Documento de testes e aplicação.

Integrantes desse trabalho:
Matheus Barbosa 1262223635,
Thiago pralon 1262217078,
Felipe Neves 1262225380,
Paulo Henrique 1262221832.

Índice.

1. Contratada e contratante.

Empresa contratante: InovaBiz.
Integrantes:
Jean Ferreira 1262216969.
Vitor Antônio 12622110215.
Widney Barreto 1262218394.
Empresa contratada: Nexa Enterprise.
Integrantes:
Matheus Barbosa.
Thiago pralon.
Felipe Neves.
Paulo Henrique.

2. Testes a serem aplicados.

Teste de unidade: Verificar se unidades individuais do código (funções, métodos) funcionam
conforme o esperado.
Teste de integração: Garantir que diferentes partes do sistema funcionem corretamente quando integradas.
Teste de sistemas: Verificar se o sistema inteiro atende aos requisitos especificados.
Teste de aceitação:Garantir que o sistema atenda aos requisitos do usuário.
Testes funcionais: Verificar se as funções específicas do sistema estão operando corretamente, como testes de entrada e saída, teste do menu de opções, teste de cálculos e tratamento de erros.
Teste de regressão: Garantir que as alterações recentes não quebraram funcionalidades existentes.

Teste de desempenho: Avaliar o desempenho e a eficiência do sistema em diferentes condições.
Teste de usabilidade: Avaliar a facilidade de uso e experiência do usuário.
Teste de segurança: Identificar vulnerabilidades de segurança no código e na infraestrutura.
Teste de manutenção: Avaliar a facilidade de manutenção do código.

- ## 2. Resultados da aplicações.

Teste de unidade: Testamos a função cadastrarContato no ContatoController para garantir que um contato seja cadastrado. Assim funcionando corretamente.
Passo a passo:

1. Criamos um caso de teste isolado para a função cadastrarContato.
2. Utilizamos o framework JUnit.
3. Definimos entradas de teste, simulando a entrada do usuário.
4. Executamos a função ContatoController e verificamos se o contato foi cadastrado com sucesso no banco de dados.

Teste de integração: Testamos a integração entre ContatoController e ContatoRepository para garantir que as operações de cadastro e visualização de contatos funcionem. Nos quais tiveram êxito.
Passo a passo:

1. Criamos casos de teste que envolvem tanto o cadastro quanto a visualização de contatos.
2. Testamos se o fluxo completo, desde a criação do contato até a visualização, ocorre sem problemas.
3. Verificamos se os dados apresentados na visualização correspondem aos dados cadastrados. No caso sim

Teste de sistemas:O teste de sistema abordou a aplicação como um todo, garantindo que todas as partes funcionassem bem em conjunto. Incluiu testes funcionais de todas as operações, desde o cadastro até a exclusão de contatos, e verificou a integração com o banco de dados. No geral em questão de sistemas, funciona corretamente.

Passo a passo:

1. Execução de casos de teste para cada operação do sistema.
2. Verificação da integridade dos dados no banco de dados após operações de CRUD.
3. Testes simultâneos de operações para garantir que a aplicação manteria o estado correto.

4. Identificação e resolução de quaisquer conflitos ou problemas de integração.

Testes funcionais: Os testes funcionais foram divididos em várias categorias, incluindo menu de opções, entrada e saída, cálculos e tratamento de erros. Cada categoria foi examinada individualmente para garantir que a aplicação funcionasse conforme esperado. E em todas as análises foi constatado secesso.

Passo a passo:

1. Menu de Opções:

- Verificação das opções disponíveis no menu.
- Execução de cada opção para garantir que as funcionalidades correspondiam às descrições.
- Teste da opção de sair para encerrar a aplicação corretamente.

2. Entrada e Saída:

Este requisito funcional

- Inserção de diferentes tipos de dados para garantir que a aplicação tratasse adequadamente cada entrada.
- Verificação das mensagens de saída para garantir clareza e informatividade.
- Testes para garantir que a entrada de dados correta resultasse em saída apropriada.

3. Cálculos:

- Verificação dos cálculos internos, como o processo de atualização de informações de contato no banco de dados.
- Testes com diferentes valores para garantir que os cálculos produzissem resultados corretos.

4. Tratamento de erros./ entrada e saída.

grupo B identificou um erro no caso de tratamento de erros, então implementamos solução e testes específicos para essa questão. Deixamos o código mais robusto para lidar melhor com entradas e saídas de dados inválidas. Onde já se encontra no git hub. Abaixo estão algumas correções.

- Introdução de erros intencionais, como dados inválidos, para verificar se a aplicação lidava com eles adequadamente na questão de entrada e saída.
- Avaliação das mensagens de erro para clareza e orientação sobre como corrigir problemas.
- Testes para garantir que a entrada de dados correta resultasse em saída apropriada.

Testes regressão: Realizamos um teste de regressão após uma atualização no código para garantir que as funcionalidades existentes não foram afetadas. E após alguns teste foram foi garantido o funcionamento desse requisito.

Passo a passo:

1. Identificamos as funcionalidades afetadas pela atualização.
2. Executamos casos de teste existentes para essas funcionalidades.
3. Verificamos se as funcionalidades estão intactas após a atualização.

Teste de desempenho: Testamos o desempenho da aplicação ao cadastrar e visualizar uma grande quantidade de contatos simultaneamente.

Passo a passo:

1. Criamos diversos cadastros no código pelo eclipse.
2. E depois fomos testando puxando os cadastros já feito inúmeras vezes.
3. E em todas funcionou com excelência.

Teste de usabilidade: O teste de usabilidade concentrou-se na experiência do usuário ao interagir com a aplicação. Garante que a interface fosse intuitiva e que os usuários pudessem realizar tarefas com facilidade. Utilizei testadores fictícios para

simular diferentes perfis de usuários. De forma geral tudo funciona com uma interface simples e uma usabilidade fácil para diversos tipos de pessoas

Passo a passo:

1. Avaliação da clareza e eficácia do menu de opções.
2. Testes com usuários fictícios, observando a facilidade de cadastro, edição e exclusão de contatos.
3. Verificação da navegabilidade e da lógica por trás das opções do menu.
4. Coleta de feedback sobre a experiência do usuário e ajustes conforme necessário.

Teste de segurança: o teste de segurança foi realizado com propósito de localizar algum erro desse fator, mas utilizando o owasp zap para realizar varreduras foi constatado que o código não possui qualquer problema no setor segurança.

Passo a passo:

1. Verificação da segurança das conexões com o banco de dados.
2. Revisão das permissões de acesso aos dados de contato.
3. Depois de varreduras com o softer não foi indentificado qualquer problema nesse setor.

Teste de manutenção: O teste de manutenção visou garantir que futuras atualizações e modificações pudessem ser realizadas sem impactos negativos. Isso envolveu revisar a estrutura do código, a clareza da documentação e a facilidade de incorporar novos recursos. Sendo muito bem sucedido.

Passo a passo:

1. Revisão da documentação para garantir clareza e detalhes sobre a estrutura do código.
2. Implementação de uma pequena atualização fictícia, como a adição de um novo campo de contato. (mostrou um bom funcionamento)

3. Verificação da facilidade de incorporar a atualização sem afetar negativamente as funcionalidades existentes.