****

**Plano de Teste VemRankSer**

Rawad Aboueltaif

Alison Silva

Felipe Noguez

Jonas

Luiza Valentini

Mailton Nascimento

Thaís Araujo

**ÍNDICE**

[**1 Introdução**](#_heading=h.afs3jbtptm32) **3**

[**2 Arquitetura**](#_heading=h.w08abia8gplp) **3**

[2.1 Referências/Documentações do Projeto](#_heading=h.c9pa7srsi1n7) 3

[**3 Requisitos**](#_heading=h.6rjr5jvdfmcl) **4**

[**4 Funcionalidades**](#_heading=h.61jrjwn9rwf8) **4**

[**5 Estratégia de Teste**](#_heading=h.li5yfbyo66x8) **6**

[**6 Ambientes e Ferramentas**](#_heading=h.robhb8upcvpi) **7**

[6.1 Definições do ambiente de testes](#_heading=h.gjxk4isi8fcr) 7

[6.2 Ferramentas de testes](#_heading=h.fvz46yijyx1s) 7

[**7 Classificação de Bugs**](#_heading=h.ttjk7epgdkh5) **8**

[**8 Cenário de Testes**](#_heading=h.i3pw9ybzby4o) **9**

[**9 Definição de Pronto**](#_heading=h.j90robq4garp) **9**

# 1 Introdução

VemRankSer é um sistema desenvolvido para ranquear os participantes do programa VemSer, auxiliar na criação de novos módulos, trilhas e atividades,

O sistema também irá facilitar o trabalho dos instrutores com o gerenciamento de criação de novas atividades e correções das mesmas, como também providenciará um visão mais ampla do andamento do VemSer para os coordenadores.

# 2 Arquitetura

O framework utilizado para a implementação do sistema VemRankSer é o React, que possui uma arquitetura orientada a componentes.

Para o armazenamento, consulta e alteração de dados da aplicação, será usada uma API que disponibiliza um conjunto de serviços REST. Em que o cliente (aplicação) envia uma requisição à uma URL (endpoint) para acessar os serviços da API, e tem como retorno um objeto na forma de um JSON

### 2.1 Referências/Documentações do Projeto

A tabela abaixo identifica a documentação utilizada para a elaboração deste Plano de teste.

| **Documento** | **Irá fazer parte do projeto ?** | **Observações** |
| --- | --- | --- |
| Especificação de Requisitos | Sim | Irá definir as funcionalidades e o ambiente que o Sistema deve possuir |
| Especificação de Casos de Uso | Sim | Descreve de forma detalhada as funcionalidades do Sistema |
| Especificação das Regras de negócio | Sim | Definirá as regras e os comportamentos do Sistema |

# 3 Requisitos

| **Requisitos Funcionais** | **Requisitos Não Funcionais** |
| --- | --- |
| **RF001** - O sistema deve permitir o cadastro de novos usuários | **NF001** - O sistema deverá ser desenvolvido utilizando o framework React |
| **RF002** - O sistema deve permitir o login dos usuários cadastrados | **NF002** - O banco de dados utilizado deverá ser o SQL |
| **RF003** - O sistema deve permitir a alteração de informações do usuário logado | **NF003** - O sistema deve ser navegável apenas com o usuário devidamente logado no sistema |
| **RF004** - O sistema deve informar o status atual dos participantes do programa VemSer |  |

# 4 Funcionalidades

| **Funcionalidades** | **Comportamento Esperado** | **Verificações** |
| --- | --- | --- |
| Login /  Tela Inicial | * Ao clicar no botão de ***Login*** deve redirecionar para tela de login * Ao tentar logar e falhar 3 vezes consecutivas o usuário ficará bloqueado | * Botão de login redireciona para a tela Dashboard * Todos os campos são obrigatórios * Exibir mensagem de erro no caso de campo vazio e dados inválidos * Bloquear caso a senha for incorreta 3 vezes seguidas * Caso login com sucesso deve redirecionar ao saldo * Botão ***Esqueceu sua senha*** deve redirecionar para página de recuperação de conta |
| Cadastro de novos usuários | * Ao digitar e-mail, nome completo, senha, confirmar a senha e selecionar a atuação, status e trilha do usuário e clicar em cadastrar, irá efetuar um cadastro na plataforma e o usuário deverá ser redirecionado para a tela de Dashboard * Deve indicar o campo obrigatório a ser corrigido pelo usuário e ter opção para mostrar a senha | * Todos os campos são obrigatórios * Campo nome deve conter apenas letras e espaço com limite de 50 caracteres * Campo e-mail deve ter um e-mail válido * Senha deve ter no mínimo 8 caracteres e no máximo 16 * Senha deve ter ao menos um número, uma letra maiúscula, uma letra minúscula e um símbolo * Exibir mensagem de alerta enquanto confirmação de senha não for igual a senha * Botão Mostrar deve alternar entre mostrar/ocultar a senha quando clicado * Exibir mensagem de confirmação em caso de cadastro com sucesso e ser redirecionado para a tela de Dashboard * Exibir mensagem de erro no caso de campo vazio, dados inválidos e e-mail já cadastrado no sistema |

# 5 Estratégia de Teste

O plano de testes abrange todas as funcionalidades descritas na tabela acima. Serão executados testes em todos os níveis conforme a descrição abaixo.

**Testes Unitários/Back-end**: O código terá uma cobertura de 100% de testes unitários, que são de responsabilidade dos desenvolvedores.

**Testes Unitários/Front-end:** O código terá uma cobertura de 70% de testes unitários, que são de responsabilidade dos desenvolvedores.

**Testes de Integração**: Serão executados testes de integração em todos os endpoints, e esses testes serão de responsabilidade do time de qualidade.

**Testes Automatizados**: Serão realizados testes end-to-end em pelo menos 90% de toda a aplicação.

**Testes Manuais**: Todas as funcionalidades serão testadas manualmente pelo time de qualidade seguindo o Cenário de Testes deste plano de testes.

# 6 Ambientes e Ferramentas

### 6.1 Definições do ambiente de testes

* Teremos testes unitários desenvolvidos individualmente pelos desenvolvedores com um volume pequeno de dados.
* Teremos testes em toda a aplicação com testadores e com um volume grande de dados, são utilizados dados reais ou criados na hora, também teremos testes de aceitação em toda a aplicação.
* Todos os testadores deverão ter desktops similares aos da empresa que será implantado o sistema, além de ter todos os programas instalados que os usuários terão disponíveis em suas máquinas.

Os testes serão feitos do ambiente de homologação, e contém as mesmas configurações do ambiente de produção com uma massa de dados gerada previamente pelo time de qualidade.

### 6.2 Ferramentas de testes

| **Ferramenta** | **Versão** | **Time** | **Descrição** |
| --- | --- | --- | --- |
| Cypress | 9.54 | Qualidade | Ferramenta para realização de testes de end-to-end |
| Rest Assured | 4.0.0 | Qualidade | Ferramenta para realização de testes de API |
| Jest | 27.5.1 | Desenvolvimento | Framework utilizada para testes unitários |
| Junit | 5.8.2 | Desenvolvimento/  Back-end | Framework utilizada para testes unitários |

# 7 Classificação de Bugs

Os Bugs serão classificados com as seguintes severidades:

| **ID** | **Nível de Severidade** | **Descrição** |
| --- | --- | --- |
| 1 | Blocker | * Bug que bloqueia o teste de uma função ou feature causa crash na aplicação. * O Botão não funciona impedindo o uso completo da funcionalidade. * Bloqueia a entrega. |
| 2 | Grave | * Funcionalidade não funciona como o esperado * Input incomum causa efeitos irreversíveis |
| 3 | Moderada | * Funcionalidade não atinge certos critérios de aceitação, mas sua funcionalidade em geral não é afetada * Mensagem de erro ou sucesso não é exibida |
| 4 | Pequena | * Quase nenhum impacto na funcionalidade, porém atrapalha a experiência * Erro ortográfico * Pequenos erros de UI |

# 8 Cenário de Testes

### 8.1 Cenários de Testes API

| **Cenário de Testes (CT)** |
| --- |
| **Usuario** |
| 1. testeDeveAtualizarFotoUsuarioComSucesso 2. testeDeveAtualizarEmailDoUsuarioComSucesso 3. testeDeveAtualizarSenhaDoUsuarioComSucesso 4. testeDeveMensagemDeErrorAoAtualizarEmailPraOutroExistente 5. testeDeveCadastrarAlunoComSucesso 6. testeDeveCadastrarInstrutorComSucesso 7. testeDeveCadastrarCooerdenadorComSucesso 8. testeDeveCadastrarGestaoComSucesso 9. testeDeveRetornarErroAoCadastrarNovoUsuarioComEmailExistenteNoBancoDeDados 10. testeDeveRealizarLoginComSucessotesteDeveRetornarUsuarioLogado 11. testeDeveRetornarErroPadraoAoTentarLogarComSenhaInvalida 12. testeDeveRetornarErroPadraoAoTentarLogarComEmailInvalido 13. testeDeveRetornarErroPadraoAoTentarLogarComEmailSenhaVazios 14. testeDeveListarTodosUsuarios 15. testeDeveRetornarFotoDoUsuario 16. testeDeveRetornarErroAoListarUsuariosSemEstarLogado 17. testeDeveRetornarErroAoListarUsuariosComTamanhoListaZerado 18. testeDeveRetornarUsuarioBuscadoPorNome 19. testeDeveRetornarErroAoListarUsuariosInexistentePorNome 20. testeDeveRetonarAlunosPorLogin 21. testeDeveRetonarErroPadraoAoBuscarAlunoPorLoginInexistente 22. testeDeveRetornarListaDeAlunos 23. testeDeveRetornarErroAoListarAlunosPorPaginaNegativa 24. testeDeveRetornarListaDeTrilhaDoAluno 25. testeDeveRetornarUsuarioBuscadoPorId 26. testeDeveRetornarErroAoBuscarUsuarioPorIdNegativo |
| **Trilha** |
| 1. testeDeveCadastrarTrilhaComSucesso 2. testeDeveVincularTrilhaAoAlunoComSucesso 3. testeDeveRetornarErroAoVincularTrilhaAAlunoInexistente 4. testeDeveRetornarErroAoVincularTrilhaIdNegativoAoAluno 5. testeDeveRetornarErroAoCadastrarTrilhaSemAnoEdicao 6. testeDeveVincularTrilhaAoInstrutorComSucesso 7. testeDeveRetornarErroAoVincularTrilhaIdNegativoAoInstrutor 8. testeDeveListarPorTrilhaComAlunos 9. testeDeveRetornarListaVaziaAoBuscarPorTrilhaInexistenteComAlunos 10. testeDeveRetornarListaDeTrilhaPorNome 11. testeDeveRetornarErroAoBuscarListaDeTrilhaPorNomeInexistente 12. testeDeveRetornarListaPorRanking 13. testeDeveRetornarErroAoBuscarListaDeTrilhaPorIdNegativo 14. testeDeveRetornarVazioAoBuscarListaDeTrilhaPorEdicaoNegativo 15. testeDeveRetornarTrilhaBuscadaPorId 16. testeDeveRetornarErroAoBuscarTrilhaPorIdNegativo |
| **Modulo** |
| 1. testeDeveCadastrarModuloComSucesso 2. testeDeveRetornarErroAoCadastrarModuloComDataNoFormatoErrado 3. testeDeveRetornarErroAoCadastrarModuloSemNome 4. testeDeveVincularModuloATrilha 5. testeDeveRetornarMensagemErroAoTentarVincularModuloComIdNegativoAUmaTrilhaComIdNegativo 6. testeDeveListarTodosOsModulos 7. testeDeveListarModuloPorId 8. testeDeveRetornarMensagemDeErroAoBuscarIdModuloComNumeroFloat |

### 8.2 Cenários de Testes Front-End

| **Cenário de Testes (CT)** |
| --- |
| **Login** |
| 1. testeDeveLogarComSucesso 2. testeDeveRetonarMensagemErroPadraoAoPreencherCampoEmailNaoCadastrado 3. testeDeveRetonarMensagemErroPadraoAoPreencherCampoSenhaIncorreta 4. testeDeveRetonarMensagemErroPadraoAoPreencherCampoSenhaVazio 5. testeDeveRetonarMensagemErroPadraoAoPreencherCampoEmailVazio 6. testeDeveRetonarMensagemErroPadraoAoPreencherCampoEmailInvalido |
| **Admin** |
| 1. testeDeveCadastrarUsuarioCompletoTipoGestaoComSucesso 2. testeDeveCadastrarUsuarioCompletoTipoAlunoComSucesso 3. testeDeveCadastrarUsuarioCompletoTipoInstrutorComSucesso 4. testeDeveCadastrarUsuarioCompletoTipoCoordenadorComSucesso 5. testeDeveCadastrarNovoUsuarioApenasPreenchendoOsCamposObrigatorios 6. testeDeveRetornarMensagemErroAoCadastrarNovoUsuarioComLoginExistenteNoBancoDeDados 7. testeDeveRetornarMensagemErroAoCadastrarNovoUsuarioComEmailExistenteNoBancoDeDados 8. testeDeveRetornarMensagemDeErroAoCadastrarUsuarioComCamposObrigatoriosVazios 9. testeDeveRetornarMensagemDeErroAoCadastrarUsuarioComTodosCamposVazios 10. testeDeveRetornarMensagemDeErroAoEditarUsuarioPreenchandoTodosOsCamposVazios 11. testeDeveRetornarAPaginaDeUsuariosAoClicarNoBotaoCancelar 12. testeDeveRetornarAPaginaDeLoginAoClicarNoBotaoSairDaConta 13. testeDeveVerificarAcaoDasSetasDireitaEsquerdaListaDeUsuarios 14. testeDeveVerificarAcaoDosSelectsAtivoEInativo |
| **Modulo** |
|  |

# 9 Definição de Pronto

Será considerada pronta as funcionalidades que passarem pelas verificações e testes descritos neste plano de teste, não apresentarem bugs com a severidade acima de moderada, e passarem por uma validação de negócio de responsabilidade do time de produto.