



Gama Cognizant – Desafio Final - QE&A Automation Academy

A empresa **XBookMarket** lançou um novo serviço de empréstimos e vendas online de livros digitais e deseja gerar **Planos de Testes** completos que contemplem todas as etapas do processo de desenvolvimento de software. Os Clientes devem realizar seu cadastro através da aplicação e apenas assim estarão habilitados para realizarem e confirmarem suas solicitações de empréstimos e /ou venda de obras literárias em formato digital. O desafio consiste em construir em Java a parte operacional da aplicação e uma proposta de camada de integração (recomenda-se a implementação de web services) que deverá ser usada como **objeto de teste**. Após isso, a equipe deverá planejar e documentar os testes automatizados, demonstrando a estabilidade, segurança, performance e confiabilidade dos principais processos do sistema, principalmente as operações envolvendo **CADASTROS**.

Requisitos:

- 1) O cliente deve possuir: Nome, E-mail, Telefone, CPF, RG e Endereço.
- 2) A IDE usada deverá ser o Eclipse.
- 3) Deverão ser feitos testes automatizados usando, no mínimo, as ferramentas Jmeter, SoapUI, Selenium e Cucumber.
- 4) Deverá ser usado o Git (para versionamento do código da aplicação) e Github para o repositório da equipe.
- 5) Métodos Ágeis (e suas respectivas ferramentas de suporte) deverão ser utilizados para organizar e estruturar as tarefas que serão desenvolvidas pela equipe.



Nome do projeto: Spark lms

Grupo: Kratos

Repositório do projeto: <https://github.com/GraceTorresLeite/Kratos>

Membros:

**George Maia, Grace Tôrres Leite, José Duarte,
Juliana Margato, Wesley da Silva Pierrotte, Xiaoying He**

1 - Análise de requisitos:

O projeto deve atender os critérios:

- Cadastrar usuário: o sistema deve permitir o usuário cadastrar-se no sistema habilitando os serviços propostos da plataforma comercial de livros;
- Realizar login de usuário: o sistema deve permitir o usuário realizar login no sistema garantindo a segurança e os registros individuais do suas solicitações;
- Solicitar empréstimo de livro: o sistema deve permitir ao usuário requisitar o empréstimo de livros pesquisado e selecionado na plataforma;
- Vender livros em formato digital: o sistema deve permitir ao usuário requisitar a comprar de livros pesquisado e selecionado na plataforma.



2 - Planejamento de Testes:

Este parágrafo descreve o planejamento de testes relatando os objetivos, as estratégias recomendadas, assim como as ferramentas utilizadas na implantação dos testes.

1. Objetivos:

a. Prototipação:

Este plano de testes suporta os seguintes caso de uso:

- i. Identificar o sistema que deve ser testado.
- ii. Listar os requisitos de teste recomendados.
- iii. Recomendar e descrever as estratégias de teste a serem empregadas.
- iv. Identificar os recursos requeridos de testes.
- v. Listar os elementos das tarefas de teste.

b. Escopo:

Esse projeto descreve os testes unitário e de integração , garantindo com que todas as funcionalidades tidas como critério de aceitação do sistema tenham uma boa cobertura.

As funcionalidades serão testadas:

- i. Cadastrar usuário
- ii. Realizar login
- iii. Selecionar livro pelo autor
- iv. Selecionar livro pela categoria
- v. Solicitar empréstimo e ou compra de livro

As interfaces cobertas pelo caso de teste:

- vi. Web - browser - Chrome
- vii. Mobile - sistema operacional android



2. Requisitos para o teste:

A lista a seguir apresenta os itens que foram identificados como alvos de testes a serem abordados na planilha de planejamento e execução.

a. Teste de Integridade dos Dados e do Banco de Dados:

Verificar a conexão com banco de dados.

Verificar o acesso ao banco de dados.

Verificar carga de acessos simultâneos

b. Teste de Função:

O sistema deve permitir ao usuário cadastrar-se.

O sistema deve permitir ao usuário realizar login.

O sistema deve permitir ao usuário buscar livros pelo nome.

O sistema deve permitir ao usuário buscar livros pelo nome do autor.

O sistema deve permitir ao usuário solicitar empréstimo de livro.

O sistema deve permitir ao usuário comprar um livro .

c. Teste de Ciclo de Negócio:

O desafio deste projeto prevê entregas semanais para cada tarefa

proposta, o ciclo completo será no término do mês com a

apresentação do projeto contemplando todos os exercícios executados com êxito.

d. Teste de Interface com Usuário:

Verificar a facilidade de navegação utilizando um conjunto de amostras de telas no desktop.

Verificar a facilidade de navegação utilizando um conjunto de amostras de telas no mobile.

e. Teste de Desempenho:

Nenhum.

f. Teste de Carga:

Verificar a capacidade de carga do banco de dados.

g. Teste de Estresse:

Nenhum.

h. Teste de Volume:

Nenhum.

i. Teste de Segurança e Controle de Acesso:

Verificação de usuário e permissões de acesso



- j. Teste de Failover / Recuperação:
Nenhum.
- k. Teste de Configuração:
Acessos das variáveis de sistema devidamente cadastrados para cada ferramenta deste projeto.
- l. Teste de Instalação:
Ambiente local

3. Estratégia do teste:

A estratégia de teste apresenta a abordagem recomendada pelo curso e mentoria realizada pela Gama Academy.

As principais considerações apontadas nesta seção, são as técnicas e as ferramentas a serem utilizadas e a orientação de validação.

a. Tipos de Teste:

i. Teste de Integridade dos Dados e do Banco de Dados:

O sistema de banco de dados deve ser testado em duas etapas, sem integração com as funcionalidades do sistema e outra com o fluxo completo (E2E).

- Objetivo do teste:

Garantir o funcionamento de todos os métodos de acesso ao banco de dados, como CRUD (Cadastrar, Consultar, Atualizar e Deletar) .

- Técnica:

Chamar cada método e listar os dados para verificar a mudança no banco de dados.

- Critério de conclusão:

Todos os processos e métodos de acesso ao banco de dados funcionam conforme projetado e sem nenhuma perda de dado.

ii. Teste de Função:

Os testes de função devem ser focados em requisitos e regras de negócio em casos de uso. Verificar a função interagindo com banco de dados por meio da GUI e código, analisando os resultados de saídas.



- **Objetivo:**
Assegurar a navegação correta do aplicativo, além da entrada, processamento e recuperação de dados.
- **Técnica:**
Executar cada caso de teste com dados válidos, inválidos e os devidos tratamentos de exceção.
Verificar se as saídas dos testes corresponderam ao resultado esperado.
- **Critério de conclusão:**
Todos os casos de testes são executados.
Todos os defeitos são encontrados e tratados.

iii. **Teste de Interface com Usuário:**

Os testes da interface com o usuário verificam a interação do sistema e identificam a facilidade e usabilidade do mesmo.

- **Objetivo:**
Verificar os fluxos de funcionamento com a integração do banco de dados, camadas de serviços e interface.
- **Técnica:**
Criar modelagens de testes contemplando o fluxo de navegação do sistema mobile e desktop.
- **Critério de conclusão:**
Todos os casos de testes são executados.
Todos defeitos são encontrados e tratados.

iv. **Teste de Carga:**

O teste de carga tem foco de testar a capacidade de carga do banco de dados de forma individual.

v. **Objetivo:**

Verificar a capacidade de carga e desempenho do banco de dados.

- **Técnica:**
Criar casos de teste com uma quantidade grande de entrada de dados para todos os métodos no banco de dados.



- Critério de conclusão:
Todos os casos de testes são executados.
Todos defeitos são encontrados e tratados.

b. Ferramentas:

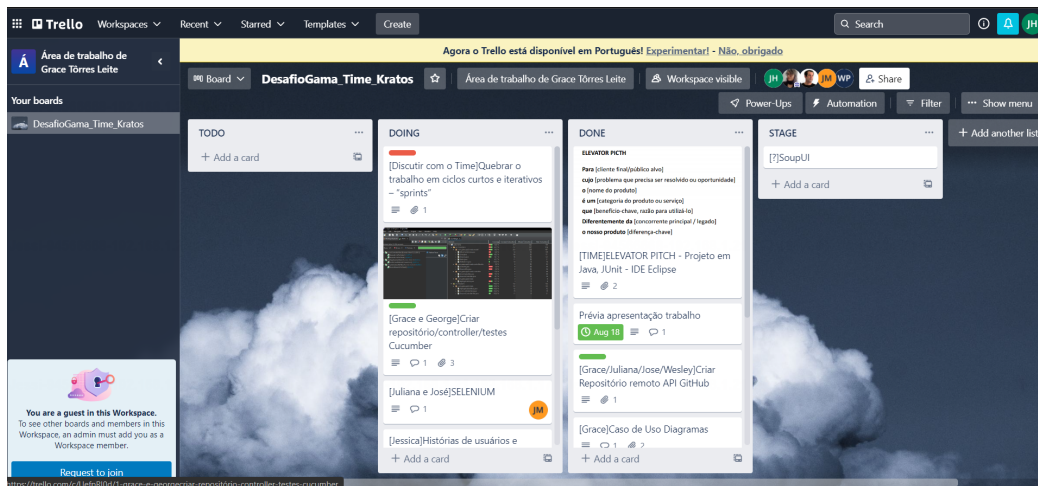
	Ferramenta	Versão
Gerenciamento de desenvolvimento	Trello	A definir
Linguagem dos testes	Java	11
Teste de função	Cucumber	7.4.1 4.0.0
Teste de interface com usuário	Selenium IDE/ WebDriver Appium	4.0.0 7.3.0
Teste de carga	Jmeter	5.5



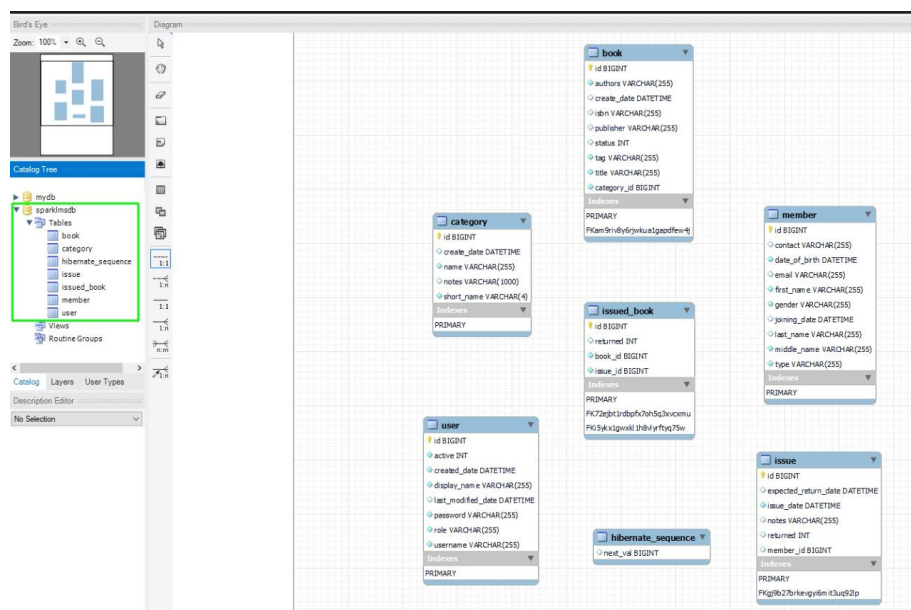
3 - Desenvolvimento:

Apresentação do progresso desde a criação até o momento atual do projeto:

1. Listar os requisitos do sistema.
2. Organizar e separar as tarefas no Trello.



3. Definir e montar o diagrama de classes:



4. Construir o banco de dados no MySQL baseado no UML.
5. Desenvolvimento de código e conexão com banco de dados baseado nos critérios de integração com a plataforma do springboot.
6. Criar casos de testes unitários e integração.
7. Executar casos de testes, identificar e tratar os erros encontrados.



4 - Cenários e Casos de teste:

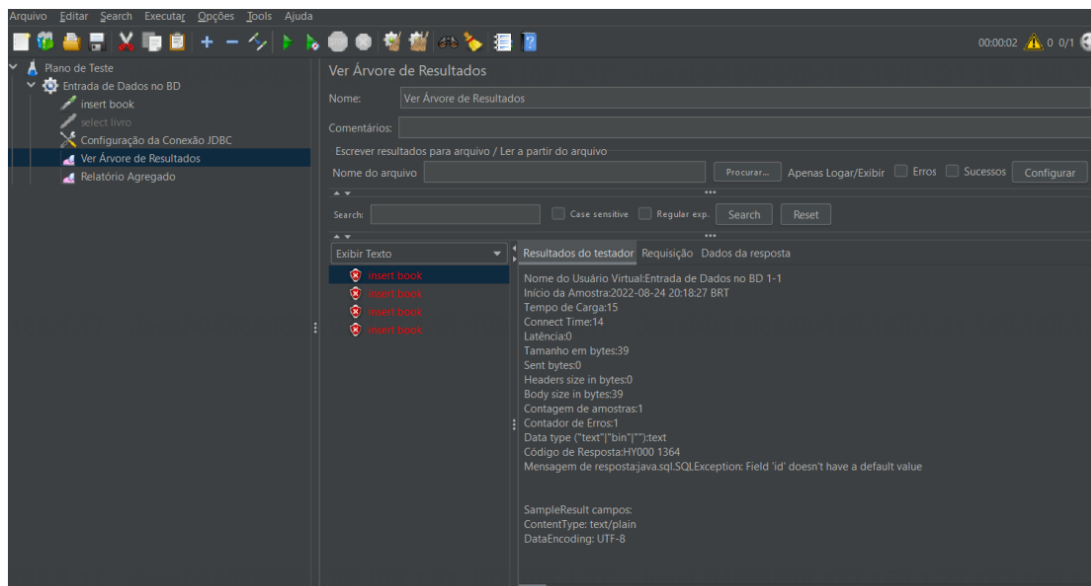
Cenário de teste 001:

Manipulando registros de livro no banco de dados.

Objetivo: Verificar a integridade de dados e a capacidade de carga do banco de dados

Caso de Teste 1:

- Tipo de teste: Teste de integridade de dados ao banco de dados.
- Ferramenta utilizada: Jmeter
- **Eu como** usuário do sistema
- **Quando** cadastrar um livro
- Dados de entradas:
400 de requisições ao método INSERT
- Resultado desejado:
Então o sistema deve salvar todos os dados de livros sem erro.
- Resultado atual:
Um erro “Field ‘id’ doesn’t have a default value” foi encontrado.



- Solução:
Alterado o campo ID como auto incremento.




Table Name: Schema: **sparkmsdb**

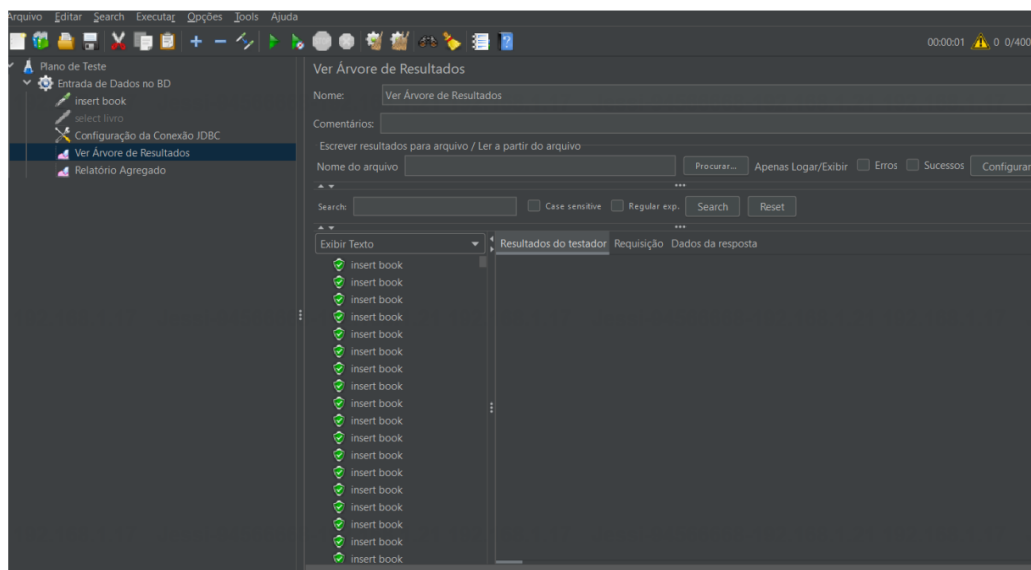
Charset/Collation: Engine: **MyISAM**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 authors	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 create_date	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 isbn	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 publisher	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

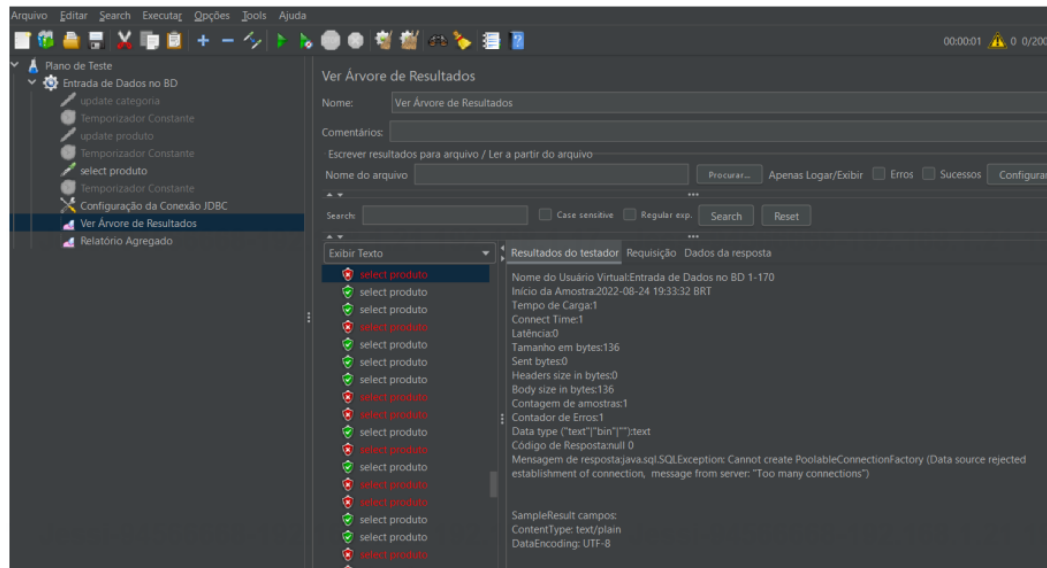
Column Name: Data Type:

- Testes após da correção:
Todos os testes foram realizados com sucesso.



Caso de Teste 2:

- Tipo de teste: Teste da capacidade de carga do banco de dados
- Ferramenta utilizada: Jmeter
Objetivo: Verificar a capacidade de carga do banco de dados.
Eu como usuário do sistema
Quando alterar dados de categorias e produtos e selecionar produtos
- Dados de entradas:
Mais 200 de requisições
- Resultado desejado:
Então o sistema deve alterar e retornar todos os dados de categorias e produtos sem erro.
- Resultado atual:
Um erro "Too many connections" foi encontrado.



- Solução:

Acessar o comando line do mysql Executar o comando show variables like “max_connections” para verificar o máximo de conexão que permite. Executar o comando set global max_connections = 500. Verificar se foi alterado utilizando o comando show variables like “max_connections”.

```
MySQL 8.0 Command Line Client
Server version: 8.0.29 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like "max_connections";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151 |
+-----+-----+
1 row in set (0.03 sec)

mysql> set global max_connections = 500;
Query OK, 0 rows affected (0.01 sec)

mysql> show variables like "max_connections";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 500 |
+-----+-----+
1 row in set (0.00 sec)
```

- Testes após da correção:

Após ajustes, o banco validou todas as requisições.



Cenário de teste 002:

Cadastro e pesquisa de livro, categoria, empréstimo de livro e devolução, assim como o cadastro de usuário e login.

Caso de Teste 1:

- Tipo de teste: Teste de funcionalidade para Desktop.
- Ferramenta utilizada: Selenium IDE

Objetivo: Verificar as interações com múltiplos navegadores e telas.

Eu como usuário do sistema

Quando manipular as requisições propostas

- Dados de entradas:
Cadastro de usuário, categoria, livro e acesso via login.
- Resultado desejado:

Então o sistema deve executar todos os casos de teste no Selenium IDE sem erro.

- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.

Project: Kratos_IDE*

Tests +

Search tests...

http://localhost:8080

Command	Target	Value
01_Login	/login	
02_Cadastro de Membro*	1280x775	
03_Cadastro de Categoria	name=username	
04_Cadastro Livro	name=username	admin
05_Cadastro de Empréstimo*	name=password	
06_Devolução de Um Livro	name=password	admin
	linkText=Log in	
	linkText=Mr. Admin	
	linkText=Log Out	

Command: open

Target: /login

Value:

Description:

Log Reference

11: click on id=select-all OK 19:43:43

12: click on css=returnBookChk OK 19:43:43

13: click on css=btn-primary OK 19:43:44

14: click on css=dataTables_empty OK 19:43:44

15: click on css=user-profile > span:nth-child(1) OK 19:43:46

16: click on linkText=Log Out OK 19:43:47

06_Devolução de Um Livro completed successfully 19:43:47



Caso de Teste 2:

- Tipo de teste: Teste de funcionalidade para Desktop.
- Ferramenta utilizada: Selenium IDE
Objetivo: Verificar a funcionalidade de cadastro de uma categoria
Eu como usuário do site
Quando cadastrar uma nova categoria
- Dados de entradas:
Nome de uma nova categoria
- Resultado desejado:
Então o sistema deve salvar a categoria sem erro.
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.

Caso de Teste 3:

- Tipo de teste: Teste de funcionalidade para Desktop.
- Ferramenta utilizada: Selenium IDE
Objetivo: Verificar a funcionalidade de cadastro de um livro
Eu como usuário do site
Quando cadastrar um novo livro
- Dados de entradas:
Os dados de um novo livro
- Resultado desejado:
Então o sistema deve salvar o livro sem erros.
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.



Caso de Teste 4:

- Tipo de teste: Teste de funcionalidade para Desktop.
- Ferramenta utilizada: Selenium IDE
Objetivo: Verificar a funcionalidade de solicitar um empréstimo
Eu como usuário do site
Quando solicitar um empréstimo de livro
- Dados de entradas:
Os dados requisitos a empréstimo de livro
- Resultado desejado:
Então o sistema deve salvar a solicitação de empréstimo sem erro.
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.

Caso de Teste 5:

- Tipo de teste: Teste de funcionalidade para Desktop.
- Ferramenta utilizada: Selenium IDE
Objetivo: Verificar a funcionalidade de devolver o livro
Eu como usuário do site
Quando devolver o livro
- Dados de entradas:
Selecionar o empréstimo a devolução
- Resultado desejado:
Então o sistema deve executar a devolução de livro sem erro.
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.



Caso de Teste 6:

- Tipo de teste: Teste de função
 - Ferramenta utilizada: Selenium WebDriver e Cucumber
- Objetivo:** Verificar a funcionalidade de selecionar um livro pelo autor
- Eu como** usuário do site
- Quando** selecionar um livro pelo autor
- Dados de entradas:
Nome de autor
 - Resultado desejado:
Então o sistema deve exibir os dados do livro com sucesso.
 - Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.

Caso de Teste 7:

- Tipo de teste: Teste de função
 - Ferramenta utilizada: Selenium WebDriver e Cucumber
- Objetivo:** Verificar a funcionalidade de selecionar um livro pela categoria
- Eu como** usuário do site
- Quando** selecionar um livro pela categoria
- Dados de entradas:
Nome da categoria
 - Resultado desejado:
Então o sistema deve exibir os dados do livro com sucesso.
 - Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.



```
1 package cucumber.steps;
2
3 import java.util.concurrent.TimeUnit;
13
14 public class CategoriaCucumber {
15
16     public static WebDriver obj;
17
18     @Given("Eu logada no site")
19     public void eu_logada_no_site() {
20         System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + ("\\driver\\chromedriver.exe"));
21         obj=new ChromeDriver();
22         obj.manage().window().maximize();
23         obj.get("http://localhost:8080");
24         obj.findElement(By.name("username")).sendKeys("admin");
25         obj.findElement(By.name("password")).sendKeys("admin");
26         obj.findElement(By.xpath("//*[@id='loginform']/div[3]/a")).click();
27     }
28     @When("Eu escolho o livro por categoria")
29     public void eu_escolho_o_livro_por_categoria() {
30         obj.manage().timeouts().implicitlyWait(30000, TimeUnit.MILLISECONDS);
31         obj.findElement(By.xpath("//a[contains(text(),'Dashboard')]")).click();
32         obj.findElement(By.xpath("//a[contains(text(),'Books')]")).click();
33         obj.findElement(By.xpath("//a[contains(text(),'List')]")).click();
34         obj.findElement(By.xpath("//input[@type='search']")).click();
35         obj.findElement(By.xpath("//input[@type='search']")).sendKeys("Testes automatizados de software");
36     }
37     @Then("tereí a visualização do livro desejado por categoria")
38     public void terei_a_visualizacao_do_livro_desejado_por_categoria() {
39         Assert.assertEquals("Testes automatizados de software", obj.findElement(By.xpath("//table[@id='datatable']/tbody/tr/td[3]")).getText());
40     }
41 }
42
43 }
```

Caso de Teste 8:

- Tipo de teste: Teste de funcionalidade para dispositivo mobile.
- Ferramenta utilizada: Appium
Objetivo: Verificar a funcionalidade de cadastro e login com sucesso.
- **Eu como** usuário do sistema pelo app baixado no celular
Quando inserir os dados para cadastro
E realizar o login
- Dados de entradas:
Dados válidos
- Resultado desejado:
Então o sistema deve habilitar sua entrada e navegação.
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.

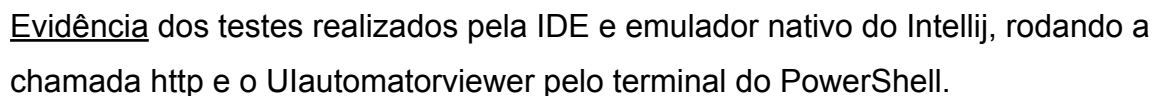


Caso de Teste 9:

- Tipo de teste: Teste de funcionalidade para dispositivo mobile.
- Ferramenta utilizada: Appium
- **Objetivo:** Verificar a funcionalidade de cadastro e login com erro na senha
- **Eu como** usuário do sistema pelo app baixado no celular
- **Quando** inserir os dados para cadastro
- **E** realizar o login
- Dados de entradas:
Dados inválidos na confirmação de senha
- Resultado desejado:
Então o sistema deve disparar uma mensagem de erro
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.

Caso de Teste 10:

- Tipo de teste: Teste de funcionalidade para dispositivo mobile.
- Ferramenta utilizada: Appium
- **Objetivo:** Verificar a funcionalidade de cadastro e login com erro no usuário
- **Eu como** usuário do sistema pelo app baixado no celular
- **E** já possuindo meus dados cadastrados anteriormente
- **Quando** inserir os dados para cadastro
- **E** realizar o login
- Dados de entradas:
Dados inválidos, duplicação no registro de usuário
- Resultado desejado:
Então o sistema deve disparar uma mensagem de erro
- Resultado atual:
Todos os testes foram realizados e nenhum erro foi encontrado.



O projeto iniciou com uma aplicação Rest simples em SpringBoot, integrando o banco nativo em H2.

Com a necessidade de implantar os testes de Selenium, adaptamos nosso planejamento para um sistema pronto disponibilizado no GitHub onde realizava a migração de banco com o MySql e a interface com a biblioteca do Thymeleaf.

Ao iniciar os testes com o Cucumber, identificamos uma documentação diferenciada para as aplicações em [SpringBoot](#) , o que despertou a necessidade de revisarmos a demanda na implantação de testes de integração com o uso da ferramenta. Resultando na implantação do Cucumber apenas nos testes envolvendo a camada de interface com o uso do Selenium WebDriver.