# SGII – Sistema de Gestão Integrada Inteligente

## Tema:

Desenvolvimento de uma plataforma web para gerenciamento inteligente de espaços físicos, controle de chaves, reservas e inventário de recursos compartilhados em ambientes institucionais diversos, como empresas, escolas e laboratórios.

## Resumo do Escopo:

O SGII tem como objetivo permitir que qualquer instituição — pública ou privada — organize de forma centralizada a reserva de salas e equipamentos, o empréstimo e rastreamento de chaves, e o controle de inventário de recursos físicos. A plataforma pode ser aplicada com pouca ou nenhuma customização em diferentes tipos de organizações, trazendo economia de tempo, redução de erros humanos e maior transparência na gestão de infraestrutura.

## Funcionalidades Principais:

* Reserva de ambientes: salas, laboratórios, auditórios, veículos, etc.
* Controle de chaves e acessos físicos: com histórico e relatórios.
* Gestão de inventário: equipamentos, móveis, materiais permanentes.
* Painel administrativo: com indicadores de uso, notificações e alertas.
* Sistema multiusuário com permissões: adequado para empresas, escolas, universidades, clínicas ou qualquer organização com estrutura física.

## Tecnologias Sugeridas:

Frontend: React

Backend: Node.js com Express

Banco de dados: PostgreSQL ou MongoDB

Extras:

* Autenticação com JWT
* Upload de arquivos (comprovantes, laudos, etc.)
* Relatórios PDF e gráficos interativos

## Estrutura do Banco de Dados:

A seguir, a descrição das principais tabelas do sistema:

### instituicoes

* id (PK)
* nome
* tipo (empresa, escola, laboratório, etc.)
* cnpj\_ou\_codigo
* endereco
* telefone
* email

### usuarios

* id (PK)
* nome
* email
* senha\_hash
* cargo (admin, técnico, usuário, etc.)
* instituicao\_id (FK)
* nivel\_acesso

### espacos

* id (PK)
* nome
* tipo (sala, laboratório, auditório, etc.)
* capacidade
* recursos (texto ou JSON: projetor, AC, etc.)
* instituicao\_id (FK)

### reservas

* id (PK)
* espaco\_id (FK)
* usuario\_id (FK)
* data\_inicio
* data\_fim
* finalidade
* status (pendente, confirmada, cancelada)

### chaves

* id (PK)
* codigo\_identificador
* espaco\_id (FK)
* disponivel (boolean)
* observacoes

### movimentacoes\_chaves

* id (PK)
* chave\_id (FK)
* usuario\_id (FK)
* data\_retirada
* data\_devolucao
* responsavel\_entrega
* responsavel\_recebimento
* observacoes

### tens\_inventario

* id (PK)
* nome\_item
* descricao
* numero\_patrimonio (único)
* localizacao\_atual
* quantidade
* estado\_conservacao (novo, bom, danificado, etc.)
* data\_aquisicao
* instituicao\_id (FK)

### logs\_uso

* id (PK)
* usuario\_id (FK)
* acao (reserva criada, chave retirada, item adicionado, etc.)
* entidade (reservas, chaves, inventario, etc.)
* entidade\_id
* timestamp

## Extras:

* Tabela notificacoes (para alertas de vencimento, reservas próximas)
* Tabela fotos\_documentos (para anexar arquivos ou fotos aos registros)
* Tabela permissoes\_customizadas (para controle granular de acessos)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabela A | Tabela B | Cardinalidade A → B | Cardinalidade B → A | Tipo de Relação |
| instituicoes | usuarios | (1, N) | (1, 1) | Cada usuário pertence a uma instituição. |
| instituicoes | espacos | (1, N) | (1, 1) | Cada espaço pertence a uma instituição. |
| instituicoes | itens\_inventario | (1, N) | (1, 1) | Cada item pertence a uma instituição. |
| usuarios | reservas | (0, N) | (1, 1) | Usuários fazem váresreservas. |
| espacos | reservas | (0, N) | (1, 1) | Espaços podem ser reservados. |
| espacos | chaves | (0, 1) | (0, 1) | Cada chave está ligada a um espaço. |
| chaves | movimentacoes\_chaves | (0, N) | (1, 1) | Uma chave pode ter várias movimentações. |
| usuarios | movimentacoes\_chaves | (0, N) | (1, 1) | Um usuário pode movimentar várias chaves. |
| usuarios | logs\_uso | (0, N) | (1, 1) | Cada log é gerado por um usuário. |

|  |  |
| --- | --- |
| Entidade | Atributos |
| instituicoes | id (PK), nome, tipo, cnpj\_ou\_codigo, endereco, telefone, email |
| usuarios | id (PK), nome, email, senha\_hash, cargo, nivel\_acesso, instituicao\_id (FK) |
| espacos | id (PK), nome, tipo, capacidade, recursos, instituicao\_id (FK) |
| reservas | id (PK), espaco\_id (FK), usuario\_id (FK), data\_inicio, data\_fim, finalidade, status |
| chaves | id (PK), codigo\_identificador, espaco\_id (FK - opcional), disponivel, observacoes |
| movimentacoes\_chaves | id (PK), chave\_id (FK), usuario\_id (FK - opcional), data\_retirada, data\_devolucao, responsavel\_entrega (FK), responsavel\_recebimento (FK), observacoes |
| itens\_inventario | id (PK), nome\_item, descricao, numero\_patrimonio, localizacao\_atual, quantidade, estado\_conservacao, data\_aquisicao, instituicao\_id (FK) |
| logs\_uso | id (PK), usuario\_id (FK - opcional), acao, entidade, entidade\_id, timestamp |

-- 🏛️ INSTITUIÇÕES

CREATE TABLE instituicoes (

id SERIAL PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

tipo VARCHAR(50), -- empresa, escola, laboratório, etc.

cnpj\_ou\_codigo VARCHAR(30) UNIQUE,

endereco TEXT,

telefone VARCHAR(20),

email VARCHAR(100)

);

-- 👥 USUÁRIOS

CREATE TABLE usuarios (

id SERIAL PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

senha\_hash TEXT NOT NULL,

cargo VARCHAR(50), -- admin, técnico, usuário, etc.

nivel\_acesso INTEGER DEFAULT 1, -- 1=comum, 2=técnico, 3=admin

instituicao\_id INTEGER NOT NULL REFERENCES instituicoes(id) ON DELETE CASCADE

);

-- 🏠 ESPAÇOS

CREATE TABLE espacos (

id SERIAL PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

tipo VARCHAR(50), -- sala, laboratório, auditório, etc.

capacidade INTEGER,

recursos JSONB, -- exemplo: {"projetor": true, "AC": false}

instituicao\_id INTEGER NOT NULL REFERENCES instituicoes(id) ON DELETE CASCADE

);

-- 📅 RESERVAS

CREATE TABLE reservas (

id SERIAL PRIMARY KEY,

espaco\_id INTEGER NOT NULL REFERENCES espacos(id) ON DELETE CASCADE,

usuario\_id INTEGER NOT NULL REFERENCES usuarios(id) ON DELETE CASCADE,

data\_inicio TIMESTAMP NOT NULL,

data\_fim TIMESTAMP NOT NULL,

finalidade TEXT,

status VARCHAR(20) DEFAULT 'pendente' -- pendente, confirmada, cancelada

);

-- 🔑 CHAVES

CREATE TABLE chaves (

id SERIAL PRIMARY KEY,

codigo\_identificador VARCHAR(50) UNIQUE NOT NULL,

espaco\_id INTEGER REFERENCES espacos(id), -- opcional: (0,1) cardinalidade

disponivel BOOLEAN DEFAULT TRUE,

observacoes TEXT

);

-- 🔄 MOVIMENTAÇÕES DE CHAVES

CREATE TABLE movimentacoes\_chaves (

id SERIAL PRIMARY KEY,

chave\_id INTEGER NOT NULL REFERENCES chaves(id) ON DELETE CASCADE,

usuario\_id INTEGER REFERENCES usuarios(id) ON DELETE SET NULL,

data\_retirada TIMESTAMP NOT NULL,

data\_devolucao TIMESTAMP,

responsavel\_entrega INTEGER REFERENCES usuarios(id) ON DELETE SET NULL,

responsavel\_recebimento INTEGER REFERENCES usuarios(id) ON DELETE SET NULL,

observacoes TEXT

);

-- 📦 ITENS DE INVENTÁRIO

CREATE TABLE itens\_inventario (

id SERIAL PRIMARY KEY,

nome\_item VARCHAR(100) NOT NULL,

descricao TEXT,

numero\_patrimonio VARCHAR(50) UNIQUE,

localizacao\_atual VARCHAR(100),

quantidade INTEGER DEFAULT 1,

estado\_conservacao VARCHAR(30), -- novo, bom, danificado, etc.

data\_aquisicao DATE,

instituicao\_id INTEGER NOT NULL REFERENCES instituicoes(id) ON DELETE CASCADE

);

-- 📜 LOGS DE USO

CREATE TABLE logs\_uso (

id SERIAL PRIMARY KEY,

usuario\_id INTEGER REFERENCES usuarios(id) ON DELETE SET NULL,

acao TEXT NOT NULL, -- ex: reserva criada, chave retirada, etc.

entidade VARCHAR(50) NOT NULL, -- ex: reservas, chaves, inventario, etc.

entidade\_id INTEGER NOT NULL,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

