Relatório Geral dos Testes

(Verificação e validação)

OnCatalog

Data de conclusão do relatório: 20/10/2024

Autores: Silvio Martins Santos; Allyson Bruno de Freitas Fernandes

Repositório: https://github.com/silviosmsantos/Verificacao-e-Validacao

1. INTRODUÇÃO

Neste relatório, será descrito o processo de testes, uma das etapas do processo de verificação e validação do produto OnCatalog, realizados até o momento. Trata-se de uma aplicação desenvolvida para a disciplina de verificação e validação, voltada para a criação de catálogos online, que facilita e organiza a exibição de produtos por empresas, tornando-as cada vez mais próximas de seus clientes. A seguir, serão abordados os testes realizados (unidade, integração e sistema), as ferramentas utilizadas e os principais problemas identificados.

- **Observações:** Detalhes sobre como ver a cobertura do testes está no arquivo readme.md, disponível no repositório.

2. FERRAMENTAS UTILIZADAS

Aqui está a lista de ferramentas empregadas para a realização dos testes:

- **Pytest**[1]: ferramenta empregada para a execução de testes unitários e de integração.
- **Selenium**[2]: utilizado para realizar testes funcionais e de sistema (end-to-end) em interface gráfica.
- **Postman**[3]: usado neste projeto para teste de integração das APIs;
- **K6**[4]: utilizado para testes de carga/estresse do sistema em ambiente de produção;

[1] https://docs.pytest.org/en/stable/index.html
[2] https://www.selenium.dev/
[3] https://www.postman.com/
[4] https://k6.io/open-source/

- **Git Actions[5]:** ferramenta de integração contínua utilizada para automatizar o processo de testes e deploy;
- **Coverage.py[6]:** utilizado para medir a cobertura dos testes unitários, indicando a porcentagem do código que foi exercitado pelos testes.

3. TESTES DE UNIDADE

3.1. Objetivo

Os testes de unidade foram aplicados para garantir o funcionamento esperado de componentes isolados da aplicação. Para este projeto, cada classe foi considerada como uma unidade. Todos os testes de unidade realizados para **services** e **repositories** estão disponíveis em:

A. catalog_project/core/tests/repositories
B. catalog_project/core/tests/services

3.2. Casos de teste

As seguintes classes foram testadas, os teste das classes abaixo podem ser conferidos nos diretórios mencionados anteriormente (A e B) :

catalog_project/core/repositories	catalog_project/core/services
audit_log_repository.py	audit_log_service.py
catalog_repository.py	catalog_service.py
category_repository.py	category_service.py
company_repository.py	company_service.py
message_repository.py	message_service.py
product_repository.py	product_service.py
user_repository.py	user_service.py

Classe sem teste de unidade associado:

- permission_repository.py
- permission_service.py

[5] https://github.com/features/actions
[6] https://coverage.readthedocs.io/en/7.6.3/

3.3. Resultados

Como foi utilizado o TDD (Desenvolvimento Orientado a Testes) durante o desenvolvimento do projeto, todos os testes de unidade passam com sucesso.

4. TESTES DE INTEGRAÇÃO

4.1. Objetivo

Os testes de integração realizados durante o desenvolvimento do OnCatalog visam verificar o comportamento esperado da aplicação quando diferentes módulos interagem entre si. Os arquivos que constam dos testes de integração podem ser acessados pelo seguinte diretório: "**catalog_project\catalog\tests**".

4.2. Casos de teste

Os seguintes arquivos são compostos por testes de integração:

- test_catalog_view.py
- test_company_view.py;
- test_home_view.py;
- test_login_view.py;
- test_message_view.py;
- test_password_view.py;
- test_register_view.py;
- test_user_view.py
- test_userProfile_view.py

4.3. Resultado

Como foi utilizado o TDD (Desenvolvimento Orientado a Testes) no desenvolvimento do projeto, todos os testes de integração passam com sucesso.

5. TESTES DE SISTEMA

5.1. Testes funcionais

Os testes funcionais foram realizados utilizando a ferramenta Selenium, com o objetivo de verificar se as funcionalidades implementadas na aplicação operam de acordo com os requisitos especificados. Além disso, há o planejamento para realizar um processo contínuo de atualização da suíte de testes (inclusive para utilizar como testes de regressão

posteriormente), adaptando-a a cada nova funcionalidade desenvolvida. Esses testes foram realizados considerando os recursos humanos e o tempo disponíveis, o que acabou limitando a cobertura completa de todas as funcionalidades desenvolvidas até o momento.

5.1.1. Casos de teste

A documentação detalhada dos casos de teste executados com o Selenium pode ser encontrada no repositório: **tests_docs/testes_funcionais_OnCatalog(selenium).pdf**. Esta documentação inclui os cenários de teste, os passos seguidos e os resultados esperados para cada funcionalidade verificada.

5.1.2. Resultados

Devido a limitações de tempo e equipe, nem todos os requisitos foram verificados. Além disso, alguns testes não puderam ser executados corretamente, e os motivos dessas falhas ainda não foram identificados até o momento. Detalhes sobre esses problemas podem ser consultados no documento:

- **tests_docs/testes_funcionais_OnCatalog(selenium).pdf**.

5.2. TESTES FUNCIONAIS EXPLORATÓRIOS

Os testes exploratórios foram realizados sem scripts predefinidos, permitindo uma livre navegação pela aplicação de forma livre para identificar comportamentos inesperados ou problemas não capturados pelos testes automatizados. Esses testes se concentraram em cenários reais de uso inclusive os testes acontecem em ambiente de produção, os testes cobrem áreas como:

1. **Navegação por diferentes catálogos**: Verificação de que a interação dos usuários com os catálogos de produtos ocorre conforme esperado.
2. **Envio de mensagens por clientes**: Avaliação do fluxo completo de envio de mensagens e do retorno das empresas.
3. **Verificação da responsividade**: Testes para avaliar o comportamento da aplicação em diversos dispositivos e resoluções de tela.
4. **Módulos administrativos**: Avaliação de grande parte das funcionalidades disponíveis para funcionários, incluindo o gerenciamento de catálogos e o controle administrativo de usuários.

5.2.1 Resultados

Os testes exploratórios ajudaram a identificar problemas menores na interface de usuário, como falhas na exibição de mensagens longas, formulários relacionados a número de telefone, e problemas de navegação em dispositivos móveis, que não foram capturados pelos testes automatizados. Os maiores achados com os testes exploratórios podem ser conferidos na Seção 6.

5.3. TESTES DE CARGA/ESTRESSE

Os testes de carga e estresse foram realizados para avaliar um dos requisitos não funcionais da aplicação, especificamente sua capacidade de lidar com um grande número de usuários e requisições simultâneas, simulando cenários de alta demanda. Esses testes mediram o desempenho do sistema em ambiente de produção sob condições extremas e o comportamento da aplicação sob estresse prolongado.

- **Ferramenta utilizada**: K6, essa ferramenta utilizada para simular múltiplos acessos simultâneos e monitorar o comportamento da aplicação.

- **Cenários testados:**
    - **Carga**: simulação de 100 usuários simultâneos navegando por um catálogo de produtos, onde o tempo de resposta esperado era ser menor ou igual a 3 segundos por requisição..
    - **Estresse:** simulação de um aumento progressivo no número de usuários até 400 simultâneos interagindo com uma página de catálogo. O tempo de resposta das requisições foi monitorado durante esse aumento chegando ao tempo de 22 segundos para aplicação responder uma requisição.

5.3.1. Resultados:

A aplicação se comportou com até 100 usuários simultâneos, atendendo às requisições dentro do tempo de resposta esperado de 3 segundos, o que atende ao requisito não funcional correspondente. O sistema também apresentou dificuldades em processar um grande número de mensagens simultâneas, sugerindo a necessidade de otimização no módulo de comunicação. Além disso, os testes revelaram gargalos de desempenho no banco de dados e na renderização de páginas com muitas imagens, indicando que são necessárias melhorias

para suportar cargas mais elevadas. Todo o processo realizado com o testes de carga e estresse podem ser verificados no documento no repositório:

- **tests_docs/teste_carga_estresse(K6).pdf**

6. PRINCIPAIS PROBLEMAS ENCONTRADOS

Esta seção apresenta os principais problemas identificados por meio dos testes realizados até a data de elaboração deste relatório. Todos os problemas relatados foram descobertos utilizando os recursos humanos e de tempo disponíveis. Cada problema já foi reportado e há um entendimento claro sobre suas causas e possíveis soluções.

6.1. PROBLEMA 1: IMAGENS SÃO APAGADAS NO DEPLOY

- **Descrição:** Ao realizar um novo commit, o diretório /app/media, onde são armazenadas as imagens dos produtos, é recriado, resultando na perda de todas as imagens previamente salvas.
- **Verificação**: Este problema pode ser verificado nos logs do GitActions e da aplicação hospedada no Railway app.
- **Possíveis Soluções:** Configurar um armazenamento de mídia persistente que não seja recriado a cada novo deploy. Pode-se usar uma solução de armazenamento externo, como AWS S3 ou volumes persistentes no Docker.

6.2. PROBLEMA 2: FALTA DE MÁSCARAS NO CADASTRO DE TELEFONE

- **Descrição**: Ao cadastrar o telefone, os usuários podem inserir números incorretos ou caracteres como “(“ ou “-” ou sem formatação, o que causa problemas de validação.
- **Verificação**: O problema pode ser observado ao inserir números como: (99) 9 9999-1000. A aplicação falha ao processar esses formatos.
- **Possíveis soluções**: Implementar máscaras de telefone no frontend, usando bibliotecas para garantir a formatação correta.

6.3. PROBLEMA 3: MENSAGENS LONGAS NO MÓDULO ADMINISTRATIVO

- **Descrição:** Quando uma mensagem enviada por um cliente tem muitos caracteres, ela não é completamente exibida no painel administrativo da empresa, onde funcionários podem ver mensagens dos clientes, o mesmo acontece com nomes de pessoas que são longos.
- **Verificação**: O problema pode ser observado no módulo de mensagens do painel administrativo ao visualizar uma mensagem com cerca de 80 caracteres ou mais.
- **Possíveis soluções**: Implementar um scroll automático ou expandir a área de visualização de mensagens para que todo o conteúdo possa ser lido.

7. CONCLUSÃO

O processo de verificação e validação da aplicação foi realizado com sucesso, utilizando testes de unidade, integração e sistema, respeitando os recursos disponíveis. Embora a aplicação esteja, em partes, operando conforme o esperado, foram identificados alguns problemas considerados graves. As soluções propostas são essenciais para melhorar a estabilidade e a experiência do usuário. A implementação dessas melhorias será crucial para corrigir as falhas observadas em busca de garantir a qualidade do produto final. Além disso, a continuidade do processo de verificação e validação é importante para assegurar a evolução constante da aplicação.