DOCUMENTAÇÃO DE SOFTWARE ATIVEMENTS

(Modelo de documentação de software)

**Histórico de Revisões**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versão | Descrição | Autor |
| 01/01/000  0 | 1.0 | Elaboração para análise da primeira versão do documento. | Nome do Autor |
|  |  |  |  |

**Sumário**

[1. Visão Geral 5](#_Toc175905122)

[Objetivo 5](#_Toc175905123)

[Escopo 5](#_Toc175905124)

[2. Introdução 5](#_Toc175905125)

[Contexto 5](#_Toc175905126)

[3. Requisitos 6](#_Toc175905127)

[Requisitos Funcionais 6](#_Toc175905128)

[Requisitos Não Funcionais 6](#_Toc175905129)

[4. Regras de Negócio 7](#_Toc175905130)

[5. Restrições 7](#_Toc175905131)

[6. Arquitetura 8](#_Toc175905132)

[Visão Geral da Arquitetura 8](#_Toc175905133)

[Diagramas de Arquitetura 8](#_Toc175905134)

[Modelagem do banco 9](#_Toc175905135)

[7. Design do Sistema 9](#_Toc175905136)

[Padrões de Design 9](#_Toc175905137)

[Estrutura do Código 10](#_Toc175905138)

[8. Detalhamento dos Módulos 10](#_Toc175905139)

[Módulo x 10](#_Toc175905140)

[9. Configuração e Instalação 10](#_Toc175905141)

[Pré-requisitos 10](#_Toc175905142)

[Passos de Instalação 10](#_Toc175905143)

[10. Uso 11](#_Toc175905144)

[Guia do Usuário 11](#_Toc175905145)

[Exemplos de Uso 11](#_Toc175905146)

[11. Testes 11](#_Toc175905147)

[Estratégias de Testes 11](#_Toc175905148)

[Casos de Teste 11](#_Toc175905149)

[Resultados dos Testes 11](#_Toc175905150)

[12. Manutenção 11](#_Toc175905151)

[Plano de Manutenção 11](#_Toc175905152)

[Registro de Alterações 12](#_Toc175905153)

[13. Protótipos 12](#_Toc175905154)

[14. Referências 13](#_Toc175905157)

# Visão Geral

## Objetivo

O **Ativements** é um sistema que tem como objetivo gerenciar ativos da escola **“S Informática”** proporcionando um controle eficaz dos equipamentos e otimizando os recursos da escola.

## Escopo

O sistema possibilitará:

- Que os usuários se cadastrem no sistema

- Que os usuários se autentiquem no sistema

- Que os usuários acessem a lista de todos os ativos

- Que os usuários cadastrem um ativo

- Que os usuários alterem dados de um ativo

- Que os usuários excluam um ativo

# Introdução

## Contexto

A **S Informática** possui um grande número de equipamentos de TI, incluindo computadores, notebooks, impressoras, projetos e outros dispositivos, utilizados em seus cursos. O controle desses ativos se torna um desafio, com dificuldades no:

**Rastreamento:** Dificuldade em localizar e identificar cada ativo, gerando perdas e desperdícios.

Após a implementação do sistema **Ativements**, será possível:

**- Organizar e catalogar:** Registrar todos os ativos com informações detalhadas (modelo, serial, data de aquisição, etc.).

**- Controlar localização:** Utilizar etiquetas de identificação e softwares de rastreamento para localização precisa.

# Requisitos

## Requisitos Funcionais

|  |  |
| --- | --- |
| **RF1- Login** | * Autenticação dos usuários no sistema |
| **RF2- Listagem de ativos** | * Exibir lista de ativos cadastrados |
| **RF3- Cadastro de usuário** | * Cadastrar usuários |
| **RF4- Cadastro de ativos** | * Cadastrar ativos |
| **RF5- Adição de ativos** | * Atualizar dados de um ativo |
| **RF6- Logout** | * Remover a autenticação do usuário logado |

## Requisitos Não Funcionais

|  |  |
| --- | --- |
| **RNF1- Performance** | * O tempo de resposta para cada operação do sistema deve ser   inferior a 3 segundos. |
| **RNF2- Disponibilidade** | * O sistema deve estar disponível 24 horas por dia, 7 dias por semana, com o mínimo de interrupções possíveis. |
| **RNF3- Segurança** | * O acesso ao sistema deve ser restrito a usuários indentificados por autenticação. |
| **RNF4- Usabilidade** | * O sistema deve ser intuitivo e ter uma interface amigável e simples. |

# Regras de Negócio

|  |  |
| --- | --- |
| **RN1** | * O login deverá ser realizado através de username do github. |
| **RN2** | * O cadastro no sistema deverá ser realizado com dados username do github válido. |
| **RN3** | * Se já existir um usuário cadastrado, o sistema deverá informar que o usuário já existe. |
| **RN4** | * Os ativos deverão ser exibidos com nº do ativo, nome do ativo e data e hora do registro, local e status de ativo ou inativo. |
| **RN5** | * O usuário não poderá cadastrar um ativo sem que todos os campos estejam preenchidos. |
| **RN6** | * Na atualização de um ativo, deverá ser registrado o usuário, data e hora da última modificação. |
| **RN7** | * Os usuários poderão filtrar os ativos pelo local. |
| **RN8** | * Ao se deslogar do sistema, deverá ser registrado a data e hora. |
| **RN9** | * Ao informar o local do ativo no momento do cadastro, se já houver o mesmo local já cadastrado, a ação deverá ser impedida. |

# Restrições

|  |  |
| --- | --- |
| **RS1** | * Sistema operacional no mínimo Windows 10. |
| **RS2** | * O sistema deve ter capacidade de armazenamento flexível. |

# Arquitetura

## Visão Geral da Arquitetura

* **Frontend:** Desenvolvido em ReactJS, responsável pela interface do usuário.
* **Backend:** Json server para simular toda a persistência de dados.

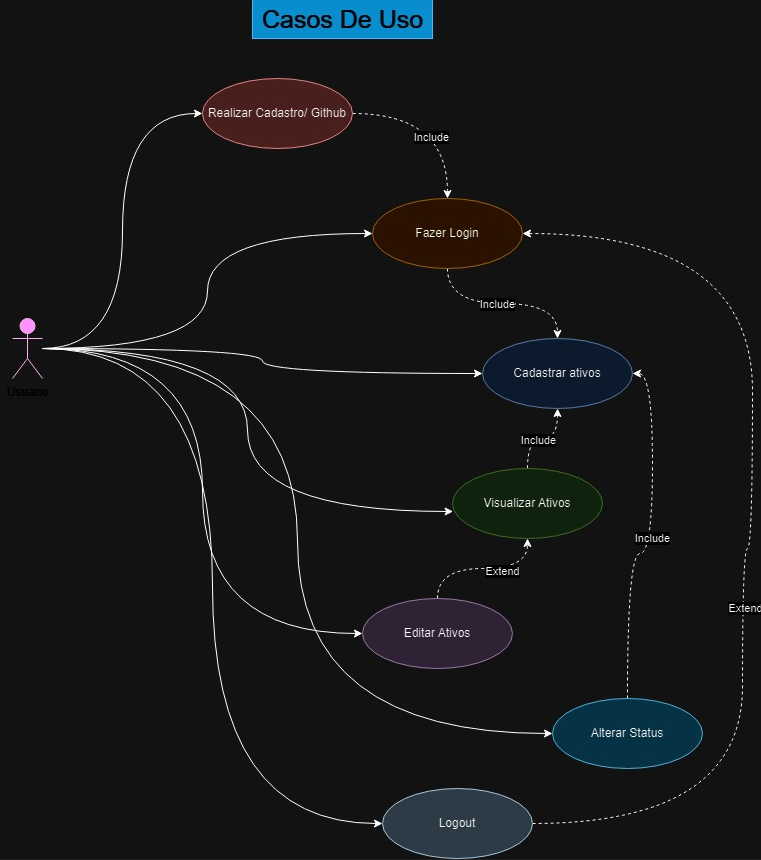
**Resumo:**

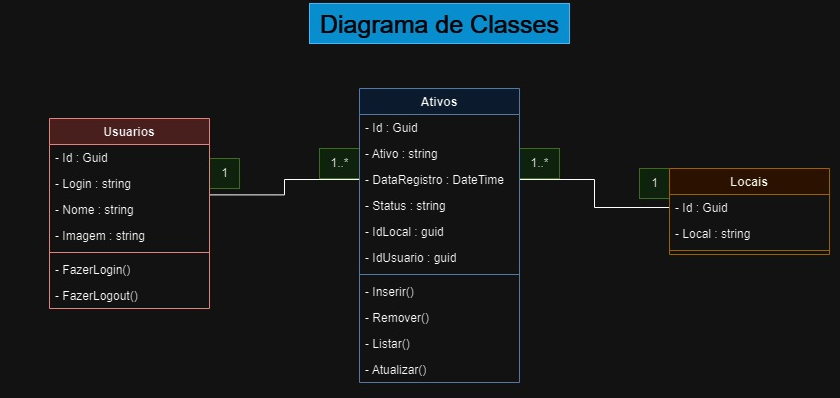
* O usuário interage com o frontend (React JS)

O frontend faz requisições para o json server e através de uma rota disponibiliza os recursos.

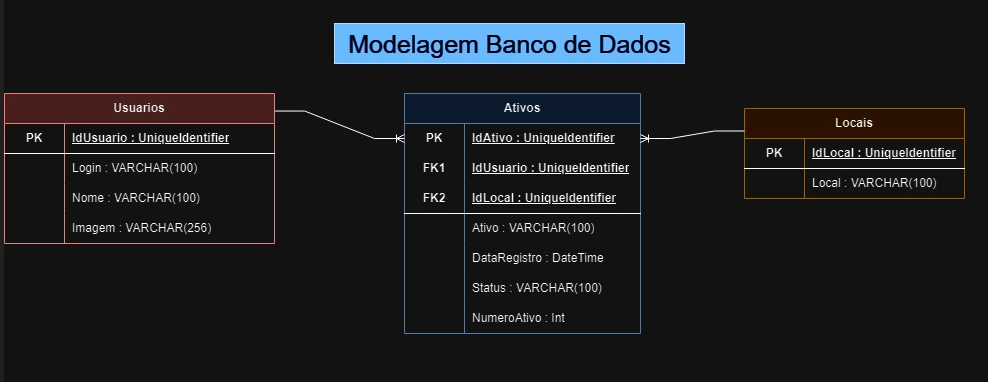
## Diagramas de Arquitetura

**Diagrama de caso de uso**

****

**Diagrama de classe**

## Modelagem do banco

****

# Design do Sistema

## Padrões de Design

O sistema usa como base para o frontend o Component-based Architecture (Arquitetura baseada em Componentes) que é um estilo de design de software em que a aplicação é dividida em componentes reutilizáveis e independentes. Cada componente encapsula uma e independentes. Cada componente encapsula uma parte especifica da funcionalidade da aplicação, incluindo a interface de usuário, lógica de negócios, e dados associados.

No contexto de ReactJS, cada componente é geralmente uma função ou classe que retorna um fragmento da interface do usuário (UI). Esses componentes podem ser combinados para formar páginas e funcionalidades completas da aplicação. Isso promove a modularidade, facilitando a manutenção, testes e reuso de código.

Para o backend é usado o JSON-Server que é uma biblioteca Node.js que simula uma API RESTful a partir de um arquivo JSON.

## Estrutura do Código

Detalhar a estrutura do código, incluindo pacotes, módulos, etc.

# Detalhamento dos Módulos

## Módulo x

Descrever em detalhes o Módulo x, suas funcionalidades, interfaces, etc.

# Configuração e Instalação

## Pré-requisitos

- Node JS

## Passos de Instalação

- Clone o repositório do projeto.

- No diretório do frontend, execute **npm install** para instalar as dependências

- Execute **npm start** no diretório do frontend para iniciar a aplicação React.

- Acesse <http://localhost:3000> no navegador para utilizar o sistema.

# Uso

## Guia do Usuário

Fornecer um guia de uso para os usuários finais.

## Exemplos de Uso

Incluir exemplos de uso do software.

# Testes

## Estratégias de Testes

Descrever a estratégia de testes adotada.

Por enquanto não foi elaborado estratégia de testes.

## Casos de Teste

Por enquanto não foi elaborado casos de testes.

## Resultados dos Testes

Incluir os resultados dos testes.

No momento ainda não existem alterações

# Manutenção

## Plano de Manutenção

Descrever o plano de manutenção do software.

O plano de manutenção ainda não foi desenvolvido.

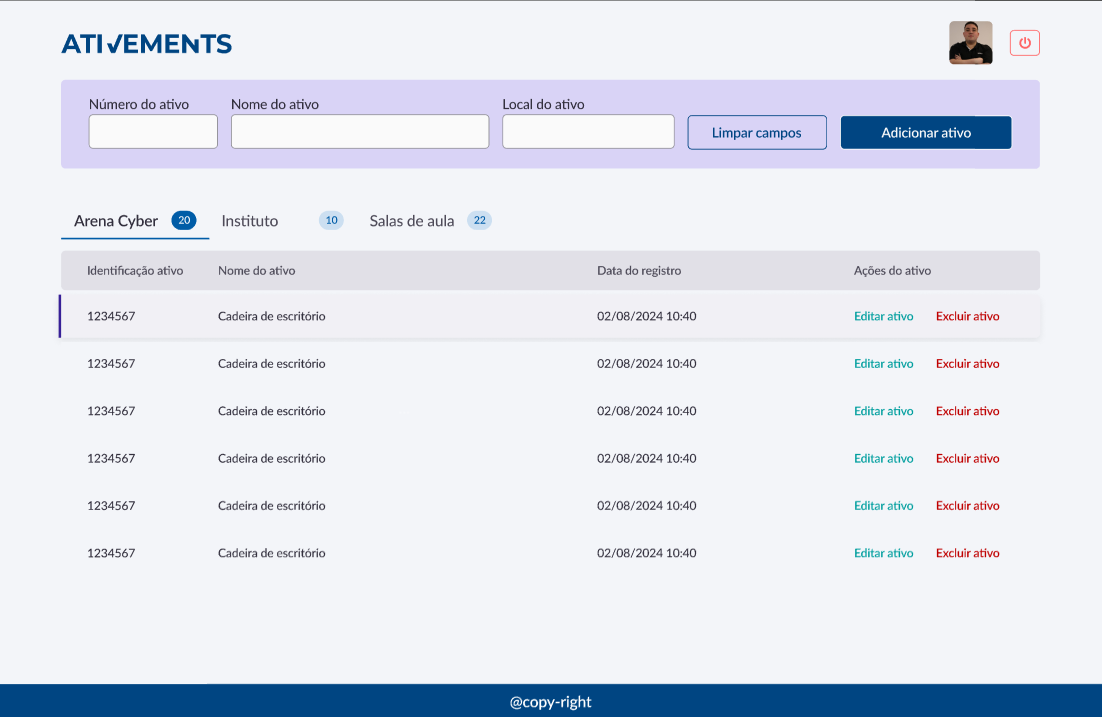
## Registro de Alterações

Incluir um registro de todas as alterações feitas no software.

# Protótipos

# 

# 

****

# Referências

Listar todas as referências utilizadas para o desenvolvimento do software, como livros, artigos, sites, etc.