Desenvolvimento de uma biblioteca de consulta a inspetores para a indústria extrativa

Junior Zilles, Mario R. N. Marques Junior, Eduardo N. Borges, Eder M. Gonçalves, Danúbia B. Espíndola, Sílvia S. da Costa Botelho

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Av. Itália, km 8, 96203-900, Rio Grande – RS – Brazil

{juniorzilles123,marioricardonascimento,}@gmail.com

{eduardoborges,edergoncalves,dmtdbe,silviacb}@furg.br

Resumo. *Uma das atividades mais importantes na indústria extrativa é a inspeção de solda, equipamentos e obras. Esta atividade envolve contratar profissionais com habilitações diversificadas, incluindo técnicas de ensaios não destrutivos. Com o objetivo de facilitar a busca por profissionais qualificados em diferentes órgãos certificadores, este artigo apresenta a construção de uma API RESTful que implementa um serviço Web de consulta por inspetores usando similaridade textual. A API desenvolvida será utilizada na construção de diferentes sistemas computacionais em uma arquitetura de microsserviços.*

1. Introdução

A indústria de extração minérios, incluindo petróleo e gás, faz uso de uma série de equipamentos que exigem diversas inspeções antes de entrarem em aplicação. Estas inspeções devem ser realizadas apenas por inspetores certificados. A consulta das técnicas habilitadas por um determinado inspetor pode ser realizada por meio da página Web de cada órgão certificador. Estes órgãos publicam de diferentes formas as informações sobre os profissionais certificados e as técnicas habilitadas. Além disso, os inspetores são identificados por matrículas ou chaves distintas em cada órgão. Uma empresa interessada precisa consultar vários órgãos usando diferentes sistemas. Além disso, as consultas são realizadas por igualdade, ou seja, o nome pesquisado precisa estar igual ao salvo na base de dados. Não são consideradas variações de grafia ou erros de digitação.

Este trabalho surgiu da necessidade de uma empresa extrativa brasileira em validar em uma única base de dados unificada registros de projetos de construção e montagem de unidades de extração e/ou processamento. Estes registros digitais são oriundos de um processo de reconhecimento ótico de caracteres, portanto podem apresentar erros na detecção de padrões.

Neste artigo é apresentado o desenvolvimento de uma Application Programming Interface (API) RESTful que implementa um serviço Web de consulta por inspetores em múltiplas fontes. Através de funções de similaridade textual, a API trata erros de digitação e variações de grafia.

O restante do texto está organizado da seguinte forma. Conceitos básicos são abordados na Seção 2. Trabalhos relacionados são sintetizados na Seção 3. Na Seção 4 é descrito como foi criado o serviço e as tecnologias utilizadas. Os resultados alcançados, incluindo a estrutura de rotas da API, são apresentados na Seção 5. Por fim, o trabalho é concluído na Seção 6.

2. Fundamentação teórica

REST é uma coleção de princípios e restrições arquiteturais para o desenvolvimento de aplicações distribuídas na Web [Salvadori 2015]. Em outras palavras, REST é uma arquitetura com princípios que dizem respeito a recursos disponíveis na aplicação. A interação se dá através dos verbos do protocolo HTTP GET, PUT, POST e DELETE. Logo, uma API é chamada de RESTful quando segue todos os princípios da arquitetura REST.

Segundo [Malipense and Zuchi 2018], arquitetura de microsserviços é um modelo arquitetônico que aborda o desenvolvimento de uma aplicação composta por vários serviços autônomos, ou seja, independentes dos demais e responsáveis por um conjunto finito de funcionalidades. Estes serviços se comunicam muitas vezes através de APIs de recurso HTTP.

Docker disponibiliza uma arquitetura de microsserviços e se utiliza do sistema de contêineres, que segundo [Bernstein 2014] são uma forma de virtualização ao nível do sistema operacional, que viabilizam a execução de múltiplos componentes de *software* isoladamente em um mesmo *host*.

3. Trabalhos relacionados

Os resultados para a busca por trabalhos que utilizassem uma API para disponibilizar dados teve por retorno o trabalho de [Ferreira et al. 2018] que procura facilitar a consulta de informações do sistema acadêmico do Instituto Federal de Alagoas. Esta API permite o acesso aos dados para consumo por alunos, professores e analistas de TI. Já [Serpa et al. 2018] faz uso de três APIs distintas que trabalham em conjunto integrando diferentes sistemas de modelagem oceanográfica. Assim como os demais trabalhos, [Mazzonetto et al. 2017] diz respeito a construção de uma API para acesso a dados climáticos e meteorológicos, que até o momento só estavam disponíveis em formato binário. Apesar de não encontrar artigos que lidem especificamente com uma biblioteca de inspetores, houve trabalhos com propostas semelhantes, ou seja, fornecendo dados de diversas fontes, necessitando o usuário realizar apenas uma única consulta. Estes estudos fundamentaram o desenvolvimento do presente trabalho.

4. Metodologia

A construção da API iniciou pelo desenvolvimento do modelo relacional de dados do inspetor e suas técnicas certificadas. Esse modelo foi desenvolvido utilizando a ferramenta Draw.io (Figura 1) e mais tarde o *software* pgModeler.

A estrutura relacional foi importada no SGBD PostgreSQL com sua interface de administração pgAdmin. Os dados foram extraídos das páginas de consulta das

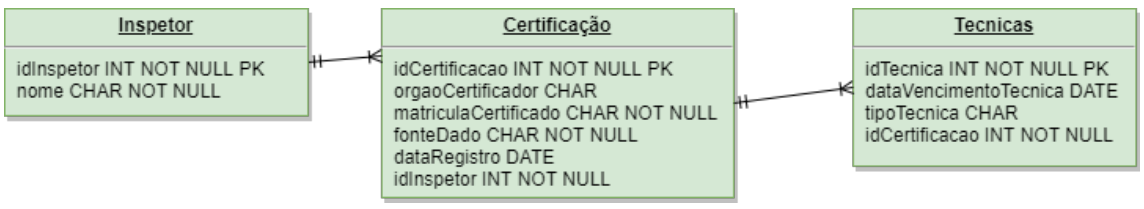

Figura 1. Estrutura relacional do inspetor.

Tabela 1. Métodos disponíveis para consulta na API

Identificador	Operação
lk	Consulta parte do nome
eq	Consulta o nome por igualdade. Ex. Nome=Joao
s	Consulta por similaridade utilizando a função *similarity**
ws	Consulta por similaridade utilizando a função *word_similarity**

* Funções implementadas pelo módulo pg_trgm do PostgreSQL

instituições certificadoras Abendi, ABRACO e FBTS utilizando Python com as bibliotecas *requests*[1] para a realização da requisição das páginas, *beautifulsoup4*[2] para manipulação do HTML e *psycopg2*[3] para realizar a conexão com o PostgreSQL para armazenar as informações. A criação da API RESTful foi feita através do Node.js, com *Swagger*[4] para a criação da documentação padrão gerado por um arquivo estruturado baseado em JSON/YAML informando as rotas, os parâmetros de entrada e a estrutura de retorno da aplicação. Além disso, é possível realizar testes na API através da interface gerada.

Por fim, se fez uso do Docker para permitir a estruturação do sistema na arquitetura de microsserviços. Logo, foi criado um contêiner para cada parte necessária, pgAdmin, PostgreSQL e Node.js, através do arquivo YAML chamado de *docker-compose* que permite a criação de mais de um contêiner simultâneo.

5. Resultados

O serviço foi desenvolvido com a criação de duas rotas. A primeira ./docs-api permite o acesso à tela de documentação gerada pelo Swagger, destacada na Figura 2 à esquerda. A segunda rota base é ./v3/obterinspetor. Para retornar o inspetor desejado é necessário informar o nome ou a matrícula, ou ambas as informações. Além destes parâmetros é possível informar o limite de registros retornados e o método de pesquisa. Foram implementados três métodos diferentes, identificados por siglas conforme a Tabela 1.

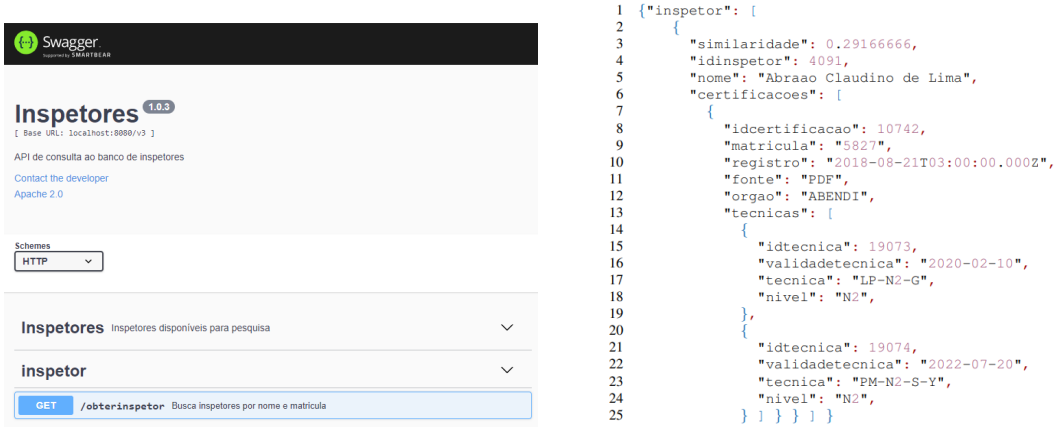

Figura 2. Tela de documentação (à esquerda) e lista de inspetores retornada por uma consulta exemplo (à direita)

[1]https://pypi.org/project/requests/
[2]https://pypi.org/project/beautifulsoup4/
[3]https://pypi.org/project/psycopg2/
[4]https://swagger.io/

As consultas por similaridade utilizam o módulo pg_trgm do PostgreSQL, que fornece um conjunto de operadores e funções de similaridade que comparam cadeias de caracteres a partir do casamento de trigramas (*3-gram matching*). Este pacote foi escolhido porque permite a indexação dos trigramas a priori, acelerando as consultas quando comparado a outras funções de similaridade. Foram utilizadas as funções *similarity* e *word_similarity*.

A Figura 2 à direita apresenta o retorno de uma consulta exemplo utilizando como argumentos nome = "abraao" e limite = 1. Os dados são apresentados em formato JSON contendo uma lista de inspetores e suas técnicas coletadas nos órgãos certificadores. A lista retornada é ordenada pelo grau de similaridade (linha 3) em relação ao nome utilizado na consulta. Para cada fonte de dados (linha 11), correspondendo a origem do dado PDF ou WEB, é apresentada um conjunto de competências (linhas 14-24), sendo as técnicas uma especialização, indicando a categoria de análise de material ou tipo solda que o inspetor pode analisar.

6. Conclusão

Nesse artigo foi apresentado o desenvolvimento de uma biblioteca de inspetores funcional, que atende aos requisitos iniciais de uma empresa de extração brasileira. Entre os trabalhos futuros destacam-se incluir mais funcionalidades, melhorar o tratamento de exceções, alterar o contêiner do Docker para realizar a restauração do banco de forma automática, implantar a API em um servidor na nuvem da empresa. Além disso, é importante definir a periodicidade e as rotinas de atualização dos dados, uma vez que novos inspetores podem ser certificados, inspetores já cadastrados podem renovar suas técnicas vencidas ou ainda deixar a profissão.

Referências

Bernstein, D. (2014). Containers and cloud: from lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

Ferreira, A. S., Nicacio, J. M., Ferreira, G. V., Filho, G. M. S., and Rodrigues, V. S. (2018). Desenvolvimento de uma api rest para um sistema acadêmico de terceiros. *Revista de Sistemas e Computação*, 8(2).

Malipense, L. M. and Zuchi, J. D. (2018). Um estudo sobre o conceito e a aplicação da arquitetura de microsserviços. *Revista Interface Tecnológica*, 15(1):122–134.

Mazzonetto, A., Borella, F., Chitolina, P., Tochetto, G., Casiraghi, J., de Oliveira, F. A. A., Chan, C. S., Pavan, W., and Holbig, C. A. (2017). Plataforma web para acesso e disponibilização de dados climáticos. In *Anais do VIII Workshop de Computação Aplicada a Gestão do Meio Ambiente e Recursos Naturais*, Porto Alegre, RS. SBC.

Salvadori, I. L. (2015). Desenvolvimento de web apis restful semânticas baseadas em json. Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2015.

Serpa, P., Weitzel, L., and Calado, L. (2018). Integração de sistemas de modelagem numérica oceanográfica e sistemas computacionais autônomos por meio de web service. *Revista de Sistemas de Informação da FSMA*, (22).