4. Tipos de Testes e Documentação

- Escreva um documento explicando:
- Diferença entre teste funcional, teste de regressão, teste de integração e teste de performance.
- Exemplos de quando aplicar cada tipo no contexto do BugBank.
- Entregável: Documento (Markdown, Word ou PDF).

Tipos de Testes de Software e Aplicação no Contexto BugBank

Este documento detalha as diferenças entre quatro tipos essenciais de testes e ilustra sua aplicação prática no sistema bancário simulado do BugBank.

1. Teste Funcional (Functional Testing)

O Teste Funcional verifica se cada função da aplicação opera de acordo com os requisitos e especificações do negócio. O foco é no *o quê* o sistema faz, testando a experiência do usuário (end-to-end) sem se preocupar com o código-fonte (caixa-preta).

Característica	Detalhe
Objetivo	Validar se o sistema atende às necessidades do usuário e às regras de negócio.
Foco	Ações do usuário, fluxos de tela e resultados observáveis.

Exemplo de Aplicação no BugBank

- **Cenário:** O usuário tenta criar uma conta com um e-mail já existente.
- **Ação:** Verificar se o sistema **bloqueia** o cadastro e exibe a mensagem de erro correta ("E-mail já cadastrado").
- Os testes de **Criação de Conta** e **Login** (válido e inválido) feitos na Etapa 1 são exemplos clássicos de Testes Funcionais.

2. Teste de Regressão (Regression Testing)

O Teste de Regressão é a reexecução de testes funcionais (e não funcionais) previamente bem-sucedidos para garantir que as alterações recentes no código (correções de bugs, novas *features*, atualizações de sistema) não introduziram novos defeitos ou quebraram funcionalidades existentes.

Característica	Detalhe
Objetivo	Garantir que o sistema não regrediu após uma modificação.
Foco	Funcionalidades previamente estáveis e áreas de alto risco.

Exemplo de Aplicação no BugBank

- **Cenário:** A equipe corrige o bug da **animação de rotação 3D** do formulário de Cadastro.
- **Ação:** Rodar novamente o teste automatizado de **Login com Sucesso** (Cenário 2) para confirmar que, mesmo após a correção do CSS/animação, a funcionalidade de Login (que já funcionava) não foi quebrada.
- Os testes de Cypress automatizados (Etapa 2) são perfeitos para formar a **suíte de regressão**.

3. Teste de Integração (Integration Testing)

O Teste de Integração verifica se módulos, componentes ou sistemas diferentes (como o front-end e o back-end, ou dois serviços distintos) trabalham juntos conforme o esperado. O foco é na comunicação, interfaces e fluxo de dados entre as partes.

Característica	Detalhe
Objetivo	Validar se as interfaces e a transferência de dados entre módulos são íntegras.
Foco	Comunicação entre subsistemas, APIs e bancos de dados.

Exemplo de Aplicação no BugBank

- **Cenário:** Um usuário realiza uma **Transferência de Valores**.
- **Ação:** Verificar se a requisição de transferência enviada pelo **Front-end** (aplicação web) é processada corretamente pelo **Back-end/API** e se o **Saldo** é subtraído da conta de origem e *adicionado* à conta de destino.
- **O que faríamos:** Usaríamos o **Postman** (Etapa 3) para testar o endpoint de `POST /transactions` diretamente e, em seguida, faríamos requisições `GET /balance` para ambas as contas para garantir que a integração do serviço de transação com o serviço de saldo ocorreu corretamente.

4. Teste de Performance (Performance Testing)

O Teste de Performance verifica a velocidade, escalabilidade e estabilidade de uma aplicação sob cargas de trabalho específicas. O objetivo é garantir que o sistema lide bem com o uso esperado (e inesperado).

Característica	Detalhe
Objetivo	Medir a capacidade de resposta, a taxa de transferência e o uso de recursos sob carga.
Foco	Velocidade de carregamento, picos de usuários (Load Test), e volume de dados.

Exemplo de Aplicação no BugBank

- **Cenário:** Dia de pico (ex: 5º dia útil) com milhares de transferências e logins simultâneos.
- **Ação:** Usar ferramentas como JMeter ou k6 para simular **1.000 usuários** tentando fazer login e **100 transferências por segundo**.
- **Objetivo:** Garantir que o tempo de resposta do endpoint de `/login` e `/transfer` permaneça abaixo de **1 segundo** e que o servidor não falhe ou retorne erros `500` (Erro Interno do Servidor) sob essa carga.