MeContrata - Uma plataforma de contratação

Rainerio L. C. Filho
UFERSA
Pau dos Ferros, Brasil
rainerio.filho@alunos.ufersa.edu.br

Rosendo L. S. Costa
UFERSA
Pau dos Ferros, Brasil
rosendo.costa@alunos.ufersa.edu.br

Sharlles A. C. Antunes
UFERSA
Pau dos Ferros, Brasil
sharlles.antunes@alunos.ufersa.edu.br

***Resumo*—No atual cenário de evolução tecnológica, a maneira como as empresas e os profissionais interagem no mercado de trabalho esta em constante transformação. Nesse cenário dinâmico e interconectado, a aplicação MeContrata surge como uma solução inovadora e altamente relevante. Este artigo explora como a utilização da arquitetura de microsserviços e APIs REST se tornou fundamental no desenvolvimento e operação dessa plataforma. Ao mesmo tempo, destaca a necessidade crucial de verificar e validar as interações por meio das quais os usuários se conectam e colaboram na "MeContrata". Além de fornecer uma visão das APIs REST e da arquitetura de microsserviços, explicando como esses conceitos são fundamentais para o desenvolvimento de software e sistemas modernos.**

Através deste estudo de caso, será possível compreender de forma mais aprofundada a complexidade e a inovação que cercam o desenvolvimento de software em um ambiente tecnológico em constante evolução. Consequentemente, nossa análise enfatiza como a "MeContrata"representa um exemplo significativo de como as APIs REST e a arquitetura de microsserviços se tornaram elementos fundamentais na criação de uma plataforma eficiente e altamente funcional para o mercado de trabalho moderno.

***Index Terms*—APIs REST, Teste de Software, Arquitetura de Microsserviços, Plataforma "MeContrata".**

I. Introdução

O MeContrata representa a revolução que o mercado de contratação estava esperando. Com um compromisso sólido com a igualdade de oportunidades, nossa plataforma conecta profissionais altamente especializados em diversas áreas a empresas em busca de talento. Especialmente no setor de tecnologia, estamos totalmente empenhados em identificar os melhores talentos e conectá-los aos nossos parceiros ideais. Nosso objetivo é tornar esse encontro perfeito uma realidade, atendendo às necessidades dos candidatos em busca de oportunidades e das empresas em busca de profissionais excepcionais. Com este projeto, temos como objetivo detalhar o desenvolvimento do sistema proposto, bem como a execução dos testes realizados para contribuir com a robustez e a funcionalidade desse sistema.

O MeContrata é uma plataforma web inovadora focada principalmente no recrutamento e seleção de profissionais altamente qualificados, com um enfoque especializado no setor de tecnologia. A plataforma será desenvolvida utilizando as tecnologias mais avançadas e seguirá as melhores práticas de desenvolvimento web. A arquitetura da plataforma é baseada em uma API (*Application Programming Interfaces*) sendo utilizado uma estrutura cliente-servidor.

II. fundamentação teórica

Descreveremos uma revisão da literatura para ilustrar os principais conceitos relacionados a API (*Application Programming Interfaces*), API REST (*Representational State Transfer*) e como foram feitos os testas em relação a sua arquitetura.

A. API's

As APIs (*Application Programming Interfaces*) desempenham um papel fundamental na interconexão de sistemas e aplicativos, possibilitando a troca de informações e funcionalidades de forma eficiente. API pode ser descrita como mediadora entre cliente e servidor, que possibilita a comunicação entre serviços (ARCURI, 2019).

Um ambiente de testes robusto e eficaz para APIs é essencial para garantir a qualidade e a confiabilidade das interações entre sistemas e, portanto, tem sido objeto de atenção significativa na área de desenvolvimento de software. Uma vez que Software são desenvolvidos com o uso de APIs, deve-se manter um padrão de qualidade elevado, para que este possibilite a comunicação entre as integrações desenvolvidas de forma segura (BANIAS *et al.*, 2021).

A evolução constante das tecnologias e a crescente dependência de sistemas interconectados tornam ainda mais crucial o investimento em ambientes de testes dedicados a APIs. Com a rápida expansão do ecossistema de software, a garantia de que as APIs funcionem de maneira confiável, atendendo aos requisitos de segurança e desempenho, é fundamental para evitar interrupções indesejadas nos processos empresariais e assegurar uma experiência positiva para os usuários. Além disso, a documentação e a padronização de APIs desempenham um papel vital nesse cenário, tornando a compreensão e a integração com esses serviços mais eficazes, contribuindo assim para a qualidade geral do software e das interações entre sistemas (BANIAS *et al.*, 2021).

B. API's REST

As APIs REST (*Representational State Transfer*) são interfaces de programação de aplicações criadas com foco em aplicativos que adotam a arquitetura de microsserviços e utilizam o protocolo HTTP para a troca de informações. O REST oferece um conjunto de linhas de orientação necessárias para se desenvolver um serviço coeso, escalável e com elevada performance.

São baseados em alguns princípios fundamentais sobre a representação do estado do recurso (como dados e serviços)

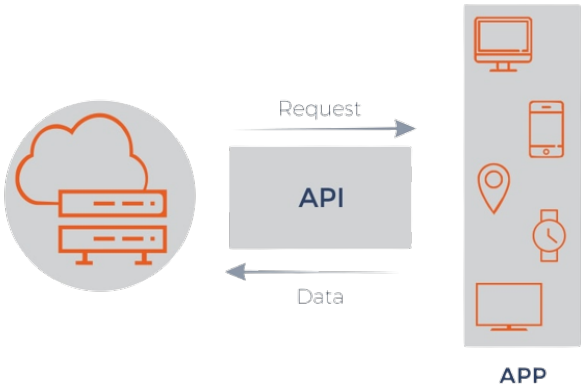
através de URLs (*Uniform Resource Locators*) e o uso de métodos HTTP (*Hypertext Transfer Protocol*) padrão para operações, como GET para leitura, POST para criação, PUT para atualização e DELETE para exclusão.

Um outro importante principio sobre as APIs REST (*Representational State Transfer*), são projetadas para serem *stateless*, o que significa que cada solicitação do cliente para o servidor deve conter todas as informações necessárias para entender e processar a solicitação. Isso torna as APIs fáceis de escalar e altamente flexíveis.

A adoção da arquitetura REST oferece um conjunto abrangente de diretrizes essenciais para a criação de serviços coesos, altamente escaláveis e de alto desempenho. (MARQUES, 2018)

No contexto do desenvolvimento com base em API REST e na arquitetura de microsserviços, é de extrema importância garantir a conformidade das solicitações e respostas a essas APIs. Qualquer retorno de status incorreto ou informação inadequada pode resultar em diversas falhas no funcionamento do software. Portanto, a aderência aos testes de software desempenha um papel vital na validação desses serviços. Normalmente, os tipos de testes mais comuns nesse cenário incluem testes unitários, testes de integração, testes de componentes e testes *end-to-end*. (LIMA, 2022)

Figura 1. Exemplo de comunicação de API



Fonte: https://qatestlab.com/assets/Uploads/API-Application-Programming-Interface.jpg. Acessado em 19 de outubro de 2023.

C. Teste de software em API's

Dentre as diversas atividades abrangidas pela Engenharia de Software, o Teste de Software pode ser considerado a mais laboriosa e cara em termos de consumo de tempo e recursos. (RAFI; REDDY, 2012)

As técnicas de validação e verificação de um software englobam uma variedade de ações destinadas a assegurar que o software e seu desenvolvimento estejam alinhados com as especificações estabelecidas. Essas atividades de verificação e validação desempenham um papel essencial dentro do âmbito das práticas de Garantia da Qualidade de Software. (PINHEIRO, 2014)

III. Abordagem

Para testar o sistema, empregamos uma estratégia conhecida como "Teste de Caixa Branca". Essa técnica desempenha um papel crucial na avaliação do funcionamento interno dos componentes de um software (NETO, 2007). Optamos por esse método devido ao acesso que tínhamos ao código-fonte da aplicação, permitindo-nos examinar detalhadamente a lógica interna do software.

A verificação e validação desempenham uma função vital no processo de desenvolvimento de APIs (Interfaces de Programação de Aplicativos), assegurando que estas operem de acordo com as expectativas estabelecidas e cumpram os padrões de qualidade necessários. Através do emprego de testes de caixa preta, os quais possibilitam que desenvolvedores e avaliadores confirmem o correto funcionamento do software, detetem eventuais problemas e garantam a confiabilidade do resultado da aplicação.

A implementação de testes de caixa preta tem como objetivo verificar o comportamento de uma aplicação com base em suas entradas e saídas, sem se preocupar com a lógica interna do código do sistema. O processo segue sendo a identificação dos requisitos e criação dos casos de teste unitários, sendo cada caso independente e testando uma funcionalidade específica.

A. Requisitos do Sistema

Nesta seção, daremos início à exploração dos requisitos do sistema, que conforme (SOMMERVILLE, 2011) destaca, os requisitos têm a finalidade de descrever as funcionalidades e serviços providos pelo sistema, bem como suas limitações. Esses requisitos são tipicamente categorizados em duas vertentes: requisitos funcionais e requisitos não funcionais. Na etapa de desenvolvimento da primeira versão do sistema de trabalho, foi realizada uma elaboração dos principais requisitos funcionais para os estudos realizados.

- RN001: Cadastrar prestador de serviço;
- RN002: Editar prestador de serviço;
- RN003: Visualizar detalhes do prestador de serviço;
- RN004: Apagar prestador de serviço;
- RN005: Cadastrar empresas;
- RN006: Cadastrar ofertas de trabalho;
- RN007: Editar oferta de trabalho;
- RN008: Visualizar oferta de trabalho;
- RN009: Listar oferta de trabalho;
- RN010: Deletar oferta de trabalho;

B. Desenvolvimento

O desenvolvimento de um sistema representa uma jornada desafiadora, que demanda não apenas habilidades técnicas, mas também um toque de criatividade. É a fase em que conceitos e ideias abstratas se materializam em código concreto. Nesse estágio inicial, concentramos nossos esforços no desenvolvimento de uma API, que desempenha um papel crucial ao receber requisições HTTP, executar processamentos e fornecer respostas. Esta API será a base sobre a qual todo o sistema será construído, garantindo uma interação sólida e eficaz com uma futura aplicação Web.

Para a criação desta aplicação, optamos por empregar a linguagem de programação Typescript, utilizando do framework Nest.js, que tem se destacado como uma ferramenta poderosa e versátil para o desenvolvimento de aplicações web e APIs robustas. O que diferencia o Nest.js é a sua arquitetura modular, que simplifica a organização do código em módulos independentes, resultando em uma base escalável e de manutenção descomplicada. Esta escolha estratégica de tecnologias promete fornecer uma base sólida necessária para a construção de uma aplicação robusta e eficiente.

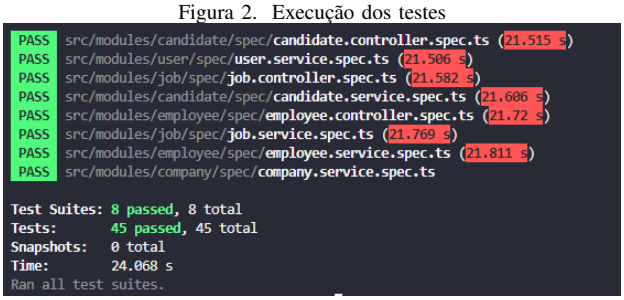
C. Testes

Na fase de testes, conforme estabelecido a utilização de abordagens de caixa branca, optamos por uma abordagem de desenvolvimento de testes unitários. O principal propósito desses testes é validar minuciosamente todas as funcionalidades presentes nos módulos do sistema, incluindo as *services* e *controllers*. Essa estratégia visa garantir que cada componente, seja ele um serviço que executa operações cruciais ou um controlador que gerencia a interação com os clientes, funcione de acordo com as especificações, atendendo aos requisitos e padrões predefinidos.

Para a realização dos testes, contamos com o poderoso framework Jest, amplamente reconhecido por suas capacidades de condução de testes automatizados em aplicações desenvolvidas em linguagens como *JavaScript* e *TypeScript*. O Jest oferece uma gama abrangente de recursos que simplificam o processo de teste e garantem a qualidade e confiabilidade de nosso código.

Conforme pode ser observado na Figura 2, realizamos um total de 8 *suits* de testes unitários. Esses testes abrangem uma ampla gama de módulos essenciais em nosso sistema, incluindo '*Job*,' '*Candidate*,' '*User*' e '*Employee*.' Cada um desses módulos desempenha um papel crítico na funcionalidade global do sistema, e os testes unitários nos fornecem a confiança de que cada componente opera de acordo com as expectativas. Isso é particularmente vital para manter a qualidade e a robustez do software à medida que continuamos a desenvolver e aprimorar nosso sistema."

Figura 2. Execução dos testes



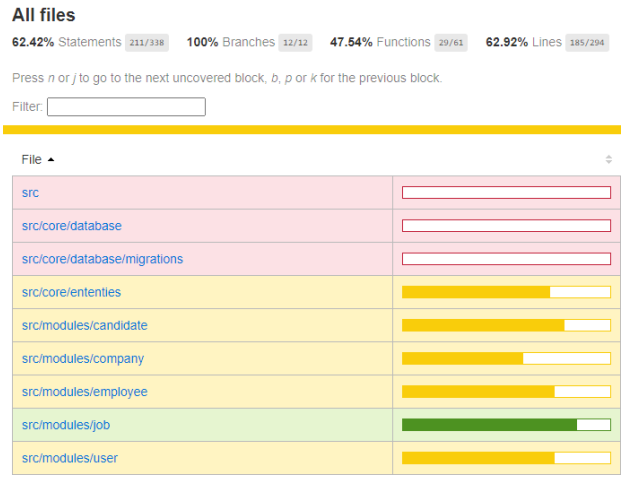
Fonte: Autoria própria.

Na Figura 3, é possível examinar as análises de cobertura nos módulos anteriormente mencionados. É notável que alguns arquivos carecem de cobertura, o que pode gerar perguntas. No entanto, essa ausência de cobertura é estrategicamente aplicada a módulos nos quais os testes unitários não teriam um impacto tão significativo ou eficiente. Nesses casos, a melhor opção seria posteriormente direcionar esforços para exeução testes de integração e testes end-to-end (E2E), uma vez que essas abordagens tendem a oferecer uma eficiência superior. Pois se trata de módulos responsáveis por estabelecer conexões com a base de dados ou inicializar estruturas particulares do framework Nest.js.

Observa-se na Figura 4 a cobertura de um dos módulos existentes no sistema. Como enfatizado anteriormente, os *services* e *controllers* desse módulo foram cobertos por testes unitários. Cada teste foi projetado para abranger minuciosamente os casos de teste relacionados aos requisitos específicos daquele módulo.

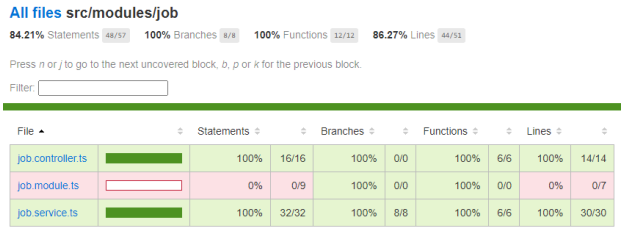
Essa abordagem garante uma validação detalhada e precisa das funcionalidades do módulo em questão, assegurando que todas as operações e cenários estejam adequadamente testados. Dessa forma, podemos ter confiança na qualidade e confiabilidade do módulo e, por extensão, de toda a aplicação.

Figura 3. Cobertura dos testes



Fonte: Autoria própria.

Figura 4. Cobertura dos testes do modulo de *job*



Fonte: Autoria própria.

IV. Considerações Finais

Com este artigo podemos afirmar que as APIs REST desempenham um papel fundamental no desenvolvimento de software moderno, especialmente quando incorporadas à arquitetura de microsserviços. Essas interfaces de programação

proporcionam uma maneira eficaz de permitir a comunicação entre sistemas e aplicativos, facilitando a troca de informações de forma eficiente. É essencial garantir que as solicitações e respostas estejam corretas, uma vez que erros podem acarretar falhas significativas no software.

A aderência aos princípios REST, como a uniformidade, o uso adequado de métodos HTTP (*Hypertext Transfer Protocol*) e a representação de recursos, é fundamental para o sucesso na implementação de APIs. Além disso, a realização de testes de software, como testes unitários, de integração, de componentes e testes *end-to-end*, desempenha um papel crucial na validação desses serviços, melhorando sua qualidade e confiabilidade.

À medida que a tecnologia continua a evoluir e a dependência de sistemas interconectados cresce, a compreensão e a aplicação adequada das melhores práticas relacionadas às APIs REST são cruciais para o desenvolvimento de software seguro, escalável e de alto desempenho.

V. Trabalhos Futuros

O propósito deste trabalho foi apresentar um sistema destinado a auxiliar profissionais de diversas áreas na divulgação de seus trabalhos, tornando-os mais acessíveis para possíveis oportunidades de colaboração com empresas. Neste contexto, proporcionamos uma visão do desenvolvimento do sistema e do processo de verificação e validação de software, mesmo que se tratasse de uma API relativamente simples.

No entanto, este trabalho não se encerra aqui. Continuaremos os esforços no desenvolvimento de uma primeira versão de um sistema web que se integrará harmoniosamente com a API existente. Nossos esforços futuros também se concentrarão no aprimoramento do *back-end*, abordando questões cruciais de autenticação e autorização do sistema.

Este projeto se concentrou na realização de testes de uma aplicação backend, ou seja, de uma API. Passaremos à direcionar esforços para a elaboração de artefatos destinados à execução de testes em aplicações *front-end*, com foco em serviços Web. Esses estudos se beneficiarão da plataforma desenvolvida para o ”MeContrata”, proporcionando um recurso valioso para o aprendizado na área de teste de software.

Referências

ARCURI, A. Restful api automated test case generation with evomaster. **ACM Transactions on Software Engineering and Methodology (TOSEM)**, ACM New York, NY, USA, v. 28, n. 1, p. 1–37, 2019.

BANIAS, O.; FLOREA, D.; GYALAI, R.; CURIAC, D.-I. Automated specification-based testing of rest apis. **Sensors**, MDPI, v. 21, n. 16, p. 5375, 2021.

LIMA, W. L. d. S. **Aplicando BDD em testes de REST API: uma experiência prática**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2022.

MARQUES, A. I. A. **Desenvolvimento de API para aplicação cloud**. Tese (Doutorado), 2018.

NETO, A. **Introdução a teste de software**. [S.l.: s.n.], 2007. v. 1. 22 p.

PINHEIRO, P. V. P. **Teste baseado em modelos para serviços RESTful usando máquinas de estados de protocolos UML**. Tese (Doutorado) — Universidade de São Paulo, 2014.

RAFI, D. M.; REDDY, K. M. K. **Automated software testing: A study of the state of practice**. 2012.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.]: São Paulo: Pearson Prentice Hall, 2011. v. 9. ed.