**Plano de Teste**

**PsychoMeet:**

*versão 1.0*

*Membros da Equipe:*

*Bruno Gabriel Malaquias*

*Karen Pereira de Oliveira*

*Matheus Sousa Fonseca Faria*

Sumário

[1 - Introdução 4](#_Toc182751802)

[2 - Requisitos a Testar 5](#_Toc182751803)

[3 - Tipos de teste 7](#_Toc182751804)

[3.1 - Métodos da Classe 9](#_Toc182751805)

[3.2 - Integração dos Componentes 9](#_Toc182751806)

[4 - Recursos 10](#_Toc182751807)

[4.1 - Ambiente de teste - Software e Hardware 10](#_Toc182751808)

[4.2 - Ferramenta de teste 10](#_Toc182751809)

[5 – Casos de Teste 10](#_Toc182751810)

[6 - Cronograma 11](#_Toc182751811)

**Histórico das alterações:**

| **Data** | **Versão** | **Descrição** | **Autor(a)** |
| --- | --- | --- | --- |
| 17/11/2024 | 1.0 | Release inicial | Matheus |

### 1 - Introdução

Este documento apresenta um plano abrangente de testes para o sistema **PsychoMeet**, desenvolvido como parte de um projeto interdisciplinar de 2024. Ele visa detalhar os requisitos a serem testados, os tipos de testes definidos para cada iteração, os recursos de hardware e software necessários e o cronograma das atividades de teste ao longo do ciclo de desenvolvimento. A estrutura foi elaborada para fornecer clareza e suporte ao gerenciamento do projeto, permitindo o acompanhamento preciso da evolução dos testes e a identificação de pontos críticos de melhoria.

**Informações do Projeto e Componentes de Software**

O **PsychoMeet** é uma solução digital destinada a democratizar o acesso a serviços psicológicos, conectando pacientes a psicólogos de maneira eficiente e acessível. O sistema é composto por quatro módulos principais:

* **Gerenciamento de Conta**: Cadastro e edição de perfis de usuários (pacientes e psicólogos).
* **Busca de Profissionais**: Localização de psicólogos conforme critérios definidos.
* **Gerenciamento de Consultas**: Agendamento, cancelamento e pagamento de consultas.
* **Pós-consulta**: Feedbacks e histórico de consultas.

Os testes descritos neste documento focam no **Módulo de Pacientes** no backend e no **Módulo de Login** no frontend. Os componentes principais a serem testados incluem:

1. Métodos backend: create, delete e update do serviço de gerenciamento de pacientes.
2. Funcionalidades frontend: Fluxo de login e autenticação de usuários.

**Requisitos a Testar**

Os requisitos a serem testados são baseados nos requisitos funcionais e não funcionais definidos nos documentos do projeto:

* **Funcionais**:
  + RF 01: Realizar cadastro.
  + RF 02: Editar perfil.
  + RF 03: Fluxo de login.
* **Não Funcionais**:
  + RNF 01: Interface do usuário amigável.
  + RNF 02: Desempenho rápido e responsivo.
  + RNF 07: Conformidade legal com normas de privacidade.

**Estratégias de Teste**

Os testes estão divididos em duas categorias principais:

1. **Testes Unitários (Backend)**:
   * **Objetivo**: Garantir o funcionamento isolado dos métodos create, delete e update.
   * **Estratégias**:
     + Validação de entradas obrigatórias.
     + Testes de persistência no banco de dados.
     + Simulação de cenários de erro, como exceções no DAO e entradas inválidas.
2. **Testes de Integração e E2E (Frontend)**:
   * **Objetivo**: Validar o fluxo completo de login, desde a interface até o backend.
   * **Estratégias**:
     + Cenários de sucesso e falha no login.
     + Testes de usabilidade e responsividade.
     + Validação de segurança no armazenamento e transmissão de dados sensíveis.

**Recursos Necessários e Estimativa de Esforço**

1. **Hardware e Software**:
   * **Backend**: Servidor Java para testes unitários, com banco de dados de teste configurado.
   * **Frontend**: Ambiente de testes E2E configurado em navegadores modernos (Chrome, Firefox).
   * **Ferramentas**:
     + JUnit e Mockito para testes unitários no backend.
     + Cypress ou Playwright para testes E2E no frontend.
2. **Equipe e Tempo**:
   * Desenvolvedor backend: 5 dias para implementação e ajuste dos testes unitários.
   * Desenvolvedor frontend: 7 dias para configuração e execução dos testes E2E.

**Elementos Resultantes do Projeto de Testes**

Os resultados esperados incluem:

* Relatórios detalhados de cobertura de testes unitários no backend.
* Resultados de testes de integração e E2E, avaliando funcionalidade e usabilidade.
* Identificação de falhas e pontos de melhoria no código e na interface do sistema.

**Programa a Ser Testado**

O programa sob teste é o sistema **PsychoMeet**, com foco inicial no módulo de gerenciamento de pacientes (backend) e no fluxo de login (frontend). Este documento serve como base para o planejamento e execução de testes, assegurando a entrega de um sistema robusto e confiável.

### 2 - Requisitos a Testar

Esta seção apresenta os casos de uso e os requisitos não funcionais identificados como objetos de teste ao longo do desenvolvimento do sistema PsychoMeet. Os requisitos foram extraídos diretamente da especificação funcional e não funcional do projeto, subdivididos para melhor organização e clareza. A seguir, são listados os requisitos a serem testados.

**Casos de uso:**

| **Identificador do caso de uso** | **Nome do caso de uso** |
| --- | --- |
| id UC1 | Realizar cadastro |
| id UC2 | |  | | --- | | Editar perfil |  |  | | --- | |  | |
| id UC3 | Realizar login |

Descrição dos Casos de Uso:

1. UC1 - Realizar Cadastro: O paciente ou psicólogo deve ser capaz de criar uma conta fornecendo dados obrigatórios, como nome, CPF, e-mail e senha.
2. UC2 - Editar Perfil: O usuário deve poder atualizar informações pessoais previamente cadastradas, desde que respeitem as regras de edição.
3. UC3 - Realizar Login: Os usuários devem acessar o sistema fornecendo credenciais válidas.

**Requisitos não-funcionais:**

| **Identificador do requisito** | **Nome do requisito** |
| --- | --- |
| RNF01 | |  | | --- | | Interface do usuário amigável |  |  | | --- | |  | |
| RNF02 | |  | | --- | | Desempenho rápido |  |  | | --- | |  | |
| RNF03 | Conformidade com privacidade |

Descrição dos Requisitos Não Funcionais:

1. RNF01 - Interface do Usuário Amigável: A interface deve ser intuitiva, garantindo facilidade de uso para pacientes e psicólogos ao realizarem operações no sistema.
2. RNF02 - Desempenho Rápido: O sistema deve apresentar respostas rápidas às ações dos usuários, incluindo operações de login, cadastro e edição.
3. RNF03 - Conformidade com Privacidade: O sistema deve atender às regulamentações de proteção de dados, garantindo o sigilo de informações sensíveis, como CPF e histórico de consultas.

### 3 - Tipos de teste

Esta seção apresenta os tipos de testes escolhidos para a próxima iteração do projeto PsychoMeet e os testes planejados para iterações futuras. Os testes foram definidos considerando os requisitos funcionais e não funcionais identificados, o tipo da aplicação e os recursos disponíveis. A seguir, detalham-se os testes que serão empregados para garantir a qualidade do sistema.

**Testes Definidos para a Próxima Iteração**

1. **Teste de Unidade**:
   * **Descrição**: Testes de unidade serão realizados para validar o correto funcionamento de métodos isolados no backend, como os métodos create, delete e update do serviço de gerenciamento de pacientes.
   * **Objetivo**: Identificar e corrigir defeitos em componentes específicos, garantindo que cada método funcione de forma independente.
   * **Ferramentas**: JUnit e Mockito para simular dependências e verificar interações.
2. **Teste de Interface de Usuário (UI)**:
   * **Descrição**: Validação do fluxo de login no frontend, verificando a usabilidade e a apresentação de mensagens adequadas para erros e sucessos.
   * **Objetivo**: Garantir que o sistema seja intuitivo, acessível e responsivo, conforme o requisito não funcional RNF01.
   * **Ferramentas**: Cypress ou Playwright para automação de testes E2E.
3. **Teste de Segurança e Controle de Acesso**:
   * **Descrição**: Testes para verificar a autenticação e autorização de usuários, incluindo a proteção de informações sensíveis.
   * **Objetivo**: Garantir conformidade com o requisito RNF03, verificando o armazenamento seguro de senhas e o correto uso de tokens de autenticação.
   * **Ferramentas**: OWASP ZAP para identificar vulnerabilidades e Cypress para validação de fluxos.

**Testes Planejados para Iterações Futuras**

1. **Teste de Performance**:
   * **Descrição**: Verificar o tempo de resposta do sistema em operações críticas, como login, cadastro e edição de perfil.
   * **Objetivo**: Garantir que o sistema atenda ao requisito RNF02 de desempenho rápido.
   * **Ferramentas**: Apache JMeter ou K6 para simular cargas e medir tempos de resposta.
2. **Teste de Carga**:
   * **Descrição**: Simular múltiplos usuários acessando o sistema simultaneamente para validar sua escalabilidade.
   * **Objetivo**: Assegurar que o sistema suporte o número esperado de usuários sem degradação significativa de desempenho.
   * **Ferramentas**: Apache JMeter para geração de carga.
3. **Teste de Stress**:
   * **Descrição**: Testar o sistema além de seus limites normais de operação para identificar pontos de falha.
   * **Objetivo**: Garantir a estabilidade do sistema sob condições extremas e identificar gargalos de desempenho.
   * **Ferramentas**: K6 ou ferramentas de simulação personalizadas.
4. **Teste de Instalação**:
   * **Descrição**: Validar o processo de implantação do sistema em diferentes ambientes.
   * **Objetivo**: Garantir que o sistema seja instalado corretamente e funcione em todos os ambientes suportados.
   * **Ferramentas**: Scripts automatizados para implantação.

**Critérios de Completude**

Os testes serão considerados completos quando:

* Todos os métodos críticos do backend forem validados com 100% de cobertura em testes unitários.
* O fluxo de login do frontend estiver funcionando corretamente em todos os navegadores suportados.
* Os requisitos não funcionais de desempenho e segurança forem atendidos com base em métricas definidas previamente.

### 3.1 - Métodos da Classe

Para teste de funcionalidade. Aqui deve-se verificar se cada classe retorna o esperado. Se possível usar teste automatizado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo** | **Teste do módulo de login do frontend** | | | |
| **Técnica:** | **( ) manual** | | **(x) automática** | |
| **Estágio do teste** | **Integração ( )** | **Sistema ( )** | **Unidade (x)** | **Aceitação ( )** |
| **Abordagem do teste** | **Caixa branca (x)** | | **Caixa preta ( )** | |
| **Responsável(is)** | **Matheus** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo** | **Teste do módulo de paciente do backend** | | | |
| **Técnica:** | **( ) manual** | | **(x) automática** | |
| **Estágio do teste** | **Integração ( )** | **Sistema ( )** | **Unidade (x)** | **Aceitação ( )** |
| **Abordagem do teste** | **Caixa branca (x)** | | **Caixa preta ( )** | |
| **Responsável(is)** | **Bruno e Karen** | | | |

### 3.2 - Integração dos Componentes

Para teste de funcionalidade. Aqui deve-se verificar se as classes e métodos conseguem fazer a integração entre elas para uma sequência de ações do programa. Se possível usar teste automatizado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo** | **Descreva aqui o objetivo** | | | |
| **Técnica:** | **(x) manual** | | **(x) automática** | |
| **Estágio do teste** | **Integração (x)** | **Sistema ( )** | **Unidade ( )** | **Aceitação ( )** |
| **Abordagem do teste** | **Caixa branca (x)** | | **Caixa preta (x)** | |
| **Responsável(is)** | **Programador(es) ou equipe de testes** | | | |

### 4 - Recursos

### Esta seção apresenta os recursos necessários para a execução dos testes do sistema PsychoMeet, detalhando o ambiente de teste, incluindo hardware e software, além das ferramentas de automatização que serão utilizadas ao longo do processo.

### 4.1 - Ambiente de teste - Software e Hardware

O ambiente de teste será configurado para suportar tanto o backend quanto o frontend, com o objetivo de simular condições reais de uso do sistema e assegurar a qualidade de todas as funcionalidades testadas.

* **Hardware**:
  + Servidores com processadores de, no mínimo, 4 núcleos e 16 GB de RAM, para suportar testes de carga e stress.
  + Máquinas locais de desenvolvimento e teste com 8 GB de RAM e processador dual-core para validação unitária e integração.
  + Conexão estável à internet para execução de testes integrados e acesso a APIs externas.
* **Software**:
  + **Sistema Operacional**:
    - Ambiente backend: Windows 11.
    - Máquinas locais: Windows 11.
  + **Servidores**:
    - Servidor Java para backend (Spring Boot 2.x).
    - Banco de dados MySQL em ambiente isolado para testes.
  + **Browsers**:
    - Google Chrome (versão mais recente).
  + **Frameworks e Bibliotecas**:
    - Spring Boot para desenvolvimento do backend.
    - Angular para frontend.

### 4.2 - Ferramenta de teste

Diversas ferramentas serão utilizadas para garantir a eficiência e a abrangência dos testes, abrangendo desde validação unitária até testes de integração e automação.

* **Ferramentas de Teste Unitário**:
  + **JUnit**: Ferramenta principal para execução de testes unitários no backend.
  + **Mockito**: Para mockar dependências e simular comportamentos de métodos.
* **Ferramentas de Automação para Testes E2E**:
  + **Cypress**: Utilizado para automação de testes de interface do usuário, garantindo funcionalidade e responsividade.
  + **Playwright**: Alternativa para validar comportamentos em múltiplos navegadores.
* **Ferramentas de Teste de Desempenho e Carga**:
  + **Apache JMeter**: Para realizar testes de carga e stress, simulando múltiplos usuários simultaneamente.
  + **K6**: Focado em medir a escalabilidade e o tempo de resposta do sistema sob condições reais.
* **Ferramentas de Teste de Segurança**:
  + **OWASP ZAP**: Para identificar vulnerabilidades no sistema e garantir a proteção de dados sensíveis.
* **Controle de Versão e Integração Contínua**:
  + **GitHub Actions**: Configurado para executar automaticamente os testes após cada commit.
  + **Jenkins**: Alternativa para integração contínua, gerenciando a execução de testes automatizados.

### 5 – Casos de Teste

Descreva aqui os casos de testes da aplicação. Seguir o template disponibilizado no FAIOline.

|  |  |
| --- | --- |
| **Número do Caso de Teste** | CT - 001 |
| **Nome** | Fazer Login com um usuario válido |
| **Atores** | Admin |
| **Requisitos Associados** | RF 01, 02 |
| **Pré-condições** | Usuário cadastrado |
| **Entradas** | Nome do usuario: josé Senha do usuario: 1234 Selecionar a opção logar |
| **Saidas** | Usuário logado com sucesso |
| **Ambientes** | Acesso a aplicação usando o browser Chrome |
| **Procedimentos** | Login - CT - 001 |
| **Dependências** | Browser Chrome instalado, usuario cadastrado |

|  |  |
| --- | --- |
| **Número do Caso de Teste** | CT - 002 |
| **Nome** | Recuperar |
| **Atores** | Admin |
| **Requisitos Associados** | RF 03, 04 |
| **Pré-condições** | Usuário cadastrado |
| **Entradas** | Nome do usuario: josé Senha do usuario:  Seleccionar a opção recuperar senha |
| **Saidas** | Usuário deve ser direcionado para a recuperação de senha |
| **Ambientes** | Acesso a internet Browser Chrome |
| **Procedimentos** | Login – Recuperar senha CT - 002 |
| **Dependências** | Browser Chrome instalado  Usuario cadastrado |

### 6 - Cronograma

| **Tipo de teste** | **Duração** | **data de início** | **data de término** |
| --- | --- | --- | --- |
| planejar teste |  | dd/mm/aaaa | dd/mm/aaaa |
| projetar teste |  | dd/mm/aaaa | dd/mm/aaaa |
| implementar teste |  | dd/mm/aaaa | dd/mm/aaaa |
| executar teste |  | dd/mm/aaaa | dd/mm/aaaa |
| avaliar teste |  | dd/mm/aaaa | dd/mm/aaaa |