**PARA RODAR**

Abra o prompt do Windows, navegue até a pasta do código e digite:

**python -m http.server**

para criar um servidor web na sua máquina.

Depois acesse no navegador:

# Exemplo 1 – Atualização de Texto HTML

<http://localhost:8000/test_ScreenHandler.html>

Interface gráfica do usuário, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Esse exemplo atualiza os dados de cada TAG e exibe os valores atuais no campo de texto.

**🧩 Visão Geral da Aplicação**

Esta aplicação é uma interface web simples para exibir dados associados a TAGs (como sensores ou variáveis de processo). Ela é composta por três partes:

**1. dataHandler.js – Lógica de Dados**

Define a classe dataHandler, responsável por:

* Armazenar e gerenciar uma lista de objetos com estrutura {Tag, Type, Data, Alarm, DateTime}.
* Atualizar os dados com validações (por exemplo, garantir que Alarm seja número e DateTime seja uma data válida).
* Fornecer métodos para acessar os dados (getDados, getDadoByTag).
* Fornecer um método updateScreen() como template para ser sobrescrito por subclasses.

**2. screenHandler.js – Lógica de Interface**

Define a classe ScreenHandler, que herda de dataHandler e implementa a atualização da interface:

* Sobrescreve updateScreen() para exibir os dados no HTML.
* Instancia a classe com TAGs predefinidas (['Temperatura', 'Tag2']).
* Atualiza dinamicamente o conteúdo de um parágrafo HTML (outputParagraph.textContent) com os valores das TAGs.

**3. test\_ScreenHandler.html – Interface HTML**

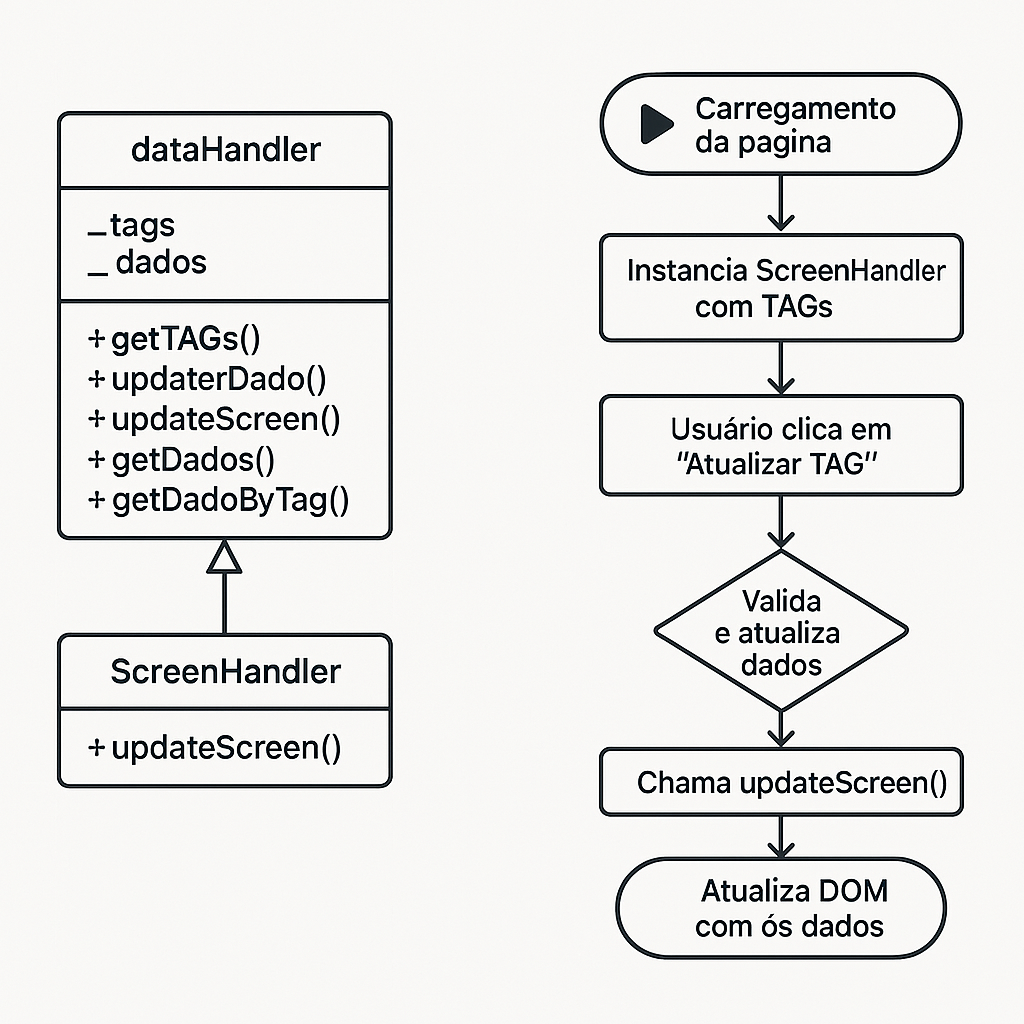
HTML básico com:

* Um botão “Atualizar TAG”.
* Um parágrafo onde os dados atualizados são exibidos.
* O script screenHandler.js é carregado e executado ao carregar a página.

**🔄 Fluxo de Funcionamento**

1. Ao carregar a página, o evento DOMContentLoaded instancia ScreenHandler com TAGs iniciais.
2. A instância é armazenada em window.myScreenHandler para acesso global.
3. Quando os dados são atualizados via updateDado(), o método updateScreen() é chamado automaticamente.
4. A tela é atualizada com os valores atuais das TAGs.

**📊 Diagramas**

****

**OBS. Para usar, edite ou crie outro arquivo ScreenHandler.js, o qual deve ser carregado na página HTML.**

**Edite as TAGs nesse arquivo, e escreva o código do método updateScreen() para atualizar os dados na página. Utilize na página HTML texto simples, gráficos, tabelas, webcomponents, etc.**

# Exemplo 2 – Dashboard com Plotly

O exemplo test\_dashboard\_plotly,html implementa um pequeno dasboard com 2 gauges. Uma lógica foi adicionada para colorir o gauge de Temperatura para vermelho caso a temperatura ultrapasse o valor de 80. Ambos os Gauges possuem escala de 0 a 100.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Diferenças em relação à aplicação anterior com apenas texto.  
  
**🔍 Principais Diferenças**

| **Aspecto** | **Versão Anterior (screenHandler.js)** | **Nova Versão (test\_dashboard\_plotly.js)** |
| --- | --- | --- |
| **Visualização** | Atualização de texto simples em um parágrafo HTML | Visualização gráfica com indicadores do Plotly |
| **Biblioteca Gráfica** | Nenhuma | Usa Plotly.js para criar gauges (medidores) |
| **Atualização de Tela** | Atualiza texto com valores das TAGs | Atualiza gráficos com valores e cores dinâmicas |
| **Lógica de Alerta** | Nenhuma | Alerta visual: cor vermelha se temperatura ≥ 80°C |

**🧠 Destaques Técnicos da Nova Versão**

**📊 Visualização com Plotly**

* Cria dois indicadores do tipo gauge (medidores circulares) para:
  + **Temperatura** (com alerta visual)
  + **Pressão**
* Layout com duas colunas e um grid independente.
* Atualização dinâmica com Plotly.react().

**🧱 Estrutura Modular**

* A classe ScreenHandler continua herdando de dataHandler.
* A lógica de atualização (updateScreen) agora manipula objetos plotly\_data em vez de texto.

**🧪 HTML de Teste**

* O arquivo test\_dashboard\_plotly.html é minimalista, com apenas um título e placeholders para TAGs e o dashboard.
* O HTML deve carregar o Plotly como script.