Sistema de Agendamento de Serviços Acadêmicos

Geísa Morais Gabriel
UFERSA
Pau dos Ferros, Brasil
geisa.gabriel@ufersa.edu.br

Lívia Beatriz Maia de Lima
UFERSA
Pau dos Ferros, Brasil
livia.lima30332@alunos.ufersa.edu.br

***Resumo*—O resumo deve responder os seguintes pontos:**

- **Qual o contexto?**
- **Qual o problema?**
- **Qual a relevância?**
- **Qual a sua contribuição?**
- **Quais as conclusões (os achados)?**

***Index Terms*—key1, key, ..., keyn**

I. Introdução

II. Fundamentação Teórica

Esta seção, possui como finalidade expor os conceitos necessários para o melhor entendimento do assunto abordado. Dessa forma, é apresentada a definição de sistemas web; logo após, é relatado aspectos associados a qualidade de software; depois disso é definido o conceito e ramificações a respeito do teste de software. Além disso, são apresentadas a documentação realizada na elaboração de sistemas, bem como as tecnologias e ferramentas utilizadas.

A. Sistemas Web

Um sistema Web é, normalmente, constituído por um conjunto de páginas Web e por uma base de dados [3]. Esses sistemas utilizam a arquitetura cliente-servidor. Neles, a comunicação do cliente no navegador ocorre através do endereço IP (*Internet Protocol*) do servidor, responsável por hospedar a estrutura de dados [5].

Nesse sentido, espera-se desenvolver um sistema Web de agendamento de serviços acadêmicos para a Ufersa, Campus Pau dos Ferros, que facilite a disponibilidade dos serviços ofertados por meio da integração e automatização do processo de agendamento. Além disso, deve-se aplicar técnicas de Verificação e Validação de Software a fim de incluir atividades de garantia da qualidade do software.

B. Qualidade de software

Historicamente, empresas perdiam bilhões de dólares em software devido a más implementações de funcionalidades. Sob esse aspecto, a qualidade de software refere-se à avaliação de um produto que atenda aos requisitos solicitados pelo cliente. O conceito de qualidade estende-se, portanto, em dois segmentos: qualidade do processo e qualidade do produto [12].

Nesse sentido, a garantia da qualidade de software engloba atividades de apoio aos processos, a fim de construir produtos adequados para o uso pretendido. Logo, a área de garantia da qualidade empenha-se em verificar os processos e suas definições, bem como validar o produto final a partir de um ciclo de revisões, melhoria e adaptação contínua [12].

C. Teste de software

Em sistemas de software cada vez mais complexos, a prática de testes para a descoberta de inconsistências torna-se cada vez mais necessária. Assim, a aplicação de testes, além de identificar a presença de erros, é fundamental para a garantia de qualidade e validação dos requisitos de um sistema de software [11].

Na realização de testes, cria-se um conjunto de casos de teste, ou seja, uma gama de condições que descrevem os comportamentos a serem testados no software a partir da análise das especificações do sistema [8]. Para tal, são estabelecidas condições necessárias a fim de determinar se o teste passou ou falhou.

Com base nisso, os testes podem ser definidos em dois aspectos gerais - a estrutura interna do programa e o seu aspecto funcional. Assim, os testes podem ser direcionados para o exercício da lógica dos componentes internos (teste de caixa branca) ou ainda podem ser voltados para descobrir erros no funcionamento, comportamento e desempenho do programa (teste de caixa preta) [8].

A organização dos testes é ainda efetuada em etapas, adotadas através das técnicas de teste de unidade, de integração e de sistema. Nesse viés, o teste unitário foca na garantia do funcionamento correto do componente do sistema. Do agrupamento entre os componentes origina-se o teste de integração. Após a integração, é realizado o teste de sistema, responsável por verificar o desempenho adequado da combinação dos elementos [9].

D. Documentação de sistemas

A documentação de software é essencial por dois motivos principais, uma das razões é facilitar a comunicação no processo de desenvolvimento do projeto e outra para esclarecer o conhecimento do programa nas atividades de manutenção (Ambler apud de [2]). Para serem elaborados os mais variados casos de testes, é necessário o entendimento sobre aquilo que está sendo testado. A documentação atua, portanto, como uma facilitadora, esclarecendo precisamente ao testador o comportamento esperado do programa, bem como os possíveis desvios de fluxo.

E. Tecnologias e ferramentas

Devido à crescente complexidade dos softwares, irrompe a necessidade de sistematizar tarefas, a fim de torná-la menos suscetível ao erro humano e menos custosa [1]. Para tal, apresentam-se algumas tecnologias e ferramentas de apoio ao processo de Verificação e Validação.

1) SonarLint: SonarLint [1] é um *plugin* para IDE gratuito e de código aberto responsável por encontrar e corrigir problemas de codificação. A disposição para o uso da ferramenta está em garantir a qualidade do código e aumentar a produtividade na resolução de problemas.

A tecnologia possui suporte para mais de 20 linguagens e usa mais de 5000 regras de *Clean Code* específicas de linguagem que buscam identificar erros comuns de codificação, *bugs* e vulnerabilidades.

2) NUnit: O NUnit [2] é um *framework* de teste unitário para todas as linguagens .Net. No processo de elaboração e identificação dos testes, o NUnit usa atributos personalizados e fornece um conjunto de asserções como métodos estáticos da classe Assert.

Para realizar testes unitários utilizando o *framework*, o código de teste desenvolvido deve conter asserções capazes de demonstrar o correto funcionamento da funcionalidade testada. Posto isto, os principais tipos de asserções estão entre as de igualdade, comparação, condição, identidade e tipos.

3) Selenium: O Selenium [3] é caracterizado como uma ferramenta para automação de testes de aplicação web. Sendo portátil e possuindo código aberto, o Selenium oferece suporte para diversos navegadores web, aplicações web e tecnologias.

O Selenium pode ser definido como um conjunto de diferentes ferramentas de software, cada qual com um objetivo específico a fim de auxiliar o processo de automação de testes baseado nas principais necessidades para testes em aplicação web [10].

4) DevTools e Lighthouse: O Chrome DevTools [4] é um conjunto de ferramentas para desenvolvedores da Web integrado diretamente ao navegador Google Chrome. Com ele, é possível diagnosticar problemas em tempo real, o que colabora no tempo de criação e edição de sites.

O Lighthouse [5] é uma ferramenta automatizada de código aberto incorporada ao DevTools e criada para melhorar a qualidade das páginas da Web. Por meio da execução dos testes usando a ferramenta, é possível gerar um relatório sobre o desempenho da página web.

III. Trabalhos Relacionados

O trabalho redigido pelos autores [6], é referente a elaboração de um software com intuito da informatização de processos e gestão do setor de Assistência Estudantil. No decorrer do artigo é destacado o processo de criação e desenvolvimento do software com proposito de otimizar os serviços relacionado a moradia estudantil.

Análogo, o trabalho de Helen [4], aborda uma análise de impacto em utilizar softwares para a automatização dos processos de gestão acadêmica. Nessa perspectiva, a autora ainda destaca, que um dos problemas da evasão dos alunos na adesão dos serviços é a demora pela conclusão do processo.

Ambos trabalhos demonstram a importância da automatização dos processos vinculados a gestão acadêmica, assim percebem-se alguns pontos em comum deste trabalho aos demais. Primeiramente por se tratar de temas de gestão acadêmica, sendo neste trabalho um nicho mais aprofundado em serviços relacionado a psicologia, nutrição e afins. Segundo, por se tratar da automatização dos processos e efetividade dos serviços realizados na universidade.

IV. Abordagem

Este estudo adota uma abordagem de cunho quali-quantitativo com a finalidade da construção e análise do Sistema de Agendamento de Serviços Acadêmicos (ASA).

Para isso, foi utilizado o método de pesquisa exploratória com intuito de explorar um problema para propor soluções e conhecimento acerca da problemática [7]. Dessa forma, o trabalho parte inicialmente da criação da identidade visual e padronização de informações do sistema, com o Manual de Identidade Visual e Prototipagem do Software.

Posteriormente, criou-se a documentação do sistema com finalidade de registrar as informações que caracterizam o software em sua totalidade. A partir disso, foi elicitado os requisitos funcionais e não funcionais, realizado a modelagem do sistema por meio de diagramas de caso de uso e de classe e também realizado a modelagem do banco de dados.

Sobre o modelo do sistema, foi designado o uso do padrão de projeto criacional *Singleton* e a linguagem de programação C# utilizando para o back-end o framework ASP.NET Core Web API e para o front-end o framework Blazor WebAssembly, diante de uma arquitetura cliente-servidor. Para a persistência dos dados foi utilizado o banco de dados MySQL e o Entity Framework Core, onde permitiu com essa estrutura criar e manter o sistema.

Para garantir o controle de qualidade, foi elaborado um plano de teste que inclui o escopo, metodologia e objetivos que guiam a avaliação do sistema. Dessa forma, foram escolhidos os testes de caixa branca e de caixa preta para a verificação e validação das funcionalidades do ASA, em diferentes níveis de testes. Sendo assim, serão conduzidos testes de unidade, integração e de sistema com auxílio de ferramentas de apoio como NUnit, Selenium, SonarLint e outros.

Por fim, após a análise diante dos testes será realizado um relatório contendo os casos de teste efetuados, bem como os erros e/ou falhas identificados. Dessa maneira, o estudo tem como principal característica a implementação e manutenção do sistema proposto.

[1] Para mais informações: https://docs.sonarsource.com/sonarlint/eclipse/

[2] Para mais informações: https://nunit.org/

[3] Para mais informações: https://www.selenium.dev/documentation/

[4] Para mais informações: https://developer.chrome.com/docs/devtools?hl=pt-br

[5] Para mais informações: https://developer.chrome.com/docs/lighthouse/overview?hl=pt-br

V. Considerações Finais e Trabalhos Futuros

Referências

[1] Boas, A. *Gestão de configuração para teste de software*. PhD thesis, Dissertação de Mestrado, FEEC/UNICAMP, Campinas, SP, 2003.

[2] de Souza, S. C. B., das Neves, W. C. G., Anquetil, N., and de Oliveira, K. M. Documentação essencial para manutenção de software ii. In *IV Workshop de Manutenção de Software Moderna (WMSWM), Porto de Galinhas, PE* (2007).

[3] Delamaro, M. E., Maldonado, J. C., and Jino, M. *Introdução ao Teste de Software*. Elsevier, 2016.

[4] Marques, H. M. F. *Gestão em processos acadêmicos: estudo de caso de uma Instituição de Ensino Superior privada na cidade de São Luis–Maranhão-Brasil*. PhD thesis, Escola Superior de Educação João de Deus, 2019.

[5] Martha, L. F. *Desenvolvimento de uma aplicação web para modelagem colaborativa*. PhD thesis, PUC-Rio, 2022.

[6] Melo, E. C., da Silva, G., and Cruz, M. A. Gestão informatizada de um setor de assistência estudantil:: Um estudo de caso no ifmg–campus. *International Journal of Management-PDVG 1*, 1 (2021).

[7] Moresi, E., et al. Metodologia da pesquisa. *Brasília: Universidade Católica de Brasília 108*, 24 (2003), 5.

[8] Pressman, R. S. *Engenharia de software*, 7 ed. AMGH, Porto Alegre, 2011.

[9] Pressman, R. S., and Maxim, B. R. *Engenharia de software*. McGraw Hill Brasil, 2016.

[10] Santori, R. P., et al. Avaliação da ferramenta de testes selenium no desenvolvimento guiado por teste de uma aplicação web. *Universidade de Brasília* (2019).

[11] Sommerville, I. *Engenharia de software*. Pearson Universidades, 2011.

[12] Zanin, A., Júnior, P. A. P., and Rocha, B. C. *Qualidade de software*. Grupo A, 2018.