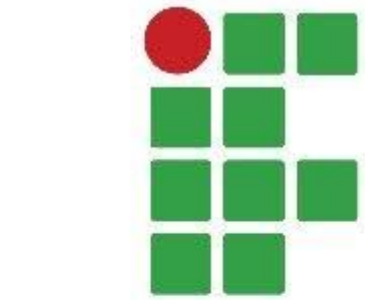

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Porto Alegre

Sistemas para Internet

Validação e Verificação de Sistemas

Alexandre Almeida
Sandro Lucas

Porto Alegre, 9 de Dezembro de 2019

1. Processo de Testes

O processo de testes será realizado através das seguintes ferramentas:

- PMD
- JUnit
- Jenkins

O processo ocorre na seguinte ordem:
É realizada a execução projeto através do Jenkins e durante a execução, ocorrem os testes PMD e JUnit, os quais, já estão configurados nas classes do projeto.

Ponto de Atenção: Sempre que houver uma alteração no código, deve ser gerado um relatório do PMD como primeiro passo para detectar problemas:

COMANDOS

Geração do relatório PMD com análise estática do código.
Comando 1: `mvn pmd:pmd`

Ponto de Atenção: Em caso de problemas, devem ser feitos os ajustes no código e o comando 1 deve ser repetido.

Se for necessário, o projeto Selenium, no navegador Firefox deve ser “regravado”.
O projeto no Selenium, deve fazer as seguintes ações:

- Inserir um atendente, um médico e um paciente
- Agendar uma consulta e cancelar a mesma
- Excluir o médico, paciente e atendente previamente cadastrados

Ponto de Atenção: Caso ocorram travamentos devido a eventos "mouseover" e"mouseout", estes eventos, bem como todos os dados inseridos, devem ser excluídos.
Se não ocorrerem problemas, o teste poderá ser finalmente exportado para o JUnit, o qual deve ser salvo no diretório: src/test/java/clinica/SeleniumJUnitTest.java.
Após isto, o arquivo deve ser modificado para comentar a linha que importa o ChromeDriver e inserir no método setUp() a linha:
Comando 2: `System.setProperty("webdriver.gecko.driver", "<raiz do projeto>/geckodriver");`

Como o teste do Selenium requer que o deploy do projeto já esteja finalizado, o primeiro "package" é feito no Maven sem executar os testes. Para isto, deve ser realizado o comando:
Comando 3: `mvn package -DskipTests`

Com o pacote WAR gerado em target/clinica.war, este é copiado para o diretório de aplicativos do Tomcat e o serviço é reiniciado:
Comando 4: `cp target/clinica.war /var/lib/tomcat9/webapps`
Comando 5: `service tomcat9 restart`

Para prosseguir com os testes, é necessário que o geckodriver esteja extraído no diretório raiz do projeto:
Comando 6: `tar -xf geckodriver-<versão>-linux64.tar.gz`

Feito isto, o "package" já pode ser feito novamente no Maven, mas desta vez sem deixar de executar os testes.
Comando 7: `mvn package`

No processo de empacotamento, devem ser executados dois testes automatizados por meio do JUnit, sendo eles:

1. Este deve envolver a manipulação direta do banco de dados (ClinicaTest.java), ocorrendo insert, update e delete de registros.
2. Este exportado do Selemium IDE, simula interações do usuário com o aplicativo via navegador (SeleniumJUnitTest.java).

Caso nenhum problema ocorra, o pacote WAR estará no mesmo local (target/clinica.war) e, novamente, o mesmo deverá ser copiado para o diretório de aplicativos do Tomcat e o serviço deverá ser reiniciado:

Comando 8: `cp target/clinica.war /var/lib/tomcat9/webapps`
Este deve fazer a cópia do arquivo war para a pasta do servidor tomcat.

Comando 9: `service tomcat9 restart`
Este deve fazer o restart do servidor tomcat, para que todo o cache seja apagado e as novas modificações sejam disponibilizadas.

2. Casos de Testes por Requisitos

Os requisitos funcionais são:

- Cadastrar, alterar e excluir atendentes
- Cadastrar, alterar e excluir médicos
- Cadastrar, alterar e excluir pacientes
- Marcar e cancelar consultas

O teste de manipulação direta do banco de dados (ClinicaTest.java) deve cadastrar os atendentes, médicos e pacientes e marca uma consulta.
No processo, há a simulação de um erro de digitação no nome de um médico. Em seguida, é simulada uma alteração para corrigir o erro.
Por fim, o teste exclui todos os dados que inseriu. Desta forma, o teste abrange a marcação e o cancelamento de consultas a inserção e exclusão de atendentes, médicos e pacientes, apesar de abranger apenas uma alteração de médico.

O teste de interação do usuário (SeleniumJUnitTest.java) insere um atendente, um médico e um paciente, além disso, marca e cancela uma consulta, também exclui o paciente, o médico e atendente inseridos previamente.

3. Requisitos de hardware e Software

Os testes são feitos em uma máquina virtual com 3 GB de memória RAM e 10 GB de armazenamento.
Os requisitos de software são:

- **Sistema operacional Lubuntu 19.04**
- **Apache HTTP Server**
- **Servidor Apache Tomcat 9**
- **Banco de Dados MySQL**
- **Apache Maven**
- **Plugin do JUnit para o Maven**
- **Plugin do PMD para o Maven**
- **Plugin do Selenium para o Maven**
- **Navegador Firefox (Versão Homologada: 70.0.1)**
- **Extensão Selenium IDE para o Firefox**
- **geckodriver**
- **Jenkins**

4. Procedimento de registro de testes

Durante a execução dos testes, as descrições de eventuais bugs podem ser localizadas em um arquivo de texto puro e também na área de log do Jenkins.
Após as correções, os testes são repetidos e os bugs no mesmo arquivo marcados como resolvidos e o Jenkins irá gerar um novo log.
A análise posterior do arquivo serve como base para a criação de novos testes.

Além disso, também é possível diversas ações pós-build através do Jenkins, como por exemplo, realizar o envio do log via e-mail.
Obs.: Neste projeto não foi configurado o e-mail pós-build, devido a restrições no Firewall do IFRS.