SENAI – SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

ESCOLA SENAI “ROBERTO MANGE”

CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

**ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE**

**“SMARTY\_CITY”**

Emily Vitória de Souza Ramos

Ds 14

Campinas – SP

SMART CITY

**Documento de Requisitos de Software**

****

**Especificação de Requisitos de Software**

# Introdução

* Objetivo: Este Documento de Especificação de Requisitos de Software (SRS) tem como objetivo definir todos os requisitos funcionais e não funcionais da plataforma web "Smarty\_City". A plataforma visa monitorar dados simulados de sensores (temperatura, umidade, luminosidade, contador de pessoas) de ambientes urbanos inteligentes, com funcionalidades de CRUD, visualização de históricos, localização no mapa, exportação em Excel e autenticação via login obrigatório.
* Escopo: O sistema Smarty\_City é uma plataforma interativa com login obrigatório para acesso às funcionalidades. Ele permite a visualização e gestão de dados em tempo real, vindos de sensores simulados, com interface responsiva. Inclui funcionalidades de CRUD para sensores, ambientes e históricos, mapa com localização via latitude e longitude, filtro por datas e exportação para Excel. O sistema usa banco de dados MySQL, construído com Django Rest Framework (DRF), front-end em React com Tailwind e gráficos interativos com Chart.js.

Principais funcionalidades incluem:

* **Dashboard** com gráficos interativos (Chart.js);
* **CRUD** completo para sensores, ambientes e históricos;
* **Mapa** com localização dos sensores (latitude/longitude);
* **Filtros** de busca e exportação para Excel (na página de sensores);
* **Login e cadastro** obrigatório para uso da plataforma;
* **Responsividade** e acessibilidade visual com base na paleta de cores definida.

Visão Geral do Produto

* 1. Perspectiva do Produto

A plataforma Smarty\_City centraliza dados de sensores simulados para fornecer visualizações inteligentes que apoiem decisões sobre o ambiente urbano. O projeto tem como foco a gestão eficiente de ambientes através de análise de históricos, integração com mapas e experiência de usuário fluida.

* 1. Funções do Produto
* Visualização de gráficos de sensores (temperatura, umidade, luminosidade, contador);
* CRUD para sensores, ambientes e históricos;
* Cadastro e login obrigatório;
* Mapa com localização de sensores;
* Exportação de dados em Excel;
* Filtro por intervalo de datas;
* Interface responsiva com design baseado nas cores:
  + #F5F5F5 (fundo),
  + #226D13 (texto),
  + #ffffff (contraste),
  + #4262FF (botões acessar),
  + #C72F2F (botões sair/excluir),
  + #D9D9D9 (inputs).
  1. Restrições
* A plataforma só funciona com usuários autenticados;
* Não há distinção entre tipos de usuário (admin/comum);
* Banco de dados MySQL (WorkBench);
* Gráficos renderizados com Chart.js;
* Exportação Excel exclusiva na página de sensores.

# Descrição dos Requisitos

* Requisitos Funcionais:
* Requisitos Não Funcionais:

## Requisitos Funcionais:

### RF01 - Login obrigatório para acesso a qualquer funcionalidade.

### RF02 - Cadastro de novo usuário.

### RF03 - Visualização de dashboard com gráficos (por sensor).

**RF04 -** CRUD de sensores

**RF05 -** CRUD de ambientes.

**RF06** - CRUD de histórico dos sensores.

### RF07 – Mapa com localização dos sensores.

### RF08 - Exportação de dados para Excel.

### RF09 - Filtro de histórico por intervalo de datas.

### RF10 - Notificações de sucesso e erro para ações do usuário.

## Requisitos Não Funcionais:

### RF01 – Interface responsiva com design padronizado.

### RF02 – Compatibilidade com Chrome, Firefox e Edge.

### RF03 – Todas as páginas devem carregar em até 2 segundos.

**RF04 –** Dados protegidos com JWT (login).

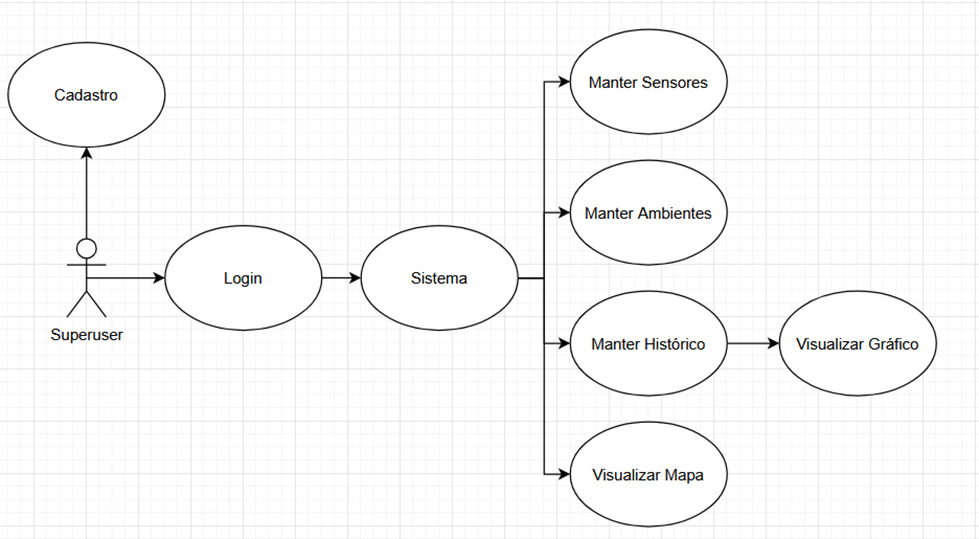
**RF05 –** O sistema sempre online (exceto manutenção programada).

**RF06** – O sistema deve suportar até 10.000 acessos simultâneos.

### RF07 – Toda interação deve ter retorno visual (sucesso/erro).

# Casos de Uso

Descrição: Esta seção descreve, em termos de "atores" e suas interações com o sistema, como o software será usado. Cada caso de uso oferece cenários que ilustram o fluxo da atividade, ajudando assim no entendimento e na construção do sistema.



Caso de Uso 1: Login do Usuário

* Ator Principal: Qualquer usuário.
* Fluxo: Usuário informa e-mail e senha > Backend verifica credenciais > Front redireciona para página Home.

Caso de Uso 2: Visualização de Gráficos

* Ator Principal: Usuário Logado
* Fluxo: Usuário acessa Home > Backend retorna dados > Chart.js renderiza gráfico

Caso de Uso 3: Exportar dados

* Ator Principal: Usuário logado.
* Fluxo: Usuário clica em exportar > Backend envia aqruivo Excel > front faz dowload.

Caso de Uso 4: Cadastro de Sensor

* Ator Principal: Usuário logado.
* Fluxo: Usuário preenche dados > Clica em salvar > Backend salva sensor > Front exibe confirmação.

# Matriz de Rastreabilidade

Descrição: A Matriz de Rastreabilidade é uma tabela usada para rastrear os requisitos ao longo do ciclo de vida de um projeto. Ela serve para entender a relação entre os requisitos e os componentes do sistema que os implementam, assim como outros artefatos do projeto, como casos de teste ou documentos.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição** | **Caso de Uso** | **Componente** |
| RF01 | Login obrigatório | Login do Usuário | Auth (JWT) |
| RF03 | Dashboard com gráficos | Visualização de Gráficos | Chart.js/DRF |
| RF08 | Exportar Excel | Exportar dados | Excel API |
| RF04 | CRUD de sensores | Cadastro de Sensor | Django DRF |

# Autor Integrador

|  |
| --- |
| **Nome** |
| Emily Vitória de Souza Ramos DS14 |
|  |
| **Links** |
|  |
| https://www.figma.com/design/AZIJA85GnZC7FiVd074EoH/SmartCity?node-id=0-1&t=O8nyqgDe3ok7MAAs-1 |
| https://github.com/ramos-emily/Integrador.git |
|  |