Relatório Final

Grupo 2
Felipe Moreira Ferreira - 204056
Lucas Guesser Targino da Silva - 203534
17 de julho de 2022

Introdução

Esse relatório visa apresentar os resultados obtidos pelos acadêmicos pela aplicação das técnicas de verificação e validação de software aprendidas na disciplina MO409 - Engenharia de Software I para exercitar a aplicação Shopizer.

Shopizer é uma aplicação web completa voltada para o e-commerce. Composta por Front-End, Back-end, e uma aplicação administrativa, fornece aos desenvolvedores todas as ferramentas necessárias para gerenciar e operar lojas online e aos usuários a experiência de compras online.

Para o desenvolvimento do presente trabalho, foram elaborados testes e cenários a fim de exercitar o Back-End da aplicação. Para tal, utilizou-se a ferramenta de teste de APIs Karate: ela permite que sejam executados testes escritos na linguagem Gherkin com sintaxe simples e intuitiva.

As seções seguintes apresentam, os testes realizados, desde sua concepção à sua especificação, bem como os resultados obtidos e análise.

Componentes da Shopizer

Além da divisão das aplicações, a Shopizer possui uma divisão de seus componentes. Entendê-los foi um passo importante a fim de decidir o escopo e objetivo dos testes desenvolvidos.

Abaixo, estão os componentes identificados:

1. Gerenciamento de inventário
    a. Catálogo
    b. Categoria
    c. Produto
2. Gerenciamento de mercado
    a. Fabricante / Marca
    b. Comerciante
3. Gerenciamento de clientes
4. Gerenciamento de compras
    a. Carrinho
    b. Pedidos

    c. Pagamento
    d. Envio
    e. Imposto
5. Gerenciamento de comentários
    a. Cliente
    b. Produtos
6. Ferramentas de busca
7. Gerenciamento de segurança

Para mais informações sobre as componentes e endpoints do backend, veja a API pública da Shopizer.

Teste de Partição de Equivalência

Para esse conjunto de testes, foram exercitadas três componentes, mostradas abaixo. Para cada uma delas, selecionou-se um endpoint do Back-End e gerou-se partições de equivalência para alguns parâmetros. Dada a alta complexidade e customização do sistema, exercitar todos os endpoints ou até mesmo todos os parâmetros de um endpoint mostrou-se custoso demais, de forma que selecionou-se apenas alguns deles para serem exercitados. O o apêndice A contém informações detalhadas sobre quais componentes, recursos e parâmetros foram testados, bem como as partições de equivalência utilizadas.

1. Gerenciamento de segurança
    a. endpoint: /api/v1/private/login
2. Gerenciamento de inventário
    a. Criação de categoria
        i. endpoint: /api/v1/private/category
    b. Criação de produtos
        i. endpoint: /api/v2/private/product/definition
3. Ferramentas de busca de produtos
    a. endpoint: /api/v1/products

Resultados

1. nenhum problema encontrado nos logins
2. nenhum problema encontrado na criação e remoção de produtos
3. foram detectados problemas na criação e remoção de categorias
    a. criação: dois testes e uma falha
    b. remoção: dois testes e uma falha
4. nenhum problema encontrado na busca pelos produtos criados
5. foram detectados problemas na associação de produtos a categorias
    a. alguns casos foram bem sucedidos e outros não
    b. os erros foram detectados através da verificação do código de retorno

Dificuldades encontradas

A API, embora apresente os endpoints e os dados esperados na requisição, não apresenta como as informações são integradas pela aplicação, o que dificulta o seu entendimento. Além disso, não é apresentado na API qualquer tipo de informação sobre os valores aceitos ou recomendados para os parâmetros.

Recomendações

Testar mais a integração de produtos e categorias. Essa específica funcionalidade não está apresentando o comportamento esperado.

Teste de Valor Limite

Para esse conjunto de testes, foi exercitada a funcionalidade de criação de categoria, similar ao que foi feito nos testes de partição de equivalência. Dessa vez, entretanto, o foco estava em utilizar valores próximos aos limites aceitos pela aplicação a fim de verificar e validar seu comportamento. Foram realizados 7 testes, 4 com valores dentro dos limites, e 3 com valores acima. O número de testes aqui é menor pois eles são limitados a requisições do tipo POST, já que é nessas que os parâmetros são fornecidos e podem, então, ser testados.

Constatou-se que, para valores no limite, a aplicação respondia conforme esperado. Entretanto, para valores acima do limite, o status HTTP de retorno esperado seria 400 (Bad Request), indicando que a requisição feita é inválida. Constatou-se que a aplicação não valida do valor enviado, ela tenta usá-lo e encontra algum erro interno, retornando então código 500 (Internal Server Error).

Dificuldades encontradas

Embora a API da shopizer forneça os endpoints e os parâmetros, não é dada nenhuma informação acerca dos valores aceitos nem de seus limites. Na prática, foi realizada uma busca: enviava-se uma requisição e observava-se o retorno da aplicação. Com essa prática, os acadêmicos puderam encontrar os valores limites. Para strings, normalmente o limite é 63 ou 255 caracteres e para inteiros, o limite de 9223372036854775807 (max int64) foi encontrado (graças a comentários na internet).

Recomendações

Utilizar os mesmos valores limites em outras componentes do sistema para verificar se em outros endpoints é feito alguma validação dos parâmetros de entrada.

User Stories

Para esse conjunto de testes, diversas componentes do sistema são testadas. O objetivo é validar casos de uso comuns. As histórias são descritas no Apêndice C. Nesses testes, verificou-se a capacidade do sistema de executar diversas operações em ordem a fim de satisfazer alguma necessidade específica do usuário.

Para esses casos foram feitas verificações tanto do código de resposta do sistema quanto do conteúdo das respostas. Nenhuma falha foi observada: a aplicação foi capaz de realizar as tarefas necessárias para cumprir com o requisito dos usuários.

Dificuldades encontradas

Para esses testes, é necessário configurar o sistema e popular a sua base de dados (usuários, produtos). A solução encontrada para isso foi, antes de executar os cenários de teste, são feitas inserções de dados (cadastro de usuários e produtos), e após a execução são realizadas limpezas (remoção dos cadastros). Isso garante repetitividade dos testes.

Recomendações

Esses são os testes mais importantes de serem feitos: eles verificam se as principais funcionalidades do sistema são entregues. Além disso, como esses testes lidam com integração de componentes, eles têm maior potencial de encontrar falhas no sistema. Deve-se salientar, entretanto, que realizar diagnósticos baseado nos resultados de tais testes é uma tarefa de maior complexidade para um desenvolvedor.

Conclusão

Durante o desenvolvimento dos projetos, ocorreram diversas discussões sobre testes:

1. O que testar?
2. Quando testar?
3. Como testar?
4. Como saber se os testes representam bem o comportamento real do sistema?
5. Como otimizar os esforços relacionados a testes?
6. Quais componentes testar?
7. Qual é a técnica mais adequada para cada componente?
8. Quais ferramentas podem ser utilizadas para otimizar o processo de teste?

Os exercícios desenvolvidos mostraram-se uma ótima oportunidade de pôr em prática os conhecimentos adquiridos ao longo do semestre na disciplina cursada. Principalmente pela parte prática: foi testada uma aplicação comercial utilizando ferramentas empregadas por profissionais da área.

Sobre a aplicação testada, a Shopizer, ela se mostrou pouco documentada, de uso moderadamente difícil, e com inconsistências entre o comportamento esperado e obtido. Entretanto, conforme mostram os resultados dos casos de teste de user stories, a aplicação consegue entregar as principais funcionalidades prometidas.

A ferramenta Karate se mostrou bastante boa para a realização dos testes. Ela é bastante intuitiva, fácil de usar, e com todos os recursos necessários para testes de API.

Apêndice A - Partições de equivalência

Componente	Recurso	Argumento	Tipo	Classes de Equivalência: Dados Inválidos	Classes de Equivalência: Dados Válidos
Security Management	Autenticação de usuário com permissão de administrador	username	String	I0: String Nula I1: String Vazia	V0: String Não Vazia
		password	String	I2: String Nula I3: String Vazia	V1: String Não Vazia
Inventory Management	Criação de Categoria	identifier	Integer	I4: Valor Nulo I5: Valor Negativo	V2: Valor Positivo
		visible	Boolean		V3: Valor Verdadeiro V4: Valor Falso
		name	String	I6: String Nula I7: String Vazia	V5: String Não Vazia
		friendlyUrl	String	I8: String Nula I9: String Vazia I10: String Não Vazia com caracteres especiais não escapados	V6: String Não Vazia com caracteres especiais escapados
	Criação de Produtos	identifier	Integer	I11: Valor Nulo I12: Valor Negativo	V7: Valor Positivo
		visible	Boolean		V8: Valor Verdadeiro V9: Valor Falso
		dateAvailable	String	I13: String Nula I14: String Vazia I15: String fora do padrão "yyyy-mm-dd"	V10: String dentro do padrão "yyyy-mm-dd"
		manufacturer	String	I16: String Nula I17: String Vazia	V11: String Não Vazia
		price	Float	I18: Valor Nulo I19: Valor Negativo	V12: Valor Positivo
		quantity	Integer	I20: Valor Nulo I21: Valor Negativo	V13: Valor Positivo
		name	String	I22: String Nula I23: String Vazia	V14: String Não Vazia

Search tools		friendlyUrl	String	I24: String Nula I25: String Vazia	V15: String Não Vazia
	Associar Produto a Categoria	categoryId	Integer	I26: Valor Nulo I27: Valor Negativo	V16: Valor Positivo
		productId	Integer	I28: Valor Nulo I29: Valor Negativo	V17: Valor Positivo
	Remoção de Categoria	identifier	Integer	I30: Valor Nulo I31: Valor Negativo	V18: Valor Positivo
	Remoção de Produtos	identifier	Integer	I32: Valor Nulo I33: Valor Negativo	V19: Valor Positivo
	Busca de Produtos baseado em seu ”SKU”	identifier	Integer	I34: Valor Nulo I35: Valor Negativo	V20: Valor Positivo
	Busca de Categoria baseada no seu Identificador	identifier	Integer	I36: Valor Nulo I37: Valor Negativo	V21: Valor Positivo

Apêndice B - Valores Limite

Componente	Recurso	Argumento	Tipo	Classes de Equivalência: Dados Inválidos	Classes de Equivalência: Dados Válidos	Valores Limite
Inventory Management	Criação de Categoria	identifier	Integer	I4: Valor Nulo I5: Valor Negativo	V2: Valor Positivo	N2: 9223372036854775807 (max int64)
		visible	Boolean		V3: Valor Verdadeiro V4: Valor Falso	
		name	String	I6: String Nula I7: String Vazia	V5: String Não Vazia	N3: String com 63 char
		friendlyUrl	String	I8: String Nula I9: String Vazia I10: String Não Vazia com caracteres especiais não escapados	V6: String Não Vazia com caracteres especiais escapados	N4: String com 63 char
	Criação de Produtos	identifier	String	I11: String Nula I12: String Vazia	V7: Valor Positivo	N2: String com 255 char
		dateAvailable	String	I13: String Nula I14: String Vazia I15: String fora do padrão "yyyy-mm-dd"	V10: String dentro do padrão "yyyy-mm-dd"	N3: datas válidas
		price	Float	I18: Valor Nulo I19: Valor Negativo	V12: Valor Positivo	N4: Valor acima de 1E+6
		quantity	Integer	I20: Valor Nulo I21: Valor Negativo	V13: Valor Positivo	N4: Valor acima de 1E+6
		name	String	I22: String Nula I23: String Vazia	V14: String Não Vazia	N2: String com 63 char
		friendlyUrl	String	I24: String Nula I25: String Vazia	V15: String Não Vazia	N2: String com 255 char

Apêndice C - User Stories

História	Número de cenários
1	1
2	5
3	2
4	3

H1

Iniciar sessão(Dados Inválidos)
-> Fazer Cadastro(Não cadastrado)
-> Iniciar sessão (Sucesso_1)
-> Acessar Dados de Usuário (Sucesso_2)
-> Modificar Senha do Usuário (Mudar Senha)
-> Modificar Senha do Usuário (Dados Inválidos)
-> Acessar Dados de Usuário (Sucesso_3)
-> Pedidos Recentes(Consultar Pedidos Recentes)

H2

1. Cenário 1: H2 story - user does not exist - create and delete
   Iniciar sessão(Dados Inválidos)
   -> Fazer Cadastro (Não cadastrado)
   -> Iniciar sessão (Sucesso_1)
2. Cenário 2: Teste Duplicado H2 story - login fails - then succeeds
3. Cenário 3: H2 story - list categories
   Iniciar sessão
   -> Listar Categorias (Sucesso_2)
4. Cenário 4: H2 story - category 1 product variants
   Iniciar sessão
   -> Listar Categorias (Sucesso_2)
   -> Exibir lista de Preços (Consultar Variação do Produto)
   -> Listar Categorias (Sucesso_3)
5. Cenário 5: H2 story - Absent Category Product Variants
   Iniciar sessão
   -> Listar Categorias (Sucesso_2)

-> Exibir Mensagem de Erro (Consultar Produto Inexistente)
-> Listar Categorias (Mensagem Exibida)

H3

1. Cenário 1: H3 story - user does not exist - create and delete
   Iniciar sessão(Dados Inválidos)
   -> Fazer Cadastro(Não cadastrado)
   -> Iniciar sessão (Sucesso_1)
2. Cenário 2: H3 story - Shopping Cart
   Iniciar sessão (Sucesso_2)
   -> Área Logada (Buscar Produtos)
   -> Lista de Produtos (Criar Carrinho / Adicionar Primeiro Produto)
   -> Produtos no Carrinho (AdicionarProdutos_1)
   -> Produtos no Carrinho (AdicionarProdutos_1)
   -> Quantidade do Produto Indisponivel (AdicionarProdutos_2)
   -> Produtos no Carrinho (Exibir_Erro_1)

H4

1. Cenário 1: H4 story - user does not exist - create and delete
   Iniciar sessão(Dados Inválidos)
   -> Fazer Cadastro(Não cadastrado)
   -> Iniciar sessão(Sucesso_1)
2. Cenário 2: H4 story - User Populate Shopping cart, Alters quantity outsize Available quantity and then Remove all items
   Iniciar sessão(Sucesso_1)
   -> Área Logada (Listar_produtos)
   -> Produtos no Carrinho (Alterar_Quantidade_1)
   -> Produtos no Carrinho (Alterar_Quantidade_2)
   -> Quantidade de Produto Indisponível (Exibir_Erro_1)
   -> Produtos no Carrinho (Desistir_Produto)
3. Cenário 3: H4 story - User Populate a Shopping Cart to Check its Total Cost
   Iniciar sessão(Sucesso_1)
   -> Área Logada (Listar_produtos)
   -> Produtos no Carrinho (Alterar_Quantidade_1)
   -> Página de Pagamento (Verificar_Valor_Total)