**PLANO DE TESTES**

# Marv Support

Software de Recebimento e Resolução de Chamados

## 1. Introdução

Bem-vindo ao nosso Software de Recebimento e Resolução de Chamados, uma ferramenta projetada para otimizar e agilizar o processo de atendimento ao cliente. Este software foi desenvolvido com o intuito de simplificar a gestão de solicitações, proporcionando uma solução eficiente para receber, rastrear e resolver chamados de maneira organizada e eficaz.

### 1.1. Objeto

O Software de Recebimento e Resolução de Chamados é uma plataforma avançada projetada para otimizar o processo de atendimento ao cliente, simplificando a gestão de solicitações e melhorando a eficiência operacional das equipes de suporte. Com recursos inovadores, essa ferramenta se destaca como uma solução abrangente para as demandas crescentes de suporte técnico e atendimento ao cliente.

### 1.2. Objetivo

O teste do Software tem como objetivo validar a eficácia, confiabilidade e usabilidade da plataforma antes do seu lançamento oficial.
Certificar-se de que o software atenda aos requisitos funcionais e de desempenho.
Identificar e corrigir quaisquer bugs ou problemas no sistema.
Verificar a usabilidade e a eficácia do software no gerenciamento de chamados e incidentes.

## 3. Abordagem

**Preencher .....**

***<Especifica a forma de realização dos testes. Abrange, entre outros aspectos, as técnicas, ferramentas e restrições, além disso, são definidos critérios para iniciação, aprovação e encerramento dos testes. também são definidas as condições para a suspensão e retomada dos testes.>***

## 4. Missão de Avaliação e Motivação dos Testes

A missão dos testes nesta iteração é garantir a entrega de um Software de Recebimento e Resolução de Chamados de alta qualidade, que atenda aos requisitos funcionais e não funcionais estabelecidos. Buscamos assegurar a confiabilidade, desempenho e segurança do sistema, proporcionando uma experiência de usuário excepcional e contribuindo para o sucesso do projeto.
A motivação dos testes é impulsionada pela busca incessante pela qualidade do software. É a convicção de que testar minuciosamente cada aspecto não apenas garante funcionalidade, mas também fortalece a resistência do produto. Nos testes, vemos a oportunidade de aprimorar, mitigar riscos e superar as expectativas dos usuários. É o compromisso de construir não apenas um produto, mas uma experiência confiável e valiosa.

## 4.2 Missão de Avaliação

A missão deste esforço de avaliação é localizar o maior número de erros possível, avaliar os riscos da qualidade perceptível e informar sobre os riscos do projeto. Buscamos, assim, assegurar a qualidade do produto, satisfazer as expectativas dos envolvidos e cumprir as determinações do processo de teste estabelecido.

## 4.3 Motivadores dos Testes

Riscos de Qualidade:

Identificar e mitigar riscos relacionados à qualidade do software, garantindo que o produto final atenda aos padrões de desempenho e confiabilidade esperados.

Riscos Técnicos:

Avaliar possíveis desafios técnicos que podem surgir durante a implementação do software, assegurando que a solução seja robusta e escalável.

Riscos do Projeto:

Minimizar riscos associados à entrega do projeto, garantindo que prazos sejam cumpridos e que o software atenda às expectativas dos stakeholders.

Casos de Uso:

Validar a funcionalidade do software em relação aos casos de uso identificados, garantindo que todas as interações previstas estejam implementadas corretamente.

Requisitos Funcionais:

Verificar se o software atende a todos os requisitos funcionais especificados, garantindo que as principais funcionalidades estejam implementadas de acordo com as expectativas.

Requisitos Não Funcionais:

Avaliar aspectos não funcionais, como desempenho, segurança e usabilidade, para garantir que o software não apenas funcione corretamente, mas também atenda aos requisitos de desempenho e experiência do usuário.

Falhas ou Erros Suspeitos:

Investigar e corrigir falhas conhecidas, bem como abordar erros suspeitos identificados durante o desenvolvimento, garantindo uma base sólida para o software.

Solicitações de Mudança:

Avaliar e testar qualquer nova funcionalidade ou alteração solicitada durante o ciclo de desenvolvimento, garantindo que as mudanças não impactem negativamente as funcionalidades existentes.

Mudanças na Arquitetura:

Testar e validar quaisquer mudanças na arquitetura do sistema para garantir que novas implementações estejam alinhadas com a visão arquitetônica e não introduzam problemas inesperados.

Experiência do Usuário:

Garantir que a experiência do usuário seja positiva, testando a usabilidade, a eficiência na execução de tarefas e a navegabilidade da interface.

## 5. Casos de Teste:

Desenvolver casos de teste que cubram todos os requisitos funcionais do software. Isso inclui:
Registro de chamados/incidentes.
Priorização de chamados.
Atribuição de chamados a técnicos.
Comunicação com os solicitantes.
Resolução de incidentes.
Geração de relatórios.
Auditoria de registros.

## 6. Testes Alvos

Equipe de Teste:
Identificar membros da equipe de teste, incluindo testadores, coordenador de teste e gerente de projeto.

Ambiente de Teste:
Especificar o ambiente de teste, incluindo hardware, software e configurações de rede.

Testes de Desempenho:
Realizar testes de desempenho para verificar a capacidade do software de lidar com cargas de trabalho típicas. Isso pode incluir testes de estresse para avaliar a escalabilidade.

Testes de Usabilidade:
Avaliar a usabilidade do software, verificando a facilidade de uso e navegabilidade da interface do usuário.

Testes de Integração:
Certificar-se de que o software se integra com outros sistemas, como bancos de dados ou serviços de autenticação, conforme necessário.

Testes de Segurança:
Avaliar a segurança do software, verificando a prevenção de acesso não autorizado e a proteção de

dados sensíveis.

Testes de Recuperação e Continuidade:
Testar a capacidade do software de se recuperar de falhas e garantir a continuidade do serviço.

Testes de Aceitação do Usuário (UAT):
- Envolva os usuários finais em testes de aceitação para garantir que o software atenda às suas necessidades e expectativas.

Relatórios de Defeitos:
- Documentar todos os defeitos encontrados durante os testes, incluindo detalhes sobre como reproduzi-los e a gravidade.

Correções e Reteste:
- Após a correção dos defeitos identificados, reteste o software para garantir que os problemas tenham sido resolvidos adequadamente.

Documentação de Teste:
- Prepare documentação detalhada de testes, incluindo casos de teste, resultados, relatórios de defeitos e documentação de UAT.

Aprovação e Liberação:
- Após a conclusão bem-sucedida dos testes, obtenha a aprovação da equipe de projeto e dos usuários finais antes de liberar o software para produção.

Monitoramento Pós-Implementação:
- Implemente um plano de monitoramento contínuo após a implementação para garantir que o software continue funcionando conforme o esperado.

## 7. Testes Executados

## 7.1 Lista de tickets

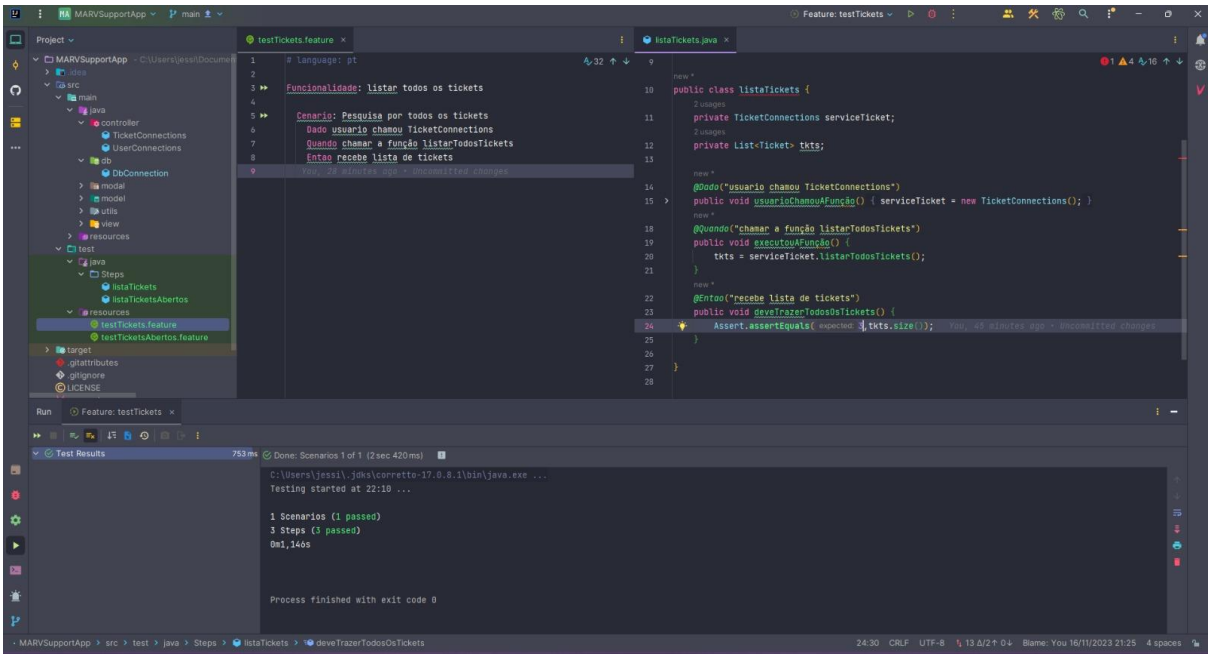
O teste de lista de tickets deve checar se a função “listarTodosTickets”, funciona corretamente dentro do código, executando uma lista com todos os tickets(chamados), independente do estado destes.

Gherkin:

```
Funcionalidade: listar todos os tickets

  Cenario: Pesquisa por todos os tickets
    Dado usuario chamou TicketConnections
    Quando chamar a função listarTodosTickets
    Entao recebe lista de tickets

Cenario: Pesquisa por todos os tickets e não encontrar nengum
    Dado usuario chamou TicketConnections
    Quando chamar a função listarTodosTickets
  Entao recebe lista vazia de tickets
```




## 7.2 Conexão com Banco de Dados

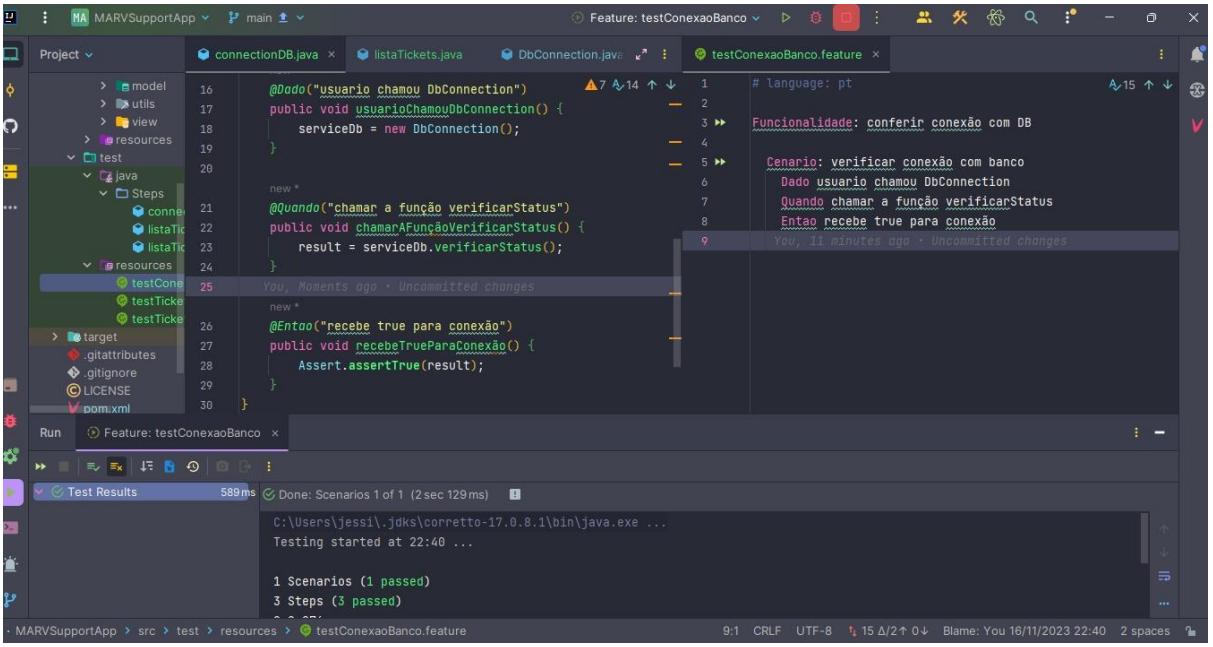
Testa se a conexão com o banco de dados foi bem estabelecida

Gherkin:

Funcionalidade: conferir conexão com DB

  Cenario: verificar conexão com banco com erro
    Dado usuario chamou DbConnection
    Quando chamar a função verificarStatus
    Entao recebe false para conexão

Cenario: verificar conexão com banco com sucesso
    Dado usuario chamou DbConnection
    Quando chamar a função verificarStatus
    Entao recebe true para conexão




## 7.3 Listar Tickets por categoria

Testa se a função “listarTicketsEspecificos” funciona corretamente e separa os tickets por categoria

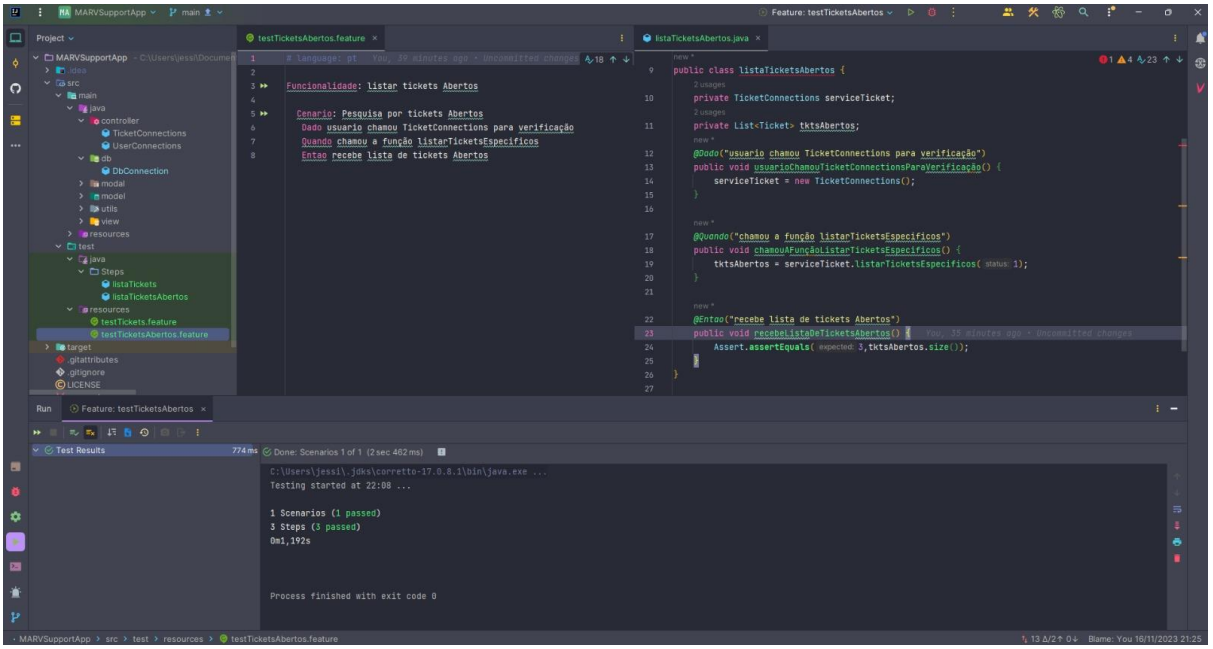
Gherkin:

Funcionalidade: listar tickets

Cenario: Pesquisa por tickets Pendente
Dado usuario chamou TicketConnections para verificação
Quando chamou a função listarTicketsEspecificos
Entao recebe lista de tickets Pendente

Cenario: Pesquisa por tickets Fechado
Dado usuario chamou TicketConnections para verificação
Quando chamou a função listarTicketsEspecificos
Entao recebe lista de tickets Fechado

Cenario: Pesquisa por tickets Abertos
Dado usuario chamou TicketConnections para verificação
Quando chamou a função listarTicketsEspecificos
Entao recebe lista de tickets Abertos


MARVSupportApp
main
Feature: testTicketsAbertos
Project
testTicketsAbertos.feature
listaTicketsAbertos.java
# language: pt
Funcionalidade: listar tickets Abertos
Cenario: Pesquisa por tickets Abertos
Dado usuario chamou TicketConnections para verificação
Quando chamou a função listarTicketsEspecificos
Entao recebe lista de tickets Abertos
public class listaTicketsAbertos {
private TicketConnections serviceTicket;
private List<Ticket> tktsAbertos;
@Dado("usuario chamou TicketConnections para verificação")
public void usuarioChamouTicketConnectionsParaVerificação() {
serviceTicket = new TicketConnections();
@Quando("chamou a função listarTicketsEspecificos")
public void chamouAFunçãoListarTicketsEspecificos() {
tktsAbertos = serviceTicket.listarTicketsEspecificos( status: 1);
@Entao("recebe lista de tickets Abertos")
public void recebeListaDeTicketsAbertos() {
Assert.assertEquals( expected: 3,tktsAbertos.size());
Run
Feature: testTicketsAbertos
Test Results
774 ms
Done: Scenarios 1 of 1 (2 sec 462 ms)
C:\Users\jessi\.jdks\corretto-17.0.8.1\bin\java.exe ...
Testing started at 22:08 ...
1 Scenarios (1 passed)
3 Steps (3 passed)
0m1,192s
Process finished with exit code 0
MARVSupportApp > src > test > resources > testTicketsAbertos.feature