UNIVERSIDADE SÃO JUDAS TADEU

SISTEMAS DE INFORMAÇÃO

GESTÃO E QUALIDADE DE SOFTWARE

JOÃO VITOR GOMES PEREIRA - 82329432

MATEUS HENRIQUE SALVADOR - 82323463

FELIPE CARDOSO SILVA – 82326693

ATIVIDADE DE AULA

PROF.º DOCENTE – ROBSON CALVETTI

SÃO PAULO – SP

2025

SUMÁRIO

PLANO DE TESTES

Exercício 1

Introdução com identificação do projeto

Projeto: Testes estruturais para o método de busca binária em Java.

Função alvo: public static int busca_binaria(int[] iVet, int iK)

Descrição: Método iterativo que recebe um vetor ordenado de inteiros e retorna o índice do valor iK se encontrado, ou -1 caso contrário.

Escopo

O presente plano de testes abrange a verificação funcional da implementação do algoritmo de busca binária em um vetor de inteiros ordenado de forma crescente. O escopo se limita à validação da lógica da função busca_binaria, escrita em Java, garantindo que o algoritmo identifique corretamente a posição de um elemento no vetor ou retorne -1 caso o elemento não esteja presente.

Serão considerados os seguintes aspectos dentro do escopo:

- Testes funcionais caixa-branca, com base na análise da complexidade ciclomática e caminhos independentes do algoritmo.
- Testes com vetores de diferentes tamanhos (inclusive vetores vazios e com apenas um elemento).
- Testes com valores presentes e ausentes no vetor.
- Verificação dos valores de retorno esperados conforme diferentes cenários de entrada.
- Validação de tipos de dados diferentes de inteiros.

Objetivos

O objetivo principal deste plano de testes é garantir que a função busca binária, desenvolvida em Java, funcione corretamente sob diversas condições de entrada, assegurando a confiabilidade e robustez do algoritmo. Especificamente, busca-se:

- Verificar a exatidão do algoritmo na identificação da posição de um valor (chave) dentro de um vetor ordenado de inteiros,
- Garantir que o algoritmo retorne -1 de forma apropriada quando a chave de busca não estiver presente no vetor.
- Cobrir todos os caminhos lógicos independentes do código
- Validar o comportamento da função frente a diferentes tamanhos de entrada
- Assegurar a integridade dos limites de busca dentro do vetor
- Testar o algoritmo com entradas variadas
- Preparar a função para integração segura com outros componentes do sistema

Esses objetivos visam não apenas detectar falhas, mas também assegurar que o comportamento da função esteja de acordo com as expectativas de um algoritmo de busca binária eficiente e correto.

Requisitos

Nome	Requisito
R1	O método deve retornar o índice correto se o valor `iK` estiver presente.
R2	O método deve retornar `-1` se o valor `iK` não estiver presente.
R3	O método deve funcionar apenas com vetores ordenados de forma crescente.

Estratégias, tipos de testes e ferramentas a serem utilizadas

Para garantir a qualidade da função busca binária, será adotada uma estratégia de testes de caixa branca, com foco na análise estrutural do código-fonte. A principal abordagem será a técnica de teste de caminho básico, utilizando a complexidade ciclomática como base para identificar os caminhos lógicos independentes do algoritmo e garantir sua cobertura completa.

Tipos de Teste a Serem Aplicados:

- Teste de Unidade: A função será testada isoladamente para verificar se produz a saída esperada para diferentes conjuntos de entrada. Esse teste visa detectar erros no comportamento lógico do algoritmo.
- Teste de Caminho: Serão mapeados os caminhos lógicos existentes na função (com base no grafo de fluxo de controle) e elaborados casos de teste que garantam a execução de cada um desses caminhos pelo menos uma vez.
- Teste de Limite: Serão criados casos que testam os limites inferiores e superiores do vetor, bem como situações com vetor vazio, vetor com um único elemento e situações de chave ausente.
- Teste de Condições de Decisão: As condições iK < iVet[iMeio], iK > iVet[iMeio] e iK == iVet[iMeio] serão testadas isoladamente para assegurar que todas as decisões possíveis dentro do laço while são executadas.

Ferramentas a Serem Utilizadas:

- IDE: Eclipse ou IntelliJ IDEA – para desenvolvimento e execução da função.
- JUnit – para implementação e automação dos testes unitários.
- Diagramas de fluxo (draw.io / Lucidchart) – para representação do grafo de fluxo de controle.
- Planilha Excel – para documentação e rastreamento dos casos de teste.

Recursos a serem empregado

Recurso	Descrição detalhada
Hardware	Computador com processador Intel ou AMD (mínimo dual-core), 4 GB de RAM, HD/SSD.
Software	Ambiente de desenvolvimento Java (IDE como Eclipse ou IntelliJ) e JDK 8+ instalado.
Linguagem	Código-fonte implementado em Java, versão 8 ou superior.
Ferramentas de apoio	JUnit para testes automatizados.
Testador	Profissional responsável pela criação, execução e documentação dos testes.
Ambiente de Execução	Sistema operacional Windows ou Linux, com suporte a execução de testes locais.
Controle de versão (opcional)	Utilização do Git/GitHub para versionamento e registro do progresso dos testes.

Cronograma das atividades

Etapa	Descrição da Atividade	Data Estimada
Análise do Código	Estudo detalhado da lógica da função `busca binária` e desenho do fluxo de controle.	[DATA]
Definição da Estratégia de Teste	Escolha dos tipos de teste, técnicas e ferramentas a serem aplicadas.	[DATA]
Elaboração do Plano de Teste	Produção deste documento, contendo escopo, objetivos, tabelas e plano de execução.	[DATA]
Elaboração dos Casos de Teste	Criação dos casos cobrindo caminhos independentes e condições limites.	[DATA]
Implementação e Execução dos Testes	Execução prática na IDE com monitoramento da cobertura.	[DATA]
Análise de Resultados	Comparação entre os resultados esperados e os obtidos; geração de relatório.	[DATA]

Revisão e Entrega Final	Consolidação de resultados e entrega do material final conforme exigido pelo professor.	[DATA]

* Como este plano de testes é baseado em um caso de uso acadêmico, e não em um sistema real implantado, optamos por manter as datas em aberto.

Exercício 2

Introdução com identificação do projeto

Projeto: Testes funcionais para o caso de uso “Login com validação em duas etapas”.

Função alvo: Processo completo de autenticação do usuário no sistema com validação de login, senha e código SMS.

Descrição: O fluxo de login descrito é composto por duas etapas: a primeira é a verificação do login e senha inseridos pelo usuário, e a segunda consiste na digitação de um código enviado por SMS. O acesso ao sistema só é liberado após ambas as etapas serem validadas com sucesso.

Escopo

O presente plano de testes abrange a verificação funcional do processo de login com autenticação em duas etapas, conforme os requisitos identificados no caso de uso. Este processo se aplica ao módulo de segurança de acesso de um sistema e está focado em garantir a correta validação das credenciais do usuário e do código de autenticação enviado via SMS.

Serão considerados os seguintes aspectos dentro do escopo:

- Verificação da entrada de login e senha com diferentes combinações válidas e inválidas.
- Validação do processo de geração e envio do código de autenticação.
- Testes de entrada correta e incorreta do código de verificação.
- Tratamento de erros e mensagens apresentadas ao usuário.
- Garantia de fluxo seguro e funcional até a liberação do sistema.
- Testes de comportamento quando o número de telefone não está cadastrado.

Objetivos

O principal objetivo deste plano de testes é assegurar que o sistema realize a autenticação em duas etapas de forma segura e eficaz, garantindo o acesso apenas a usuários válidos.

Especificamente, busca-se:

- Validar a entrada de login e senha com diferentes tipos de dados e formatos.
- Garantir que usuários inválidos recebam a mensagem correta e não avancem para a próxima etapa.
- Confirmar a geração de código dinâmico e o envio correto por SMS.
- Garantir que apenas códigos válidos autorizem o login.
- Verificar mensagens de erro e sucesso apresentadas ao usuário.
- Simular cenários de falhas e validar o comportamento do sistema.
- Garantir segurança e consistência na autenticação.

Requisitos

Nome	Requisito
R1	O sistema deve aceitar login e senha digitados pelo usuário.
R2	O sistema deve validar os dados com as credenciais armazenadas no banco de dados.
R3	Se o login e/ou a senha estiverem incorretos, deve apresentar a mensagem “Login e/ou Senha incorretos”.
R4	Se o login e a senha estiverem corretos, o sistema deve gerar um código de verificação de duas etapas.
R5	O código deve ser enviado por SMS ao número previamente cadastrado do usuário.
R6	O sistema deve permitir a entrada do código recebido.
R7	Caso o código digitado não coincida com o gerado, apresentar a mensagem “Login não autorizado!”.
R8	Se o código estiver correto, apresentar “Login realizado com sucesso” e liberar o acesso ao sistema.

Estratégias, tipos de testes e ferramentas a serem utilizadas

Para garantir a confiabilidade e robustez do processo de login em duas etapas, serão adotadas as seguintes estratégias e tipos de teste:

Tipos de Teste a Serem Aplicados:

- Teste Funcional (Caixa-Preta): Verificação de acordo com os requisitos e o comportamento esperado do sistema, sem considerar a lógica interna do código.
- Teste de Fluxo Alternativo: Cobertura de cenários com dados incorretos ou ausentes, em cada uma das etapas.
- Teste de Mensagens e Feedback: Validação das mensagens exibidas ao usuário de acordo com o tipo de falha.
- Teste de Integração (Login + SMS): Confirmação de que o envio do código por SMS está integrado ao fluxo de login.
- Teste de Segurança: Verificação de que sem a digitação correta do código, não é possível acessar o sistema.

Ferramentas a Serem Utilizadas:

- IDE (Eclipse / IntelliJ / Visual Studio Code) – para testes com protótipos ou aplicação em desenvolvimento.
- Postman ou simuladores de API – para testar o envio de SMS e respostas de código.
- Planilha Excel ou Google Sheets – para rastreamento dos casos de teste e registro dos resultados.
- Frameworks de Teste – JUnit (Java) ou pytest (Python), dependendo da linguagem utilizada.
- Mock de serviço de SMS – caso a API real não esteja disponível para testes.

Recursos a serem empregados

Recurso	Descrição detalhada
Hardware	Computador com mínimo de 4 GB de RAM, acesso à internet e navegador atualizado.
Software	Sistema web ou aplicativo com login implementado, banco de dados com usuários cadastrados, e serviço de envio de SMS configurado.
Linguagem	Backend em Java, Python, Node.js ou outro compatível com o sistema de autenticação.
Ferramentas de apoio	Ferramentas de mock para envio de SMS, e ferramentas de testes funcionais.
Testador	Profissional de QA responsável pela elaboração e execução dos testes.
Ambiente de Execução	Ambiente de homologação (ambiente de testes) com dados simulados de usuários.
Controle de versão (opcional)	GitHub ou GitLab para versionamento de scripts e evidências de teste.

Cronograma de Atividades

Etapa	Descrição da Atividade	Data Estimada
Levantamento de Requisitos	Análise do fluxo de login e das regras de autenticação	[DATA]
Definição da Estratégia de Teste	Escolha de técnicas, ferramentas e tipos de teste	[DATA]
Elaboração do Plano de Teste	Criação do documento completo com escopo, objetivos e plano	[DATA]
Modelagem dos Casos de Teste	Criação de cenários com diferentes entradas e respostas esperadas	[DATA]
Execução dos Testes	Execução prática no ambiente de homologação	[DATA]
Análise dos Resultados	Comparação dos resultados esperados com os reais e documentação	[DATA]
Ajustes e Reteste	Correção de falhas (se houver) e reexecução dos testes afetados	[DATA]

Revisão e Entrega Final	Revisão técnica do plano e envio final da documentação	[DATA]

*Como este plano de testes é baseado em um caso de uso acadêmico, e não em um sistema real implantado, optamos por manter as datas em aberto.

ROTEIROS DE TESTES

Exercício 1

Pré-condição:

O vetor deve estar ordenado em ordem crescente.

Testes Funcionais

1. Elemento no meio do vetor

- Entrada: iVet = [1, 3, 5, 7, 9], iK = 5
- Esperado: 2 (índice do 5)

2. Elemento na primeira posição

- Entrada: iVet = [10, 20, 30, 40], iK = 10
- Esperado: 0

3. Elemento na última posição

- Entrada: iVet = [5, 15, 25, 35, 45], iK = 45
- Esperado: 4

4. Elemento que não existe no vetor

- Entrada: iVet = [2, 4, 6, 8, 10], iK = 5
- Esperado: -1

Testes de Fronteira

5. Vetor com apenas um elemento (e o elemento é o procurado)

- Entrada: iVet = [7], iK = 7
- Esperado: 0

6. Vetor com apenas um elemento (e o elemento não é o procurado)

- Entrada: iVet = [7], iK = 3
- Esperado: -1

7. Vetor vazio

- Entrada: iVet = [], iK = 10
- Esperado: -1

Testes com valores negativos

8. Busca de número negativo

- Entrada: iVet = [-10, -5, 0, 5, 10], iK = -5
- Esperado: 1

Tabela do Roteiro de Testes - Busca Binária Iterativa

Descrição	Vetor de Entrada	Valor Buscado (iK)	Resultado Esperado
Elemento no meio do vetor	[1, 3, 5, 7, 9]	5	2 (índice do 5)
Elemento na primeira posição	[10, 20, 30, 40]	10	0
Elemento na última posição	[5, 15, 25, 35, 45]	45	4
Elemento que não existe no vetor	[2, 4, 6, 8, 10]	5	-1
Vetor com apenas um elemento (e o elemento é o procurado)	[7]	7	0
Vetor com apenas um elemento (e o elemento não é o procurado)	[7]	3	-1
Vetor vazio	[ ]	10	-1
Busca de número negativo	[-10, -5, 0, 5, 10]	-5	1
Elemento repetido no vetor	[2, 4, 4, 4, 6, 8]	4	1, 2 ou 3 (qualquer índice com valor 4)

Exercício 2

Pré-condição

Certifique-se de que o banco de dados do sistema possui logins e senhas cadastrados.

Caso de Teste 1: Login e senha incorretos

- Inserir no campo “Login” um login inválido.
- Inserir no campo “Senha” uma senha inválida.
- Pressionar o botão “Entrar”.
- Resultado esperado: Exibir a mensagem “Login e/ou Senha incorretos”.

Caso de Teste 2: Login correto e senha incorreta

- Inserir no campo “Login” um login válido.
- Inserir no campo “Senha” uma senha inválida.
- Pressionar o botão “Entrar”.
- Resultado esperado: Exibir a mensagem “Login e/ou Senha incorretos”.

Caso de Teste 3: Login e senha corretos

- Inserir no campo “Login” um login válido.
- Inserir no campo “Senha” uma senha válida.
- Pressionar o botão “Entrar”.
- Resultado esperado: Gerar o código dinâmico de validação em duas etapas e enviar por SMS ao celular cadastrado.

Caso de Teste 4: Código de validação incorreto

- Após receber o SMS, inserir um código de validação incorreto no campo apropriado.

- Pressionar o botão de validação.
- Resultado esperado: Exibir a mensagem “Login não autorizado!”.

Caso de Teste 5: Código de validação correto

- Após receber o SMS, inserir o código de validação correto no campo apropriado.
- Pressionar o botão de validação.
- Resultado esperado: Exibir a mensagem “Login realizado com sucesso” e liberar o acesso ao sistema.

Caso de Teste 6: Código não enviado

- Inserir login e senha corretos.
- Simular falha no envio do SMS (como falta de conexão com a rede).
- Resultado esperado: Exibir mensagem de erro relacionada ao envio do código dinâmico.

Caso de Teste 7: Código expirado

- Receber o código de validação, mas esperar o tempo limite para expiração do código.
- Inserir o código expirado no campo apropriado.
- Resultado esperado: Exibir mensagem “Código expirado, solicite um novo código”.

Caso de Teste 8: Campo vazio na validação do código

- Após receber o código, deixar o campo de validação em branco.
- Pressionar o botão de validação.
- Resultado esperado: Exibir mensagem “Preencha o campo do código”.

Caso de Teste 9: Interação simultânea

- Simultaneamente, insira login, senha e o código de validação correto.
- Pressionar os botões de validação.
- Resultado esperado: O sistema deve processar todas as etapas sem erros.