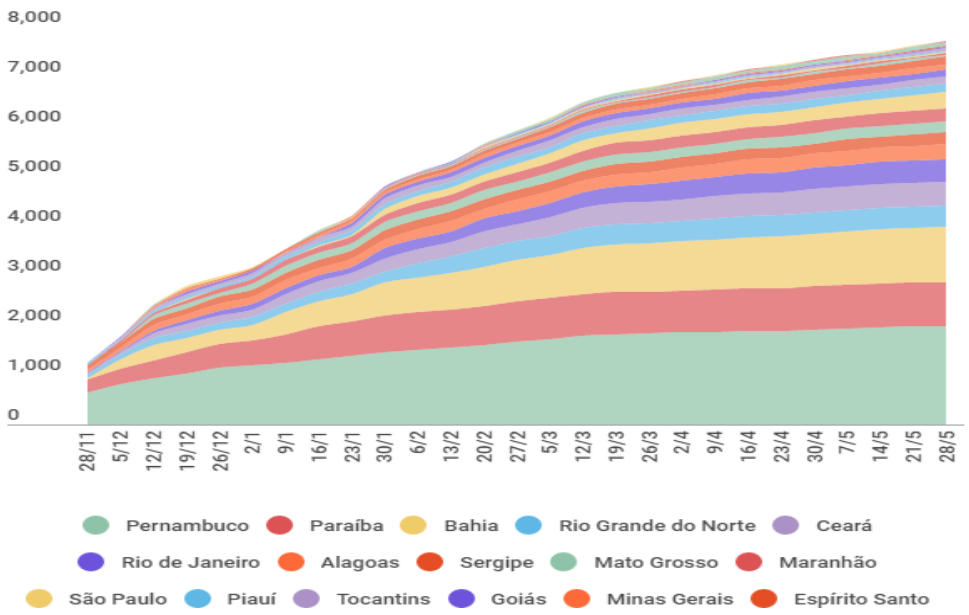

Figura 1. Participação de cada estado brasileiro nos casos de microcefalia entre 2015 e 2016.

Fonte: www.exame.abril.com.br

Ao mesmo tempo em que acontecia o surto da doença, a popularidade dos smartphones ou celulares inteligentes continuava o seu crescimento no país, com mais de 20 milhões de usuários dos aparelhos. Em função de suas características de fácil transporte e mobilidade, além de rápida acessibilidade, diversos projetos de aplicação estão em estudo em diferentes áreas, como na área da saúde, devido às possibilidades que a rede móvel oferece, como acesso a controle de dados, lembretes e diagnósticos. Tal tecnologia pode incrementar a agilidade e precisão dos profissionais em seu trabalho, independentemente de sua área. Neste contexto, este trabalho tem como objetivo principal apresentar o desenvolvimento de um sistema para auxílio no controle da doença Zica, através do suporte da tecnologia de rede móvel, visando oferecer um recurso à comunidade. Além disso, o trabalho se propõe a identificar o estado atual da doença no país, discutir brevemente o impacto da tecnologia móvel e alguns estudos já realizados sobre tecnologia móvel na área da saúde e áreas diversas.

Assim, o aplicativo aqui proposto constitui ferramenta de apoio ao processo de conscientização e educação da população na área de saúde, envolvendo um tema altamente relevante e atual que é o Aedes Aegypti e as doenças a ele associadas, as quais podem causar graves consequências como sequelas e até mesmo óbitos.

Para a abordagem do tema em foco buscamos primeiramente caracterizar a sua relevância na atualidade, visto que o ressurgimento do mosquito Aedes Aegypti e os recentes casos de Zika Vírus e Microcefalia no Brasil, passaram a frequentar cada vez mais os noticiários divulgados pela mídia em geral. Números e estatísticas alarmantes estavam sendo constantemente publicadas, conforme mostra a Figura 2, e a população começou a entender o quão grave era a situação que nos encontrávamos, fato que demanda iniciativas que proporcionem o enfrentamento e a minimização dos problemas relacionados a esta grave doença.

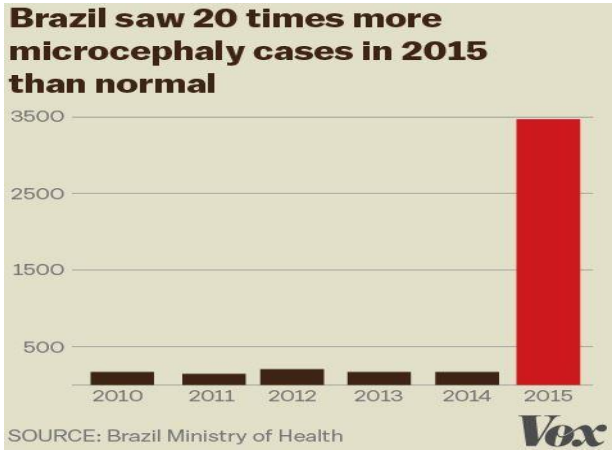

Figura 2. Número de casos de Microcefalia no Brasil entre 2010 e 2015.

Complementarmente, este trabalho enfatiza a evolução da tecnologia, com foco na grande utilização de dispositivos móveis por parte da população. Em diversos lugares se presenciam pessoas com smartphones ou tablets e percebemos que seria adequado aproveitar essa tendência em nosso estudo e, assim, buscamos identificar quais as preferências dos usuários para funções dos dispositivos. O gráfico da Figura 3 mostra o crescimento no uso de aplicativos para dispositivos móveis.

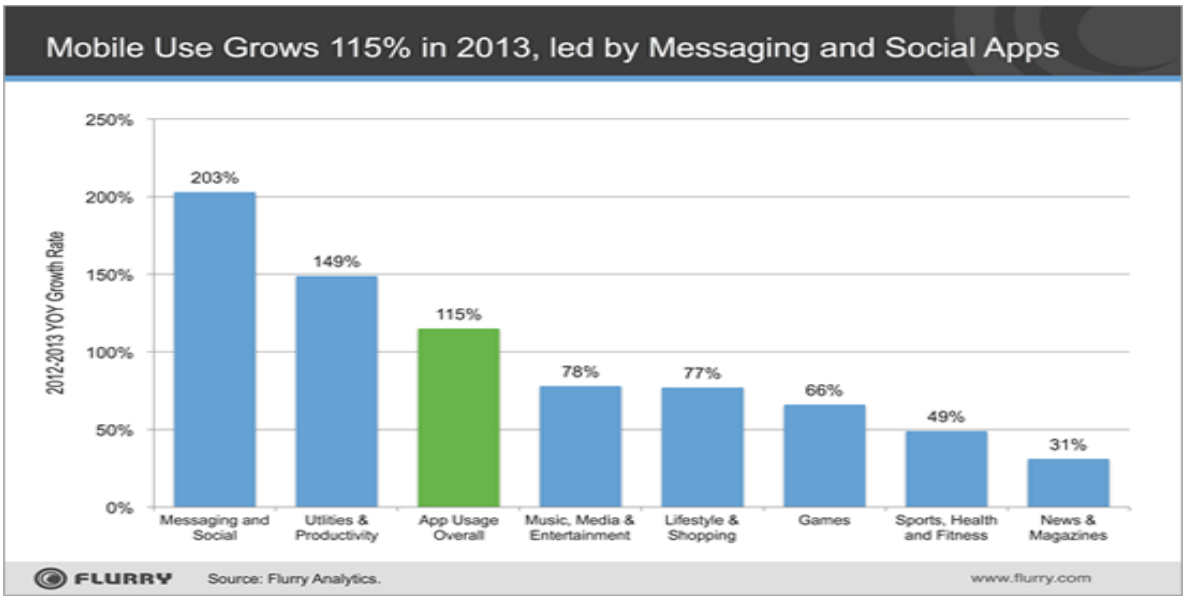

Figura 3. Crescimento do uso de aplicativos para dispositivos móveis.

Levando em consideração que os dispositivos apresentam em seus atalhos diversas funções e opções, e ainda possibilitam a opção de serem estendidas e melhoradas, os usuários recorrem com frequência a diferentes aplicativos desenvolvidos para facilitar algumas atividades de seu cotidiano. Podemos afirmar que tal fato não é surpresa, pois além de existirem milhares de aplicativos disponíveis, tais Apps variam em suas finalidades, fornecendo desde puro entretenimento em jogos, até aplicativos para aprendizagem em áreas diversas, auxílio à mobilidade, entre outros.

REFERENCIAL TEÓRICO

Visando desenvolver um mecanismo de combate ao mosquito, analisamos quais as metodologias e procedimentos são normalmente usados neste combate. O monitoramento do mosquito já estava em ação desde 2007, apresentando diferentes metodologias e tratativas para a identificação do risco que o mosquito e suas doenças trariam ao país. Em seu artigo, Braga e Valle (2007a) acreditam que certas metodologias adotadas eram inviáveis e não conseguiam obter dados suficientes para apresentar o risco do mosquito.

A primeira opção foi a coleta de larvas, sendo recipientes e depósitos de água vistoriados, com atenção especial a locais estratégicos para proliferação do mosquito (ferros-velhos, borracharias, cemitérios, entre outros locais que sejam favoráveis ao espécime). Entretanto, essa metodologia não se mostrou eficaz para medir a abundância do mosquito adulto, sendo ineficiente para estimar o risco de transmissão na área.

Na segunda metodologia, a coleta de mosquitos adultos foi referenciada. Mesmo oferecendo dados suficientes, Braga e Valle (2007a) defendem ser esta estratégia inviável, pois os mosquitos adultos geralmente pousam e se alocam em locais inacessíveis, sendo impossível determinar a quantidade total de mosquitos, pois uma coleta seria apenas uma estimativa do seu total. O segundo ponto observado seria a relação entre humanos infectados e o número de mosquitos, outro método que se mostrou falho em determinar o risco. Acerca das estratégias de controle, as autoras destacam que:

> *O monitoramento da susceptibilidade das populações de Aedes Aegypti a inseticidas, em diferentes regiões do país, contribuirá para a definição de estratégias racionais de controle fundamentadas no perfil da resistência do vetor e dos mecanismos envolvidos no nível local. (BRAGA e VALLE, 2007a, p. 298).*

Equipes de diferentes estados e municípios brasileiros realizavam diversos experimentos com larvas e mosquitos empregando inseticidas diferentes, como fenitrotion, malation e temephos, para determinar a resistência do espécime frente a estes produtos.

Em 1999, o laboratório da Fiocruz foi responsável por realizar testes em exemplares provenientes do Rio de Janeiro e do Espírito Santo em 10 de seus municípios. O resultado foi a resistência do mosquito ao temephos em todos os municípios do Rio de Janeiro, e em um dos três municípios do Espírito Santo. Ainda em 2001, novas avaliações realizadas identificaram a resistência ao produto temephos em larvas, e alterações de susceptibilidade em mosquitos adultos para os produtos organofosforados fenitrotion e malation, sinalizando possíveis alternativas de controle (BRAGA e VALLE, 2007a).

Durante o período de determinação de alternativas de controle, o primeiro passo da FUNASA foi substituir inseticidas organofosforados por piretróides e então encontrar uma alternativa para o temephos. Um dos produtos recomendados para o controle do mosquito foi o Methoprene, tendo uma característica única de controlar o surgimento de mosquitos adultos devido à sua composição química. Em estudos de laboratório, foi definido que o produto não apresentava uma taxa de mortalidade ao espécime, mas inibia o crescimento das

larvas, além de que todos os exemplares estudados mostraram-se susceptíveis ao methoprene (BRAGA e VALLE, 2007a).

Com estes primeiros resultados, é possível indicar que o methoprene seja uma alternativa de controle para o temephos, desde que acompanhada por avaliação de campo, caso a tratativa sofra algum tipo de alteração. Nesta perspectiva, Braga e Valle (2007b) cita o pyriproxifen, que possui os mesmos efeitos do methoprene, com a adição de serem utilizáveis em água potável, não causando complicações de contaminação de risco. A autora reforça que, mesmo com as alternativas apresentadas, o combate ainda não é suficiente, e que deveria ser estimulado com investimentos em laboratórios com infraestrutura adequada e capacitação técnica e apoio às redes e equipes existentes.

Buscando associar a tecnologia de rede móvel com este importante tema atual, procuramos identificar algumas contribuições da tecnologia em outras áreas e também na área de saúde. Neste sentido, o trabalho de Sena, Oliveira e Carvalho (2014) teve como foco uma aplicação da tecnologia para auxiliar no ensino da disciplina de Matemática, mas não se limitando a tal assunto, apresentando aplicativos existentes e identificando a necessidade de inovações no ensino da Matemática, sendo esta necessidade justamente uma motivação para a realização do estudo desenvolvido.

Por sua vez, Tibes, Dias e Zem-Mascarenhas (2014) oferecem em seu artigo informações sobre o surgimento e crescimento da popularidade dos dispositivos móveis, citando suas características, vantagens e razões de utilização, enquanto relaciona este assunto com a área de saúde. Para reforçar seu estudo, foram realizadas pesquisas em bases de dados de artigos conhecidas, como SCIELO, Biblioteca Virtual, Google Scholar, entre outras, usando critérios como: serem on-line, da língua portuguesa e terem sido desenvolvidos por pesquisadores brasileiros entre 2006 e 2013. Após análise, os autores concluíram que o uso da tecnologia móvel na área de saúde era algo recente e com pouca utilização até 2013, e que devido às suas características seria oportuno o crescimento da sua utilização, citando como a tecnologia móvel poderia revolucionar a área e como as pesquisas poderiam ser realizadas.

Neste cenário, podemos destacar o mobile learning (m-learning), ou aprendizagem móvel, que se caracteriza pelo uso de equipamentos móveis e portáteis, em um cenário de computação pervasiva, baseado na mobilidade do usuário, pela conexão ubíqua e pela independência de dispositivo e ambiente computacional (BARCELOS et al., 2009).

Os smartphones têm potencial para oferecer vários níveis de envolvimento do usuário, mas a maioria das aplicações educacionais para mobile ainda oferecem o conteúdo da mesma forma que apresentavam em um computador tradicional, de modo que se constata a necessidade de criação de novas metodologias e paradigmas para o desenvolvimento de aplicações para celulares (BARCELOS et al., 2009).

Para Silva et al. (2011), o mobile learning é uma forma de oferecer um ensino capaz de permitir que alunos e professores possam acessar as vantagens oferecidas pelos recursos das tecnologias móveis, destacando-se a

possibilidade de acesso, a visualização e a disponibilização de conteúdo, independente da hora e em qualquer local.

Constata-se, assim, que a utilização dos recursos tecnológicos, particularmente dos dispositivos móveis, tende a oferecer diversas contribuições para os usuários em diferentes áreas.

Identificação de Sistemas Correlatos

Primeiramente, realizamos pesquisas para localizar aplicativos que pudessem auxiliar no entendimento das funcionalidades em geral e assim identificarmos funções que poderiam constituir diferenciais em nosso aplicativo.

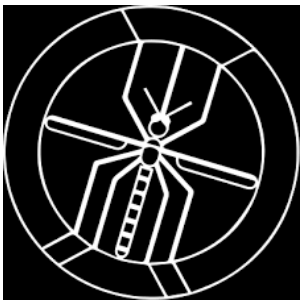

Figura 4. Logo do aplicativo Vigilantus.

O Vigilantus, cujo logo é mostrado na figura 4, é um software desenvolvido pela empresa chamada 'Perspectiva', disponível exclusivamente para o ambiente WEB, sendo pioneiro na utilização da tecnologia para apoio ao combate do mosquito. Este aplicativo ganhou reconhecimento do SEBRAE, FAPESC e o Ministério da Saúde, além de contar com o apoio de outros projetos, como 'Visão de Sucesso', destinado a pequenos empreendedores.

Em sua estruturação, a geração de relatórios é sua principal característica, por estar em conformidade com o padrão PNCD (Plano Nacional de Combate à Dengue) e tais relatórios serem encaminhados diretamente à área de saúde do governo. Além de tal função, também conta com gráficos, mapas, planilhas e indicadores. Utilizando a função de registro de pontos infectados, o software disponibiliza consultas a tais pontos, oferece uma área de infecção aproximada para o usuário e fornece orientações para agir em situações diversas em que o mosquito esteja presente.

Os registros feitos por usuários são avaliados e estudados pela equipe e encaminhados ao órgão de saúde do governo, retornando um protocolo para o usuário utilizar de referência para consulta.

Figura 5. Logo do aplicativo Sem Dengue.

O ‘Sem Dengue’ (Figura 5), é um aplicativo desenvolvido pela Colab e que se encontra disponível exclusivamente para dispositivos móveis, sendo o aplicativo de maior utilização e votação positiva apontada na PlayStore no ano de 2016. Em parceria com diversas áreas, o aplicativo teve grande sucesso, devido às suas funcionalidades e interface simples. Na página inicial do aplicativo, é apresentada uma visão geral de todos os focos reportados, sendo eles confirmados e dicas para agir contra o mosquito. Em ‘Mapa’, é oferecido ao usuário uma tela com o mapa do país, onde podem ser realizados registros de novos casos, ou consultas de casos registrados anteriormente. O usuário pode também consultar seus registros diretamente com informações mais detalhadas em outra tela, além de receber notificações sobre tais registros. Por fim, o aplicativo apresenta suas parcerias e áreas de trabalho.

Após a consulta e utilização dos aplicativos apresentados, como diferencial buscamos inserir em nosso aplicativo as opções de: notícias em tempo real, fornecidas pela revista Science Maganize Org, e notificações para lembrete de tratativas contra o mosquito, além de duas formas de acesso, ou seja, através da Web e pelo dispositivo móvel (celular, smartphone). Além disso, nosso aplicativo se diferencia dos demais existentes por fornecer alertas aos usuários, diretamente no próprio mapa, quando a região apresentar risco para a manifestação da doença.

ASPECTOS METODOLÓGICOS DA PESQUISA

Este trabalho envolve inicialmente uma pesquisa bibliográfica, além de se enquadrar como pesquisa de cunho laboratorial. A pesquisa laboratorial ocorre em situações controladas, onde a maioria das pesquisas se encontra em locais fechados (laboratórios), ao ar livre ou em ambientes artificiais. As pesquisas laboratoriais necessitam de um ambiente possível de controle, definido previamente de acordo com o estudo a ser desenvolvido (PORTAL EDUCAÇÃO, 2018).

Consideramos a parte WEB da aplicação um recurso capaz de oferecer apoio para o desenvolvimento do aplicativo e sua futura extensão. O ambiente WEB foi incluído, pois mesmo que grande parte da população opte pela utilização da rede móvel, a tecnologia ainda abrange a parte de desktops e há pessoas que optam pelo seu uso, seja por motivos de segurança ou mais conforto e outras funcionalidades.

Por fim, definimos as ferramentas necessárias para a realização do trabalho e optamos pela linguagem Java, tecnologias HTML e CSS por fornecerem opções de fácil acesso para o usuário e para o programador; o recurso RSS da linguagem XML possibilita gerar notícias e atualizações em tempo real; o Bootstrap para ajustamento de resolução para dispositivos de tamanhos variados; o Push Notifications para criação de alarmes e notificações de lembrete aos usuários; o API Locations para geração do mapa e localização da denúncia feita; e o MySQL como banco de dados e informações.

DESENVOLVIMENTO DO APLICATIVO/RESULTADOS

Inicialmente planejamos quais funções e opções seriam necessárias para o aplicativo. Definimos utilizar um mapa para localização, notícias que são encaminhadas em tempo real, testes de infecção, consulta de denúncias e registros já realizados, sintomas comuns das doenças, procedimentos básicos de tratamento, informações sobre os desenvolvedores e o apoio de acesso WEB.

Após realizar uma pesquisa em artigos publicados sobre o mosquito e suas origens, tratativas e metodologias já existentes, além do impacto da tecnologia em trabalhos publicados no Scielo, INPE, SBIE e JHI, estabelecemos quais seriam as opções inovadoras para trabalhos neste contexto.

Estruturação do Aplicativo

Para o desenvolvimento do aplicativo, cuja tela inicial é mostrada na figura 6, utilizamos as linguagens HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) e JavaScript.

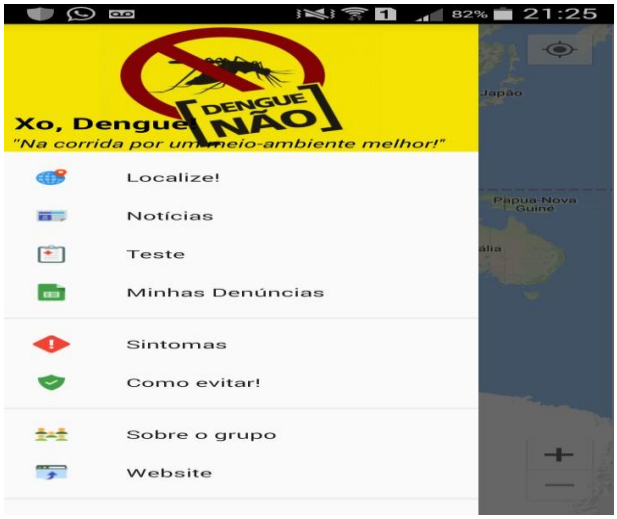

Figura 6. Visão do Menu inicial em Android (Versão 4.4.4).

Após a realização de testes em diversos aparelhos, foi constatado que o aplicativo é compatível com o sistema Android, com versão superior a 4.0 Ice Cream Sandwich, e com necessidade da utilização da internet móvel para acompanhamento de notícias. No caso, fizemos a utilização do Framework do Android Java.

Optou-se por utilizar o recurso Google Maps API, oferecido pela empresa Google, por ser compatível com os sistemas Android, iOS, WEB e WEB services, além de ser de livre acesso e oferecer diversas alternativas para tais sistemas. Para acesso às notícias, foi empregado um recurso da XML (eXtensive Markup Language), o RSS, que permitiu desenvolver a funcionalidade de atualizar-se em tempo real, devido à sua característica de recebimento de dados atualizados.

As opções de Teste, Sintomas, Como Evitar e Sobre o Grupo foram desenvolvidas utilizando comandos padrões da linguagem Java, servindo de lembretes e consultas para os usuários.

No caso das notificações, enviadas aos usuários que possuem o aplicativo instalado, lembretes de tratativas simples são encaminhados através de Push Notifications, uma tecnologia que encaminha notificações sem nenhuma ação por parte do usuário. O banco de dados para o aplicativo móvel utiliza tecnologia SQLite, consequentemente não disponibilizando os dados para o usuário. Futuramente este recurso será convertido para o MySQL. O ambiente WEB utiliza a tecnologia MySQL para seu Banco de Dados. Por fim, destaca-se que o ambiente WEB foi desenvolvido através dos recursos da linguagem Java, HTML e PHP, com apoio ao banco de dados através do MySQL.

Funcionalidades do Aplicativo

As opções de funções do aplicativo são explanadas a seguir.

- **Localize** - está diretamente ligada ao mapa, quando tal opção é selecionada, o usuário é encaminhado à tela inicial do mapa, onde poderá realizar registros de infecção ou suspeitas através do toque, com adição de comentários através do teclado.
- **Notícias** - fornece notícias atualizadas em tempo real, devido à sua estruturação RSS, acompanhando o conteúdo fornecido pela Science Magazine Org.
- **Teste, Sintomas e Como Evitar** - foram opções criadas a partir da coleta de informações de Órgãos de Saúde e Ciência, que estudaram a doença e o mosquito, fornecendo os devidos procedimentos a partir de estudos, artigos e notícias.
- **Minhas Denúncias** - permite ao usuário consultar seus registros feitos anteriormente, tendo a possibilidade de alteração ou exclusão, caso seja necessário.
- **Sobre o Grupo** - oferece informações dos desenvolvedores do trabalho.
- **Website** - encaminhará o usuário ao ambiente Web, localizado no endereço xodengue.esy.es

O Diagrama de Navegação do aplicativo é fornecido na figura 7 mostrada a seguir, enquanto o Diagrama de Sistema é apresentado na sequência, na figura 8.

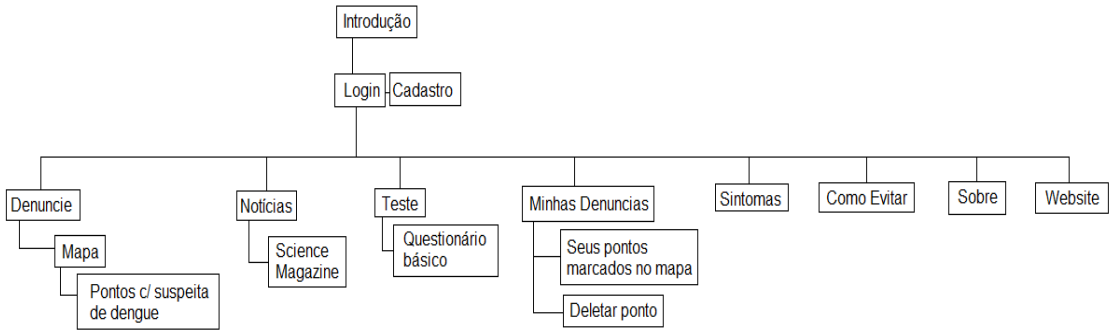

Figura 7. Diagrama para o aplicativo.

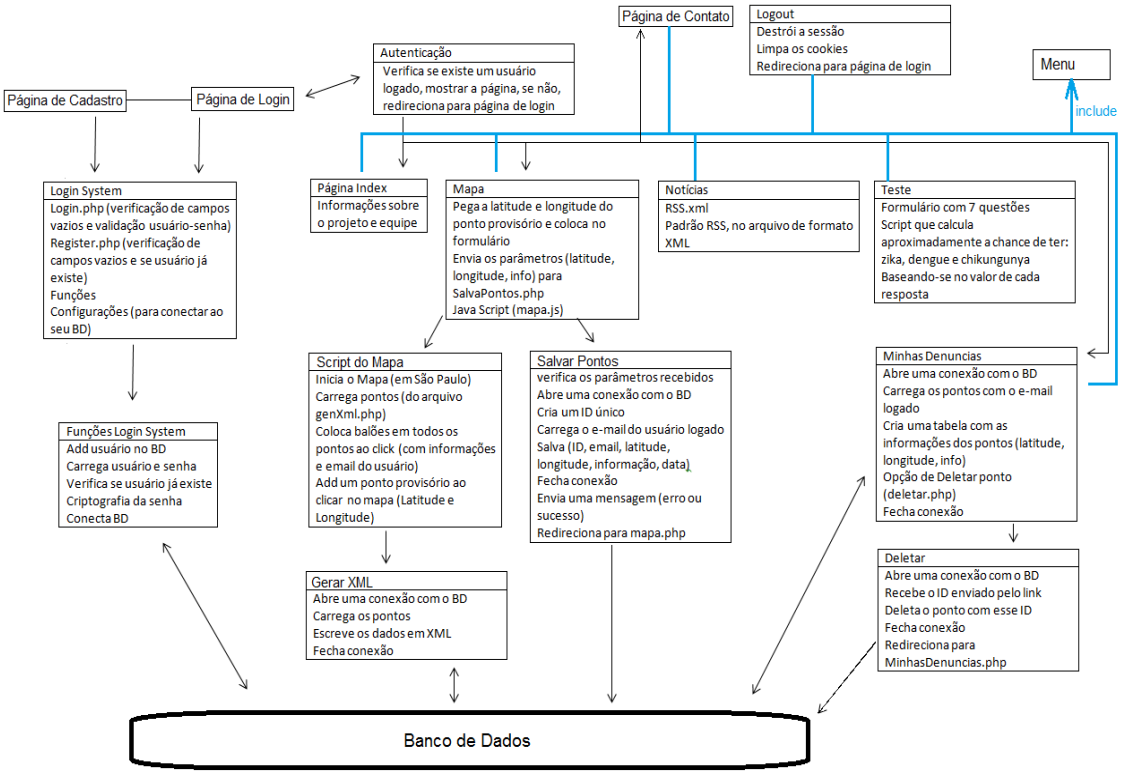

Figura 8. Diagrama de Sistema para o aplicativo.

Identificação e solução de conflitos e falhas

Durante o desenvolvimento do aplicativo, recebemos o feedback de alguns potenciais usuários, onde eles questionaram sobre como poderíamos contornar falhas específicas que não fossem decorrentes do sistema, mas sim pelo próprio uso do aplicativo. Como exemplo, cabe mencionar o que ocorreria caso um usuário tentasse, de propósito, marcar um mesmo local várias vezes a fim de sobrecarregar os dados na aplicação. Pensando neste caso específico, consideramos inserir futuramente no aplicativo um limite de marcações por local para cada usuário, com uma quantidade ainda a ser definida, para evitar problemas decorrentes desta situação. Também consideramos realizar um monitoramento das marcações em datas distantes, para validar se as marcações estão sendo feitas corretamente e coerentemente.

Serão consideradas, ainda, outras possíveis falhas que podem ser encontradas no uso do aplicativo pelo usuário, como a realização de testes de sintomas em branco ou marcações em branco, alternativas que estão sendo pensadas e que serão tratadas com cautela, conforme o feedback recebido dos usuários do aplicativo.

CONSIDERAÇÕES FINAIS

Os estudos realizados possibilitaram perceber que a tecnologia móvel tem grande potencial para ser utilizada em diversas áreas, devido à sua mobilidade e atraente usabilidade, aspectos que favorecem a agilidade e precisão deste recurso. Entendendo isso, analisamos algumas aplicações da tecnologia na área da saúde e alguns trabalhos focados em outras áreas, como a Educação.

Simultaneamente, analisamos dados e estatísticas sobre a predominância do mosquito Aedes Aegypti e doenças a ele relacionadas em nosso país no período de 2015-2016, visando incrementar o desenvolvimento do aplicativo e confirmar os graves riscos de saúde gerados pelo mesmo. Entendemos que as funcionalidades do aplicativo se mostraram adequadas, de modo que o primeiro objetivo do trabalho foi atingido, fornecendo informações adicionais, relevantes e novas para os usuários acerca do assunto.

Esperamos que futuramente seja possível oferecer o projeto do aplicativo aqui descrito para Órgãos de Saúde locais, de modo a alcançarmos o objetivo principal do mesmo, ou seja, contribuir efetivamente para a minimização dos problemas de saúde provocados pelo mosquito Aedes Aegypti, combatendo a sua proliferação.

O aperfeiçoamento do aplicativo será buscado com a implantação de melhorias, de acordo com as necessidades observadas e sugestões colhidas, sendo focadas inicialmente as funções já existentes no aplicativo, além de melhoria da interface e do seu desempenho. Cabe ressaltar ainda que será dada atenção ao ambiente WEB, criando-se novas funções que serão adicionadas, além do oferecimento de suporte para sistemas operacionais mais antigos.

REFERÊNCIAS BIBLIOGRÁFICAS

BARCELOS, R. J. S.; TAROUCO, L.; BERCH, M. *O uso de mobile learning no ensino de algoritmos*. RENOTE - Revista Novas Tecnologias na Educação, v. 7, n. 2, p. 327–337, 2009.

BRAGA, I. A.; VALLE, D. *Aedes Aegypti: vigilância, monitoramento da resistência e alternativas de controle no Brasil*, Epidemiologia e Serviços de Saúde, Brasília, v. 16, n. 4, p. 295-302, 2007a.

BRAGA, I. A.; VALLE, D. *Aedes Aegypti: histórico de controle no Brasil, Epidemiologia e Serviços de Saúde*, Brasília, v. 16, n. 2, p. 113-118, 2007b.

CAMPOS, G. S.; BANDEIRA, A. C.; SARDI, S. I. *Zika Virus Outbreak*, Bahia, Brazil, Emerging Infectious Diseases, v. 21, n. 10, p. 1885-6, 2015.

EXAME. Revista Exame, 2016. Disponível em: http://exame.abril.com.br/mundo/eua-confirmam-caso-de-zika-por-transmissao-sexual-no-pais/. Último acesso: 10/02/2018.

Portal Educação, 2018. *Metodologia Científica: Tipos de pesquisa*. Portal Educação [on line]. Disponível em: https://www.portaleducacao.com.br/conteudo/artigos/pedagogia/metodologia-cientifica-tipos-de-esquisa/50264. Último acesso: 10/02/2018.

SENA, D. M.; OLIVEIRA, E. H. T.; CARVALHO, L. S. G. *Aplicativos móveis para o aprendizado de Matemática*, In: Simpósio Brasileiro de Informática na Educação (SBIE), XXV, v. 25, n. 1, p. 174, Anais ..., Dourados – MS, 2016.

SILVA, L. C. N. DA; MENDES NETO, F. M.; JÁCOME JR., L. *MobiLE: Um ambiente Multiagente de Aprendizagem Móvel para Apoiar a Recomendação Sensível ao Contexto de Objetos de Aprendizagem*. In: XXII SBIE - XVII WIE, Anais ..., Aracaju - SE, 2011.

TIBES, C. M. S.; DIAS, J. D.; ZEM-MASCARENHAS, S. H. *Aplicativos móveis desenvolvidos para a área de saúde no Brasil: revisão integrativa da literatura*, Revista Mineira de Enfermagem, Belo Horizonte – MG, v. 18, n. 2, p. 471-478, 2014.