**CENTRO UNIVERSITÁRIO UNIMETROCAMP WYDEN**

Leonardo **BEVILACQUA**Marcelo **RODRIGUES**

Marcio **MACEDO**

Thiago **OLIVEIRA**

APLICAÇÃO DE PRÉ AVALIAÇÃO E INDICAÇÃO PARA VAGAS DO MERCADO DE TRABALHO

CAMPINAS  
2019

Leonardo BEVILACQUA  
Marcelo RODRIGUES  
Marcio MACEDO  
Thiago OLIVEIRA

APLICAÇÃO DE PRÉ AVALIAÇÃO E INDICAÇÃO PARA VAGAS DO MERCADO DE TRABALHO

Projeto de Conclusão de Curso, apresentado ao Centro Universitário Unimetrocamp Wyden.

Orientador: Prof.MSc João Ronaldo Del Ducca Cunha

CAMPINAS  
2019

Leonardo BEVILACQUA  
Marcelo RODRIGUES  
Marcio MACEDO  
Thiago OLIVEIRA

APLICAÇÃO DE PRÉ AVALIAÇÃO E INDICAÇÃO PARA VAGAS DO MERCADO DE TRABALHO

Projeto de Conclusão de Curso, apresentado ao Centro Universitário Unimetrocamp Wyden.

Aprovado em \_\_/\_\_/\_\_\_\_

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Prof. XXXXXXXXXX  
Faculdade Metrocamp

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Prof.MSc João Ronaldo Del Ducca  
Faculdade Metrocamp

RESUMO

Atualmente ao realizar contratações, é exigido uma demanda de tempo e filtragem até encontrar um candidato que se encaixe em uma determinada vaga. Para ajudar neste processo, existem os *headhunters*, profissionais qualificados para realizar uma seleção de candidatos. Entretanto, o trabalho de um *headhunter* também não é simples, pois é necessário a utilização de uma metodologia para realizar a busca dos candidatos. Alguns *headhunter* podem optar em agir de forma mais generalizada, realizando apenas entrevistas prévias, para dessa forma já possuírem indicação quando houver vagas disponíveis. Essa tática acaba sendo menos precisa para realizar indicações, pois não existe uma a garantia que os candidatos atendem às especificações da empresa. A aplicação tem a intenção de ser um facilitador para candidatos e vagas, permitindo com que *headhunters* encontrem e avaliem candidatos facilmente e indiquem para o Departamento de Recursos Humanos (RH) as melhores opções. Com o sistema, candidatos poderão ser pré avaliados e saber se estão com as habilidades necessárias para as vagas atuais do mercado de trabalho.

**Palavras-chave:** HeadHunter; Vagas; Contratação; Departamento de Recursos Humanos; Recrutamento.

**ABSTRACT**

Nowadays, when hiring, a demand for time and filtering is required until a candidate that fits into a certain vacancy job is found. To assist in this process, there are headhunters, qualified professionals to carry out a selection of candidates. However, the work of a headhunter isn't simple, as it is necessary to use a methodology to perform the search of the candidates. Some headhunters may choose to act more generally, performing only previous interviews, so that they already have an indication when there are vacant job available. This tactic turns out to be less accurate to give indications as there is no guarantee that the candidates meet the specifications of the company. The application is intended to be a facilitator for candidates and vacancies, allowing headhunters to find and evaluate candidates easily and indicate to human Resources (HR) the best options. With the system, candidates can be pre-evaluated and know if they have the skills they need for current job market positions.

**Keywords:** HeadHunter; Vacancy job; Hiring; human Resources; Recruitment.

**LISTA DE ILUSTRAÇÕES**

Nenhuma entrada de índice de ilustrações foi encontrada.

**LISTA DE TABELAS**

Nenhuma entrada de índice de ilustrações foi encontrada.

**LISTA DE ABREVIATURAS E SIGLAS**

**SUMÁRIO**

[**Introdução**](#_3znysh7) **[10](#_3znysh7)**

[Contexto e Problematização](#_2et92p0) [10](#_2et92p0)

[Objetivos](#_tyjcwt) [10](#_tyjcwt)

[Fundamentação Teórica](#_3dy6vkm) [10](#_3dy6vkm)

[*Descrição*](#_1t3h5sf) *[11](#_1t3h5sf)*

[*Justificativa de Uso da Tecnologia*](#_4d34og8) *[11](#_4d34og8)*

[**Bibliografia**](#_3rdcrjn) **13**

# Introdução

Neste capítulo será abordado sobre o que se trata o projeto, quais tecnologias serão utilizadas para conseguir entregá-lo e qual é a necessidade da criação deste produto.

## Contexto e Problematização

O mercado de trabalho está constantemente necessitando de novos candidatos, mas essa interação entre mercado e candidatos nem sempre ocorre de forma simples e transparente. Os candidatos podem se encontrar com dúvidas sobre suas próprias capacidades, questionando se irão atender as expectativas do mercado sem saber quais habilidades são realmente exigidas ou se o que eles têm para oferecer é o suficiente para começar. O resultado disso é a perda de candidatos que poderiam estar ingressando em um novo emprego.

*“Embora tenhamos o prazer de aceitar a missão de procurar um CIO, por exemplo, podemos não ter ideia sobre o que faz um bom CIO, ou se o cargo vai melhorar sua carreira. Em outras palavras, somos generalistas em um mundo de especialistas.”(Stegemann; Jorg;* epocanegocios.globo.com; *2013)*

Departamentos de recursos humanos precisam cada vez mais elaborar novas políticas inovadoras para conseguir encontrar os melhores candidatos em um mercado de matéria prima escassa, milhares de candidatos precisam ser processados rapidamente, para ajudar nesse processo a maioria das empresas criam incentivos internos para que funcionários se tornem “Headhunters” e indique amigos para empresa. Esse mecanismo ajuda as empresas a encontrarem novos funcionários, porém somente uma parte da demanda é fornecida, novos meios necessitam ser criados para que pessoas qualificadas consigam indicar novos funcionários as empresas.

Um ponto fraco em se contratar headhunters é que muitas vezes eles não são especialistas nas vagas que estão buscando sendo assim não possui conhecimento para escolher o melhor candidato

## Objetivos

Com base sobre o problema descrito acima, o objetivo é criar uma aplicação que possa ser usada como uma ponte entre candidatos e vagas do mercado de trabalho. Tendo como função primária, permitir que profissionais experientes atuem como *headhunters* para avaliar e indicar os candidatos para vagas e auxiliá-los em suas necessidades. Dentre os principais objetivos estão:

* Indicação de vagas para o candidato.
* Pré avaliação para a vaga escolhida.
* Pré seleção de candidatos, poupando o tempo da empresa.
* Escolha de profissionais experientes para avaliar candidatos.
* Permitir que empresas encontrem profissionais com maior facilidade

## Fundamentação Teórica

O sistema vai possuir serviços *online*, dessa forma vai ser utilizado a tecnologia de *containers*, para o isolamento e distribuição dos serviços, junto com a arquitetura *serverless*, permitindo a entrega dos serviços de forma contínua e sem interrupção total do sistema em caso de possíveis falhas.

### Descrição

Containers são uma forma de isolar aplicações inteiras da máquina *host*, podendo ser distribuídas posteriormente. Seu funcionamento é similar a uma máquina virtual, mas sem a necessidade de configurar um sistema operacional e as ferramentas utilizadas para executar a aplicação. O gerenciamento e planejamento da capacidade do servidor são ocultadas do desenvolvedor, isso significa que os desenvolvedores não precisam se preocupar com gerenciamento, provisionamento e manutenção de servidores ao implantar o código.

São utilizadas para distribuição de aplicações *webs* e/ou armazenamento de dados, sendo aplicações distribuídas em servidores e sendo escaladas conforme a necessidade. *Docker*, uma plataforma de *containers*, já é utilizada no serviço payPal e Visa.

A utilização de *Kubernetes* realiza o trabalho de orquestração dos *containers* ajudando garantir a saúde, escalabilidade da aplicação conforme demanda, na implantação da aplicação em ambientes *cloud* e no isolamento do servidor para acessos externos.

A arquitetura *serverless* facilita no gerenciamento da aplicação, onde somente é cobrado pela quantidade de requisições que a aplicação recebe, não possuindo custos em períodos ociosos.

### Justificativa de Uso da Tecnologia

No passado era necessário que a equipe de infraestrutura fosse responsável por gerar um executável da aplicação que então era implantado no servidor, na maioria das vezes esse servidor era um computador alocado dentro da própria empresa. Em muitos casos essa aplicação precisava ser executada em uma versão específica de sistema operacional e várias dependências precisavam ser instaladas antes que a aplicação fosse executada.

Esse processo era muito custoso, exigia uma equipe capacitada e um *hardware* que constantemente precisava ser atualizado para suportar a aplicação, gerando diversas dificuldades e um custo alto para manter.

Para resolver o problema de possuir uma equipe de infraestrutura e ter um custo alto de investimento em *hardware*, será utilizada a tecnologia de containers, na qual possibilita que a aplicação e todas as suas dependências sejam empacotadas e executadas em qualquer ambiente que possua um *container app* instalado, sendo desnecessário que o responsável por implantar a aplicação conheça quais as dependências requeridas pela aplicação com isso tem um ganho em produtividade, manutenção e custo.

Outra vantagem da utilização de *containers* é a facilidade de utilizar em servidores *cloud*, pois a maioria dos provedores já possuem ferramentas consolidadas para a execução de *containers*.

Nos últimos anos várias aplicações criadas para as diversas áreas apresentaram os mesmos problemas, incapacidade de atender a demanda em momentos de grande pico de utilização e controlar os erros no sistema, um erro acabava derrubando o sistema inteiro, outro problema enfrentado nos projetos era a dificuldade em adicionar novas funcionalidades ou corrigir um erro em aplicações que já tinha atingido um tamanho macro pois toda a aplicação estava junta no mesmo módulo. Para resolver esses problemas será utilizada a tecnologia de micro serviços, na qual consiste em dividir a aplicação em pequenos serviços focados em uma parte do domínio. Um micro serviço é facilmente replicado caso exista a demanda, ao contrário de um serviço macro no qual a aplicação inteira precisa ser replicada para suprir a demanda, outra vantagem na utilização de um micro serviço e o isolamento de erros, caso um erro ocorra em um micro serviço específico ele pode ser isolado utilizando *Circuit Breakers* do micro serviço.

A combinação de containers com micro serviços expande as possibilidades de implantação de apenas um micro serviço da aplicação, sem a necessidade de implantar a aplicação inteira caso uma atualização seja realizado em alguma regra de negócio.

Por causa desses problemas será utilizada a tecnologia de container combinada com microservice, possibilitando uma redução de custos, gerando uma maior competitividade da aplicação no mercado concorrido.

**Arquitetura Mesh**

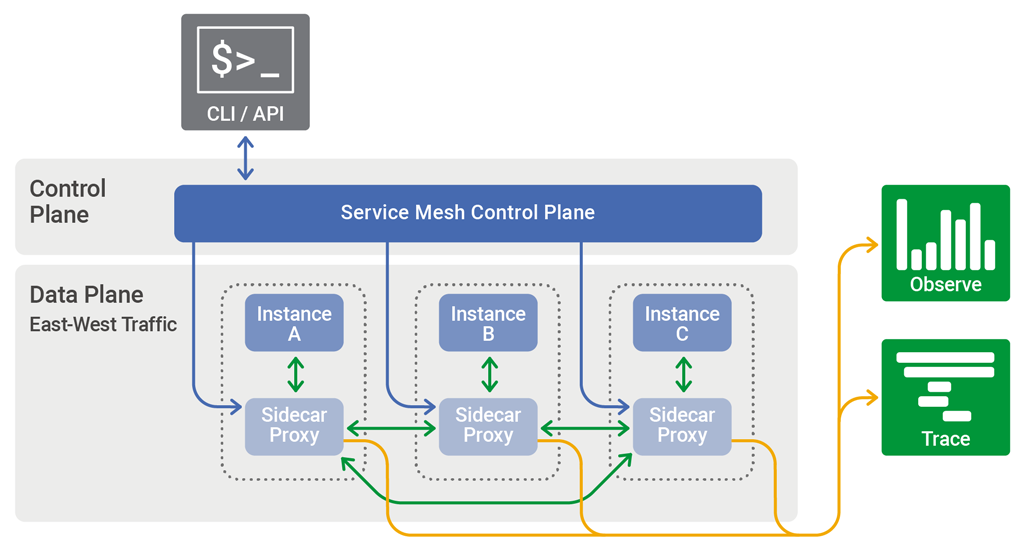
A utilização de uma arquitetura de microservices apresenta várias dificuldades na criação e execução, dentre os problemas podemos destacar os seguintes:

* Criação de um roteamento entre o frontend e os diversos *microservices*, no qual o frontend nao precisa fazer uma requisição específica para um *microservice*.
* Os *microservices* vão estar sendo executados no ambiente cloud, cada serviço vai possuir um IP específico que poderá mudar caso o serviço seja reiniciado, para isso será necessário algum mecanismo que consiga gerenciar e fornecer as informações necessárias para que esses serviços sejam visíveis dentro do cluster.
* Uma das vantagens de *microservices* é a capacidade de escalabilidade horizontal, para isso ser possível é necessário que exista um mecanismo que consiga distribuir a carga entre as instâncias existentes, para que não ocorra de uma instância ter 99% de carga enquanto outra possui apenas 10%.
* Outra necessidade é a capacidade da aplicação superar/isolar os erros/timeouts ocorridos em um serviço específico, falhar rápido, para que o sistema consiga se recuperar com mais rapidez
* Vários *microservices* torna a tarefa de monitorar e debugar muito complicado, para isso é necessário um conjunto de ferramentas que possibilite uma melhor maneira de realizar esse processo.
* Outro problema e como assegurar a utilização de TLS (Transport Layer Security ) em toda a aplicação
* Toda aplicação possui diversas chaves e passwords para acessar arquivos, banco de dados ou aplicações que estão rodando em clusters diferentes, com a utilização de *microservice* essa parte se torna complicada, compartilhar todas essas informações com todos os *microservices* pode se tornar complicado.
* Controle de requisições, observar a existência de tokens, verificar se o IP possui permissões de acesso ao cluster, esses problemas são comuns em aplicações e a utilização de microservices agrava bastante, pois é necessário replicar em todos os serviços, é necessário um mecanismo que consiga abstrair esse problema.
* Comunicação entre vários serviços é uma das etapas mais difíceis na utilização de microservices, certificar que as informações corretas estão chegando nos locais corretos sem que isso atrapalhe o funcionamento geral do sistema se mostra muito trabalhoso.

Para solucionar os problemas apresentados será utilizada a arquitetura Mesh, na qual diversos componentes são apresentados que permitem abstrair os problemas para uma entidade específica que vai gerenciar uma parte da aplicação. Para gerenciar todas as requisições externas(vindas do frontend) será utilizado o Ambassador, que funciona nativamente com um ingress controller do nosso cluster, ele será o responsável por verificar os requests, validar os tokens, validar permissões, rotear o request para o microservice responsável, toda a parte de autorização e autenticação será abstraída para esse componente, sem a necessidade de replicar a lógica em cada microservice.

O conceito de arquitetura Mesh utiliza a técnica de sidecar proxy, no qual um container é acoplado em cada pod que está sendo executado em um ambiente cloud com kubernetes, esse sidecar é responsável por conectar o serviço com a entidade que gerencia o ecossistema de mesh, através dessa conexão o sidecar consegue comunicar qual a instância de IP esse pod está rodando, esse sidecar comunica os logs da aplicação para a entidade responsável por centralizar os logs da aplicação, no caso do nosso sistema será utilizado o Grafana e o Prometheus para essa task. Outro problema resolvido pelo Mesh é o de Load Balance, através dos sidecars é possível saber o nível de utilização de cada aplicação, possibilitando o Load Balancer distribuir a carga entre as instâncias disponíveis ou criar uma nova instância caso as instâncias disponíveis não tenham capacidade de suprir a demanda.

Esses sidecars acoplados aos pods possibilitam a ideia de service discovery, no qual cada serviço não precisa saber sobre a existência dos outros servicos e suas instâncias, essa parte é abstraída para uma entidade responsável por comunicar com os sidecars e fornecer as informações necessárias para que o sistema saiba em qual network está presente o serviço desejado, qual o IP, quantas instâncias estão disponíveis.



Na imagem podemos ver os pods com os Sidecars containers acoplados em azul claro, cada pod possui o seu próprio sidecar.

Os sidecars acoplados são responsáveis por verificar o tipo de tráfico que está chegando no serviço, essa funcionalidade aliada ao service discovery ajuda na comunicação entre serviços por permitir que serviços façam uma requisição para um serviço de um tipo específico utilizando protocolos de comunicação definidos pelos sidecars, caso uma requisição seja fora dos tipos aceitáveis o sidecar pode interceptar o request e transformar para o formato ideal ou rejeitar sem a necessidade da utilização do serviço.

Cada sidecar fica responsável por um pod, isso permite que caso o container que está sendo executado dentro do pod apresente um erro o sidecar pode disparar um sinal para que o container seja terminado e uma nova instância seja inicializada.

# Bibliografia

WHAT are Linux containers?. [S. l.], 2019. Disponível em: https://opensource.com/resources/what-are-linux-containers. Acesso em: 19 fev. 2019.

What is a container?. Disponivel em: <https://www.docker.com/resources/what-container>.

What are microservices?. Disponivel em <https://smartbear.com/solutions/microservices/>.

How Software Development has changed. 07 de Junho de 2016. <https://dzone.com/articles/how-software-development-has-changed>.

Top software failures in recent history. 06 de Dezembro 2018. <https://www.computerworlduk.com/galleries/infrastructure/top-software-failures-recent-history-3599618/>

<https://epocanegocios.globo.com/Inspiracao/Carreira/noticia/2013/01/sete-coisas-que-um-headhunter-nao-conta-e-que-voce-sempre-quis-saber.html> -> acessado em 06/03/2019

<http://www.foursales.com.br/carreira/como-trabalho-headhunter-pratica/> -> 26/03/19

<https://serverless-stack.com/chapters/pt/what-is-serverless.html> -> acessado em 02/04/2019