# PRD - Sistema de Gestão de Patrimônio

## 1. Visão Geral

O Sistema de Gestão de Patrimônio é uma aplicação web desenvolvida para gerenciar o inventário de bens patrimoniais de uma organização. Ele permite o cadastro, consulta, atualização e controle de itens, seus locais físicos e usuários responsáveis pelo acesso ao sistema.

### Objetivo

* Gerenciar o cadastro de bens patrimoniais, incluindo informações como código, descrição, data de aquisição, fornecedor, local físico e situação.
* Controlar locais físicos onde os bens estão alocados.
* Gerenciar usuários com autenticação segura para acesso ao sistema.
* Oferecer uma interface amigável e fluida para engajar usuários em vídeos demonstrativos.

### Público-Alvo

* Gestores de patrimônio, administradores e funcionários responsáveis pelo controle de bens.

## 2. Tecnologias Utilizadas

* **Backend**: PHP 8.2 com framework Laravel 12.
* **Banco de Dados**: MariaDB 11.5.2 (compatível com MySQL).
* **Frontend**: Reactjs e Vite
* **Autenticação**: Laravel Authentication com login e senha.

## 3. Requisitos Funcionais

### 3.1. Cadastro de Bens Patrimoniais

* **Funcionalidade**: Registrar, editar, consultar e excluir bens patrimoniais.
* **Campos**:
  + Código (ID único, auto-incremento).
  + Código do Bem (varchar, até 10 caracteres).
  + Descrição (varchar, até 250 caracteres).
  + Data de Aquisição (date).
  + Forma de Aquisição (varchar, até 30 caracteres, ex.: compra, doação).
  + Fornecedor (varchar, até 250 caracteres).
  + Local Físico (lookup vinculado à tabela locais\_fisicos, campo descrição).
  + Situação (varchar, até 10 caracteres, ex.: ativo, inativo, manutenção).
  + Inventariado (boolean, default 0).
  + Referência (varchar, até 10 caracteres).
  + Observação (text, opcional).
* **Regras**:
  + O campo local\_fisico será preenchido via dropdown com base na tabela locais\_fisicos.
  + Validação de campos obrigatórios no backend (Laravel Validation).

### 3.2. Cadastro de Locais Físicos

* **Funcionalidade**: Gerenciar locais onde os bens estão alocados.
* **Campos**:
  + Código Local (ID único, auto-incremento).
  + Descrição (varchar, até 150 caracteres, obrigatório).
  + Ativo (boolean, default 1).
* **Regras**:
  + Apenas locais com ativo = 1 serão exibidos no dropdown de cadastro.local\_fisico.

### 3.3. Gestão de Usuários

* **Funcionalidade**: Registrar, editar, consultar e excluir usuários para acesso ao sistema.
* **Campos**:
  + ID (auto-incremento).
  + Matrícula (varchar, até 10 caracteres, opcional).
  + Nome (varchar, até 80 caracteres, obrigatório).
  + Login (varchar, até 50 caracteres, único, obrigatório).
  + Email (varchar, até 80 caracteres, opcional).
  + Senha (varchar, até 32 caracteres, criptografada com bcrypt, obrigatório).
  + Situação (boolean, default 0 - inativo).
  + Data de Cadastro (timestamp, default current\_timestamp).
* **Regras**:
  + Autenticação via login e senha.
  + Senha será armazenada com criptografia (bcrypt do Laravel).
  + Apenas usuários com situacao = 1 podem fazer login.

### 3.4. Autenticação

* **Funcionalidade**: Login seguro com validação de credenciais.
* **Regras**:
  + Redirecionamento para dashboard após login bem-sucedido.
  + Bloqueio de acesso para usuários com situacao = 0.
  + Interface de login com design atrativo (botões grandes, cores vibrantes, ideal para vídeos).

### 3.5. Relatórios

* **Funcionalidade**: Gerar relatórios de bens por local físico, situação ou data de aquisição.
* **Formato**: Exibição em tabela e exportação para CSV/PDF.
* **Regras**:
  + Filtros dinâmicos para facilitar a navegação.
  + Interface visual para destacar relatórios em vídeos (ex.: gráficos com Chart.js).

## 4. Requisitos Não Funcionais

* **Desempenho**: Resposta do sistema em menos de 2 segundos para consultas com até 10.000 registros.
* **Segurança**:
  + Criptografia de senhas com bcrypt.
  + Proteção contra SQL Injection (Eloquent ORM).
  + CSRF protection nativa do Laravel.
* **Usabilidade**:
  + Interface responsiva para desktop e mobile.
  + Design moderno com Tailwind CSS, cores vibrantes e animações sutis para engajar o público em vídeos.
* **Escalabilidade**: Suporte para até 50.000 bens patrimoniais e 1.000 usuários.
* **Compatibilidade**: Suporte aos navegadores Chrome, Firefox e Edge (versões recentes).

## 5. Banco de Dados

### 5.1. Especificações

* **SGBD**: MariaDB 11.5.2.
* **Charset**: utf8mb4\_unicode\_ci (suporte a caracteres especiais, ideal para português).
* **Mecanismo de Armazenamento**:
  + Tabela cadastro: InnoDB (transações e integridade referencial).
  + Tabela locais\_fisicos: MyISAM (consultas rápidas, sem necessidade de transações complexas).
  + Tabela usuarios: InnoDB (integridade para autenticação).
* **Índices**:
  + locais\_fisicos: Chave primária em cod\_local.
  + cadastro: Chave primária em cod, índice em local\_fisico para lookups.
  + usuarios: Chave primária em id, índice único em login.

### 5.2. Diagrama ER

O diagrama ER descreve as entidades e relacionamentos do sistema:

[Entidade: cadastro]

+--------------+------------------+

| Atributo | Tipo |

+--------------+------------------+

| cod | INT (PK, AI) |

| codigo\_bem | VARCHAR(10) |

| descricao | VARCHAR(250) |

| data\_aquisicao | DATE |

| forma\_aquisicao | VARCHAR(30) |

| fornecedor | VARCHAR(250) |

| local\_fisico | VARCHAR(6) (FK) |

| situacao | VARCHAR(10) |

| inventariado | TINYINT(1) |

| referencia | VARCHAR(10) |

| observacao | TEXT |

+--------------+------------------+

|

| 1:N

|

[Entidade: locais\_fisicos]

+--------------+------------------+

| Atributo | Tipo |

+--------------+------------------+

| cod\_local | INT (PK, AI) |

| descricao | VARCHAR(150) |

| ativo | TINYINT(1) |

+--------------+------------------+

[Entidade: usuarios]

+--------------+------------------+

| Atributo | Tipo |

+--------------+------------------+

| id | INT (PK, AI) |

| matricula | VARCHAR(10) |

| nome | VARCHAR(80) |

| login | VARCHAR(50) |

| email | VARCHAR(80) |

| senha | VARCHAR(32) |

| situacao | TINYINT(1) |

| data\_cadastro | TIMESTAMP |

+--------------+------------------+

**Relacionamentos**:

* cadastro.local\_fisico referencia locais\_fisicos.descricao (relacionamento 1:N).
* Não há relacionamento direto entre usuarios e outras tabelas, pois a autenticação é gerenciada pelo Laravel.

### 5.3. Script SQL

O script SQL fornecido será utilizado como base para o banco de dados. Abaixo está a versão corrigida e otimizada, garantindo compatibilidade com o sistema:

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";  
START TRANSACTION;  
SET time\_zone = "+00:00";

CREATE DATABASE IF NOT EXISTS Patrimonio\_L12 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;  
USE Patrimonio\_L12;

CREATE TABLE locais\_fisicos (  
cod\_local INT NOT NULL AUTO\_INCREMENT,  
descricao VARCHAR(150) NOT NULL,  
ativo TINYINT(1) NOT NULL DEFAULT 1,  
PRIMARY KEY (cod\_local)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE cadastro (  
cod INT NOT NULL AUTO\_INCREMENT,  
codigo\_bem VARCHAR(10) DEFAULT NULL,  
descricao VARCHAR(250) DEFAULT NULL,  
data\_aquisicao DATE DEFAULT NULL,  
forma\_aquisicao VARCHAR(30) DEFAULT NULL,  
fornecedor VARCHAR(250) DEFAULT NULL,  
local\_fisico VARCHAR(150) DEFAULT NULL,  
situacao VARCHAR(10) DEFAULT NULL,  
inventariado TINYINT(1) NOT NULL DEFAULT 0,  
referencia VARCHAR(10) DEFAULT NULL,  
observacao TEXT DEFAULT NULL,  
PRIMARY KEY (cod),  
FOREIGN KEY (local\_fisico) REFERENCES locais\_fisicos(descricao) ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE usuarios (  
id INT NOT NULL AUTO\_INCREMENT,  
matricula VARCHAR(10) DEFAULT NULL,  
nome VARCHAR(80) NOT NULL,  
login VARCHAR(50) NOT NULL,  
email VARCHAR(80) DEFAULT NULL,  
senha VARCHAR(255) NOT NULL,  
situacao TINYINT(1) NOT NULL DEFAULT 0,  
data\_cadastro TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
PRIMARY KEY (id),  
UNIQUE KEY (login)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

COMMIT;