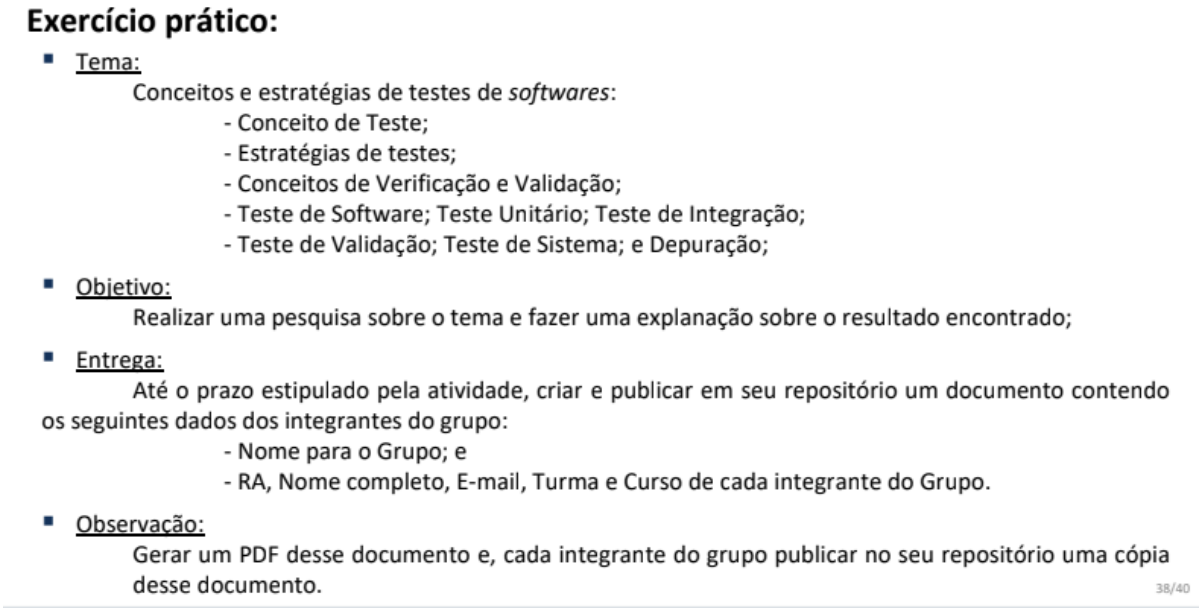
USJT - UC Gestão de Qualidade de Software - Turma: ADS1AN-BUC1-6272430

RA 823159742 - Igor Cordeiro de Souza Pereira - 823159742@ulife.com.br
RA 823217461 - Lucca Palmieri Dittrich - 823217561@ulife.com.br
RA 823123930 - Eduardo Vieira de Jesus - 823123930@ulife.com.br
RA 82426451 - Eduardo Filipe Silva S. Santos - 82426451@ulife.com.br

USJT-GQS-ADS1AN-BUC1-6272430-Atividade da Prática 03



Exercício prático:

- Tema:
  Conceitos e estratégias de testes de *softwares*:
  - Conceito de Teste;
  - Estratégias de testes;
  - Conceitos de Verificação e Validação;
  - Teste de Software; Teste Unitário; Teste de Integração;
  - Teste de Validação; Teste de Sistema; e Depuração;
- Objetivo:
  Realizar uma pesquisa sobre o tema e fazer uma explanação sobre o resultado encontrado;
- Entrega:
  Até o prazo estipulado pela atividade, criar e publicar em seu repositório um documento contendo os seguintes dados dos integrantes do grupo:
  - Nome para o Grupo; e
  - RA, Nome completo, E-mail, Turma e Curso de cada integrante do Grupo.
- Observação:
  Gerar um PDF desse documento e, cada integrante do grupo publicar no seu repositório uma cópia desse documento.

38/40

Conceitos e estratégias de testes de softwares

Conceito de teste

Testar software significa conduzir um conjunto de atividades que garantem que o sistema funcione conforme esperado. Seu objetivo é encontrar defeitos no código antes que o software seja entregue ao cliente.

Estratégia de testes

A estratégia de teste define um roteiro para testar o software. Ela envolve planejamento de testes, execução de casos de teste, coleta e avaliação de dados. Essa estratégia deve ser flexível, mas com um planejamento razoável para garantir o sucesso do projeto.

Conceitos de Verificação e Validação

Verificação assegura que o software foi construído corretamente de acordo com as especificações. Validação garante que o software atenda às necessidades e expectativas do cliente. Em outras palavras, verificação responde à pergunta "Estamos construindo o produto certo?" e validação, "Estamos construindo o produto certo?".

Teste de Software

O teste de software é a última defesa contra erros que podem ter passado despercebidos durante o desenvolvimento. Ele permite a descoberta de defeitos e garante que o software esteja funcionando de acordo com os requisitos.

Teste unitário

O teste unitário verifica partes individuais (unidades) do código para garantir que funcionem isoladamente. Geralmente é conduzido pelo próprio desenvolvedor durante a fase de codificação.

Teste de integração

Esse teste verifica a interação entre os diferentes módulos ou componentes do sistema, assegurando que eles funcionem corretamente em conjunto.

Teste de validação

Valida se o sistema completo cumpre os requisitos especificados e funciona corretamente em um ambiente real de uso. É focado em testar as funcionalidades visíveis para o usuário.

Teste de sistema

O teste de sistema envolve testar o software como um todo integrado com outros componentes do sistema (hardware, pessoas, etc.). Ele verifica se todos os elementos do sistema funcionam corretamente em conjunto.

Depuração

A depuração é o processo de identificar, diagnosticar e corrigir os defeitos descobertos durante o teste. Embora possa ser planejada, é considerada uma "arte", pois conectar um sintoma ao erro raiz requer experiência e intuição.

Exemplo de teste

Engenharia do caos

Conforme os softwares evoluíram e se tornaram mais complexos, diferentes metodologias e tipos de testes foram desenvolvidos, conforme os requisitos de qualidades esperados para cada projeto. Em ambientes modernos que hospedam aplicações com arquitetura de microservices em cloud multi-Região, ou ainda multi-cloud, é esperado que a aplicação continue com seus componentes

principais funcionando, mesmo se houver problemas no ambiente de produção em algum componente ou região cloud. A metodologia da engenharia do caso nasce da necessidade de testar as aplicações que devem continuar a operar, ao menos parcialmente, mesmo com problemas em componentes e falhas de infraestrutura, para isso realiza- se experimentos para identificar falhas e compreender a causa raiz, ajudar a evitá-las e manter o sistema resiliente.

Outros tipos de testes

Teste de Cobertura

O teste de cobertura consiste em verificar se todas as linhas do código estão sendo usadas, o que implica nas condições (if-else), nos loops e métodos auxiliares. Visa atingir 100% do código.
Caso o teste esteja sendo usado parcialmente, deverá ser corrigido para evitar códigos “fantasmas” ou inúteis.

Técnica Estrutural (ou teste caixa-branca)

Segundo Pressman, o teste da caixa-branca avalia o comportamento interno do software, analisando os seguintes aspectos:
- teste de condição
- teste de fluxo de dados
- teste de ciclos
- teste de caminhos lógicos

Um exemplo de aplicação prática está no uso de JUnit para testar métodos e classes feitos em Java
A técnica de testes de caixa-branca está associada aos testes de integração, validação e sistema.

Teste Funcional (ou teste caixa-preta)

O teste da caixa preta já é mais simples, não considera o comportamento interno do código e sim os dados de entrada e saída.
Os testes são realizados e depois comparados a um resultado esperado, caso seja igual, o teste será um sucesso, do contrário, terá uma falha.
A técnica de de testes de caixa-preta está associada aos testes de integração, cobertura e unitário.

Referências:

PRINCIPLES OF CHAOS ENGINEERING. Disponivel em: <https://principlesofchaos.org/>. Acesso em: 18 sep. 2024.

Resilience engineering: Where do I start? Disponivel em: <https://www.resilience-engineering-association.org/resources/where-do-i-start/>. Acesso em: 18 sep. 2024.

PRESSMAN, R. S. Software Engineering: A Practitioner's Approach. 7. ed. New York: McGraw-Hill, 2010. cap. 17, p. 449.

Dias Neto, A. C. (s.d.). *Introdução a Teste de Software*. Recuperado de Universidade de São Paulo: https://edisciplinas.usp.br/pluginfile.php/3503764/mod_resource/content/3/Introducao_a_Teste_de_Software.pdf