**Test Plan MuscleUuup PUC**

# Introdução

# Este plano de testes foi desenvolvido para garantir a qualidade do MuscleUuup, uma plataforma de exercícios físicos personalizada. O objetivo é assegurar que o MuscleUuup atenda aos padrões de desempenho esperados. Este documento define as estratégias e procedimentos para testar todas as funcionalidades, identificando possíveis falhas para uma experiência do usuário satisfatória.

# Arquitetura

Para o desenvolvimento do MuscleUuup, serão adotadas as seguintes diretrizes arquiteturais:

1. **Front-end:**

O framework React será utilizado para a implementação do front-end do MuscleUuup. O React é reconhecido por sua arquitetura orientada a componentes, o que permite uma estrutura modular e reutilizável para a construção da interface do usuário.

1. **Back-end:**

O back-end do MuscleUuup será desenvolvido em Node.js, utilizando TypeScript para garantir a segurança e tipagem estática durante o desenvolvimento. Será adotada a arquitetura MVC (Model-View-Controller), separando a lógica de negócios (Model), a apresentação (View) e o controle das requisições (Controller), facilitando a manutenção e escalabilidade do sistema.

1. **Persistência:**

O banco de dados relacional PostgreSQL será utilizado para armazenar e gerenciar os dados do MuscleUuup. O PostgreSQL é uma escolha robusta e confiável para aplicativos que exigem um alto nível de integridade e consistência dos dados.

1. **Integração:**

A comunicação entre o front-end, back-end e aplicativo móvel será realizada através de serviços RESTful. O front-end e o aplicativo móvel enviarão requisições HTTP para endpoints específicos no back-end, que responderá com os dados necessários em formato JSON.

1. **Teste de Software:**

Para garantir a qualidade do software, serão implementados testes unitários e de integração. O framework Jest será utilizado para escrever e executar esses testes, assegurando o correto funcionamento de cada componente e integração do sistema.

Essa arquitetura foi escolhida visando a modularidade, escalabilidade e eficiência do MuscleUuup, garantindo uma experiência de usuário fluida e uma base sólida para futuras expansões e atualizações do sistema.

# Funcionalidades

|  |  |
| --- | --- |
| **Funcionalidade** | Cadastro |
| **Comportamento Esperado** | Ao digitar email, nome completo, usuário e senha irá efetuar um cadastro na plataforma e o usuário deverá ser redirecionado para a tela de login.  Deve indicar o campo obrigatório a ser corrigido pelo usuário. |
| **Verificações** | * Senha min 8 caracteres e no máximo 18 * Todos os campos devem ser obrigatórios. * Exibir uma mensagem de confirmação em caso positivo. * Redirecionar o usuário para tela de login. * Exibir a mensagem de falha em caso de usuário existente. * Exibir mensagem de falha em caso de confirmação de senha não ser igual * Exibir mensagem de falha no caso de campo obrigatório incompleto. |
| **Critérios de Aceite** | * Senha de 8 caracteres, no mínimo. * Todos os campos devem ser obrigatórios. * Exibir uma mensagem de confirmação em caso positivo. * Redirecionar o usuário para tela de login. * Exibir a mensagem de falha em caso usuário já existente. * Exibir mensagem de falha caso a confirmação de senha não seja igual a senha digitada anteriormente. * Exibir mensagem de falha caso o campo obrigatório esteja incompleto. |

|  |  |
| --- | --- |
| **Funcionalidade** | Login |
| **Comportamento Esperado** | Ao digitar seu usuário e senha corretamente, o usuário irá logar na plataforma.  Ao tentar se logar e falhar 3 vezes consecutivas, o usuário terá que esperar 15 minutos para tentar logar novamente.  O sistema deve aceitar usuários que já está logado em outro *device* se logar novamente |
| **Verificações** | * Login no Sistema com sucesso * Usuário Inválido * Usuário não preencher campo obrigatório * Senha Incorreta * Senha Incorreta 3 vezes |
| **Critérios de Aceite** | * Ter acessibilidade no sistema * Redimensionar a Tela |

|  |  |
| --- | --- |
| **Funcionalidade** | Excluir cadastro |
| **Comportamento Esperado** | Ao clicar no item em excluir perfil, o mesmo deverá ser excluído.  Deve aparecer uma caixa de confirmação perguntando se usuário deseja mesmo excluir o perfil.  O usuário deve ter a opção de cancelar exclusão.  O usuário deve confirmar a exclusão. |
| **Verificações** | * Exclusão do perfil * Mensagem de confirmação em caso positivo. * Mensagem de cancelamento. |
| **Critérios de Aceite** |  |

|  |  |
| --- | --- |
| **Funcionalidade** | Gerenciar Treino |
| **Comportamento Esperado** | O usuário deve ser capaz de criar, visualizar, modificar e excluir treinos personalizados. Ao acessar a funcionalidade de gerenciamento de treino, o usuário poderá realizar as seguintes operações: criar um novo treino, visualizar os treinos existentes, editar um treino existente e excluir um treino. |
| **Verificações** | * O usuário deve ter acesso à funcionalidade de gerenciamento de cadastro. * Todas as informações do perfil devem ser editáveis. * Após realizar as edições, o usuário deve poder salvar as alterações. * O sistema deve validar as informações inseridas para garantir a integridade dos dados. |
| **Critérios de Aceite** | * O usuário deve ser capaz de editar todas as informações do seu perfil com sucesso. * Todas as alterações devem ser salvas corretamente no sistema. * O sistema deve fornecer feedback claro ao usuário sobre o sucesso ou falha das operações de edição de perfil. |

|  |  |
| --- | --- |
| **Funcionalidade** | Gerenciar Cadastro |
| **Comportamento Esperado** | O usuário deve poder editar suas informações cadastrais, como nome, idade, e-mail, etc. Ao acessar a funcionalidade de gerenciamento de cadastro, o usuário poderá editar as informações do seu perfil de forma simples e intuitiva. |
| **Verificações** | * O usuário deve ter as opções de criar, visualizar, editar e excluir treinos. * Ao criar um novo treino, todos os campos obrigatórios devem ser preenchidos. * Ao visualizar os treinos, todas as informações relevantes devem ser exibidas de forma clara e organizada. * Ao editar um treino, o usuário deve poder modificar as informações do treino de forma intuitiva. * Ao excluir um treino, o usuário deve confirmar a ação para evitar exclusões acidentais. |
| **Critérios de Aceite** | * O usuário deve ser capaz de realizar todas as operações de gerenciamento de treino de forma eficiente. * Todas as operações devem ser acompanhadas por mensagens claras de sucesso ou falha, conforme apropriado. * O sistema deve garantir a integridade dos dados, evitando inconsistências ou perda de informações. |

|  |  |
| --- | --- |
| **Funcionalidade** | Listar Alunos |
| **Comportamento Esperado** | O personal trainer deve ser capaz de visualizar uma lista de alunos que estão sob sua supervisão. Ao acessar a funcionalidade de listagem de alunos, o personal trainer poderá ver os nomes dos alunos e outras informações relevantes para cada um. |
| **Verificações** | * O personal trainer deve ter acesso à funcionalidade de listagem de alunos. * A lista de alunos deve ser apresentada de forma clara e organizada. * Todas as informações importantes sobre cada aluno devem ser exibidas na lista. |
| **Critérios de Aceite** | * O personal trainer deve ser capaz de visualizar a lista de alunos sem problemas. * Todas as informações na lista de alunos devem estar corretas e atualizadas. * A lista de alunos deve ser fácil de navegar e entender. |

|  |  |
| --- | --- |
| **Funcionalidade** | Gerenciar Alunos |
| **Comportamento Esperado** | O usuário deve ser capaz de adicionar, remover e modificar informações de treino dos alunos sob sua supervisão. Ao acessar a funcionalidade de gerenciamento de treinos compartilhados, o usuário poderá realizar as seguintes operações: adicionar um novo usuário ao treino compartilhado, editar informações de um usuário do treino compartilhado existente e excluir um usuário do treino compartilhado. |
| **Verificações** | * O usuário deve ter as opções de adicionar e excluir usuários do treino compartilhado. * Ao editar as informações de um usuário do treino compartilhado, o usuário que compartilhou o treino deve poder modificar os detalhes do aluno conforme necessário. * Ao excluir um aluno, o personal trainer deve confirmar a ação para evitar exclusões acidentais. |
| **Critérios de Aceite** | * O personal trainer deve ser capaz de realizar todas as operações de gerenciamento de alunos de forma eficiente. * Todas as operações devem ser acompanhadas por mensagens claras de sucesso ou falha, conforme apropriado. * O sistema deve garantir a integridade dos dados, evitando inconsistências ou perda de informações. |

|  |  |
| --- | --- |
| **Funcionalidade** | Atribuir Treino a outro usuário |
| **Comportamento Esperado** | O usuário deve ser capaz de atribuir treinos a outros usuários por meio de links compartilháveis. |
| **Verificações** | * O usuário deve ter acesso à funcionalidade de atribuição de treino por meio de link para outro usuário. * O usuário deve poder escolher entre atribuir um treino existente ou criar um novo treino para o aluno. |
| **Critérios de Aceite** | * O usuário deve ser capaz de atribuir treinos aos alunos com até 4 cliques. * Todas as informações sobre a atribuição de treino devem ser registradas corretamente no sistema. * O sistema deve fornecer feedback claro ao usuário sobre o sucesso ou falha da atribuição de treino. |

# Estratégia de Teste

* **Escopo de Testes**

O escopo dos testes abrangerá todas as funcionalidades listadas abaixo, excluindo a funcionalidade de gerenciar cadastro:

* Cadastro
* Login
* Excluir cadastro
* Gerenciar Treino
* Gerenciar Cadastro
* Listar Alunos
* Gerenciar Alunos
* Atribuir Treino a um Aluno

**Níveis de Testes e Responsabilidades**

1. **Testes Unitários:**

O código será coberto por testes unitários, visando atingir uma cobertura mínima de 60%. Esses testes serão de responsabilidade dos desenvolvedores, utilizando o framework Jest.

1. **Testes de Integração:**

Testes de integração serão realizados em todos os endpoints da aplicação. Esses testes serão de responsabilidade do time de qualidade.

1. **Testes Automatizados:**

Serão realizados testes end-to-end na funcionalidade de Login, utilizando a ferramenta Cypress. Esses testes serão automatizados para garantir a estabilidade e funcionamento adequado do processo de login.

1. **Testes Manuais:**

Todas as funcionalidades serão testadas manualmente pelo time de qualidade, seguindo a documentação de cenários de teste e o presente Test Plan. O objetivo é validar a usabilidade, integridade e correto funcionamento das funcionalidades.

1. **Versão Beta:**

Antes do lançamento oficial, será disponibilizada uma versão beta para 3 usuários pré-cadastrados. Essa versão será submetida a testes de aceitação pelos usuários finais para identificar possíveis problemas e garantir a qualidade antes do release.

* **Ambiente e Ferramentas**

Os testes serão realizados no ambiente de homologação, que replica as configurações do ambiente de produção e contém uma massa de dados gerada previamente pelo time de qualidade.

Ferramentas Utilizadas:

|  |  |  |
| --- | --- | --- |
| **Ferramenta** | **Time** | **Descrição** |
| [Insomnia](https://insomnia.rest/download) | Qualidade | Ferramenta para realização de testes de API |
| [Jest](https://jestjs.io/pt-BR/) | Desenvolvimento | Framework utilizada para testes unitários |
| [Cypress](https://www.cypress.io/) | Qualidade | Ferramenta para testes end-to-end |
| [Lighthouse](https://developer.chrome.com/docs/lighthouse/overview/) | Desenvolvimento | Utilizado para avaliação de performance e acessibilidade da aplicação |
| Gravador de Passos | Desenvolvimento | Proverá evidências dos testes realizados para posterior análise e documentação |

# Classificação de Bugs

Os bugs encontrados durante os testes serão classificados de acordo com a seguinte tabela de severidades:

|  |  |  |
| --- | --- | --- |
| **ID** | **Nivel de Severidade** | **Descrição** |
| 1 | Blocker | * Bug que bloqueia o teste de uma função ou feature causa crash na aplicação. * 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 |

# Definição de Pronto

Será considerada pronta as funcionalidades que passarem pelas verificações e testes descritas nestes TestPlan, não apresentarem bugs com a severidade acima de Minor, e passarem por uma validação de negócio de responsabilidade do time de produto.