UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

OMAR CHOQUEHUANCA BELTRAN

Estudo de Caso: Desenvolvimento de uma aplicação para um sistema de limpeza do tipo CIP utilizando a plataforma *HMI* da *Siemens*

Sorocaba
2025

OMAR CHOQUEHUANCA BELTRAN

Estudo de Caso: Desenvolvimento de uma aplicação para um sistema de limpeza do tipo CIP utilizando a plataforma *HMI* da *Siemens*

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba, como parte dos requisitos para obtenção do grau de Bacharel(a) em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Everson Martins

Sorocaba

2025

B453e

Beltran, Omar Choquehuanca
Estudo de caso : desenvolvimento de uma aplicação para um sistema de limpeza do tipo CIP utilizando a plataforma HMI da Siemens / Omar Choquehuanca Beltran. -- Sorocaba, 2025
145 p. : il., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba
Orientadora: Everson Martins

1. Software de aplicação. 2. Interfaces de usuário (Sistema de computador). 3. Sistemas homem-máquina. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

OMAR CHOQUEHUANCA BELTRAN

Estudo de Caso: Desenvolvimento de uma aplicação para um sistema de limpeza do tipo CIP utilizando a plataforma *HMI* da *Siemens*

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel(a) em Engenharia de Controle e Automação.

Data da defesa: 18/06/2025

BANCA EXAMINADORA:

Prof. Dr. Everson Martins
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Prof.ª Dr.ª Maria Glória Caño de Andrade
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

Prof.Dr. Eduardo Verri Liberado
UNESP – Instituto de Ciência e Tecnologia – Campus de Sorocaba

AGRADECIMENTOS

Agradecimentos à Instituição que me acolheu, desafiou e transformou, não apenas em um profissional qualificado, mas em uma pessoa melhor e mais completa.

O presente trabalho foi realizado com apoio da empresa VPA Automação que ofereceu tempo e recursos para o desenvolvimento.

Agradeço aos professores que me acompanharam ao longo do curso e que, com empenho, se dedicam à arte de ensinar.

Agradeço ao professor/doutor Everson Martins, pela orientação acadêmica, apoio e confiança.

RESUMO

Em conjunto com a empresa colaboradora, foi desenvolvido um projeto para um sistema de limpeza do tipo CIP. Este projeto foi dividido em duas equipes, uma para o desenvolvimento do código do PLC e outra para o desenvolvimento das telas de visualização e controle da máquina. Na segunda equipe foi utilizando uma HMI e os softwares TIA Portal, para o desenvolvimento das animações; e o PAINT.NET para a criação das imagens dos equipamentos. Com estas ferramentas foi desenvolvido todas as telas principais de operação do projeto e através da metodologia de parametrização vetorial de equipamentos, além de criar uma biblioteca de objetos de animação para representação e controle do sistema de limpeza. Todo o progresso das telas de animação, foi de acordo com as memórias disponíveis pela equipe do PLC. O texto produzido neste documento, procura explicar todos os passos decorridos ao longo do projeto, além de detalhar como foi desenvolvido e parametrizado estes objetos de animação. Este projeto conta também com receitas de produção, que estavam no desenvolvimento da equipe do PLC, mas terá sua animação de passo a passo criada pela equipe da HMI.

Palavras-chave: HMI; PLC; código; animação.

ABSTRACT

In conjunction with the collaborating company, a project for a CIP cleaning system was developed. This project was divided into two teams, one for the development of the PLC code and the other for the development of the machine's visualization and control screens. The second team used an HMI and TIA Portal software to develop the animations; and PAINT.NET to create the equipment images. With these tools, all the project's main operation screens were developed using the equipment vector parameterization methodology, in addition to creating a library of animation objects for representing and controlling the cleaning system. All the progress of the animation screens was in accordance with the memories available to the PLC team. The text produced in this document seeks to explain all the steps taken throughout the project, in addition to detailing how these animation objects were developed and parameterized. This project also includes production recipes, which were developed by the PLC team, but will have their step-by-step animation created by the HMI team.

Keywords: HMI; PLC; code; animation.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CIP	*Clean in Place*
HMI	*Human Machine Interface*
CLP	Controlador Lógico Programável
IP	Internet Protocol
AS-i	Actuator-Sensor Interface
TIA Portal	Totally Integrated Automation Portal
DB	*Data Base*
Tag	Identificação do equipamento
PLC	*Programmable Logic Controller*

SUMÁRIO

1. INTRODUÇÃO

A empresa cliente, onde foi implementado a solução do projeto, vinha realizando o processo de limpeza de suas tubulações e equipamentos manualmente. O que acarretava num tempo de ciclo muito grande, especificamente para limpeza, pois tinham que se assegurar que tudo estivesse limpo para que não ocorresse contaminação com a produção do próximo produto.

E, para realizar a limpeza completa desses equipamentos, só podia ocorrer de duas formas: ou se desmontavam todos os equipamentos e realizava a limpeza manualmente, ou utilizava a própria estrutura física do equipamento para realizar uma limpeza por condições extremas - por exemplo, com pressão de ar comprimido para remover todo resíduo químico remanescente das tubulações. Nesse último caso, porém, é necessário o monitoramento constante de todos os equipamentos envolvidos no processo, pois qualquer tipo de falha pode acarretar num acidente com o operador.

O documento apresentará um estudo de caso de um projeto já realizado pela empresa, no qual o aluno presta serviços. O projeto tem como objetivo automatizar o processo de limpeza do tipo Clean-in-Place (CIP), um método que visa no custo-benefício e eficiência na limpeza de tubulações de passagem de matéria prima e/ou produto.

Utilizando os recursos e conhecimentos obtidos pela empresa de automação, duas equipes participaram na produção deste projeto: uma equipe focada na criação da lógica do controlador do processo e outra no desenvolvimento de uma interface homem-máquina, com o objetivo de centralizar todo o controle e visualização das etapas da limpeza de tubulações.

A trabalho teve como objetivo descrever os métodos de desenvolvimento da interface home-máquina do projeto. Essa interface foi implementada com uma tela touchscreen da empresa Siemens. Esse modulo conta com uma biblioteca de animações padrões e um código aberto para criação de novas animações, o que foi o foco da solução proposta pelo projeto.

2. REVISÃO BIBLIOGRÁFICA

2.1. Clean-in-Place (CIP)

Com o avanço das indústrias na produção de diversos materiais de consumo, a quantidade de resíduos contaminantes também aumentou consideravelmente. Por essa razão, muitas indústrias têm adotado diferentes métodos de limpeza para seus equipamentos de produção. Nesse contexto, quando se trata de uma indústria com processos automatizados, é evidente a necessidade de automatizar também os processos de limpeza.

O sistema CIP é amplamente adotado nas indústrias devido ao seu foco no custo-benefício e na eficiência dos processos de limpeza. Ele permite a higienização de tubos, tanques e equipamentos de processo, como válvulas e filtros, sem a necessidade de desmontagem dos componentes, tornando o procedimento mais prático e eficaz (HIGTOP, 2024).

O uso contínuo ou periódico deste sistema proporciona maior eficiência na produção, assegurando segurança e qualidade dos produtos. Além disso, destaca-se o impacto ambiental positivo do método, que reduz o desperdício de água, diminui o consumo de detergentes e economiza tempo durante o processo de limpeza.

Uma das ferramentas mais importantes utilizadas nesses sistemas é o PIG (Pipeline Inspection Gadget). Trata-se de um projétil sólido e de fácil rastreamento, capaz de atravessar válvulas, tubulações e diversas conexões, incluindo curvas e ramificações. Seu uso viabiliza a recuperação do produto remanescente nas tubulações, reduzindo os custos associados aos sistemas CIP e permitindo uma troca mais ágil entre variantes de produtos.

2.2. Human Machine Interface (HMI)

É evidente que processos automatizados em indústrias exigem códigos complexos para configurar os equipamentos, independentemente da linguagem de programação utilizada. Contudo, na maioria das vezes, as pessoas responsáveis por operar esses equipamentos não são as mesmas que desenvolvem os códigos.

Por esta razão, no setor de automação industrial, além dos sofisticados códigos de programação, também são criados mecanismos ou softwares que facilitam a interação entre o operador e a máquina. Esses sistemas são projetados com o objetivo de tornar o gerenciamento e o controle dos processos mais intuitivos e acessíveis (*Siemens*, 2022).

Frequentemente, esses mecanismos assumem a forma de tablets de visualização ou computadores de monitoramento, mas são amplamente conhecidos como interfaces homem-máquina (HMI). Um exemplo de uma HMI pode ser observado na Figura 1.

Figura 1 - Exemplo de HMI.

Fonte: Siemens Product Catalog.

O modelo de HMI utilizado neste trabalho é o "*MTP1000 Unified Comfort*", na versão de *firmware* V18. Este modelo, diferentemente dos demais disponíveis no portfólio da Siemens, oferece uma liberdade significativamente maior na criação de telas e animações. No entanto, essa flexibilidade também apresenta um desafio inicial, pois, em projetos novos, ele não fornece blocos de funções ou telas pré-configuradas. Isso torna necessário um estudo aprofundado em diversos fóruns para o desenvolvimento adequado deste projeto.

2.3. Linguagem *JavaScript*

Um dos principais fóruns consultados durante o desenvolvimento do projeto da HMI foram aqueles relacionados à linguagem *JavaScript*, especialmente os tópicos que abordam as linhas de código específicas no formato de *JavaScript* que a Siemens incorporou na linha de HMI *Unified* (*Siemens*, 2020)

Em 1995, com o avanço acelerado do desenvolvimento de páginas web, a *Netscape Communications* contratou Brendan Eich para trabalhar na incorporação da linguagem de

programação *Scheme* em seu navegador. Contudo, com a colaboração da *Sun Microsystems* e a utilização do Java – uma linguagem de programação mais estática –, foi criado o primeiro script que permitia a execução de animações em páginas web, sem a dependência de um servidor.

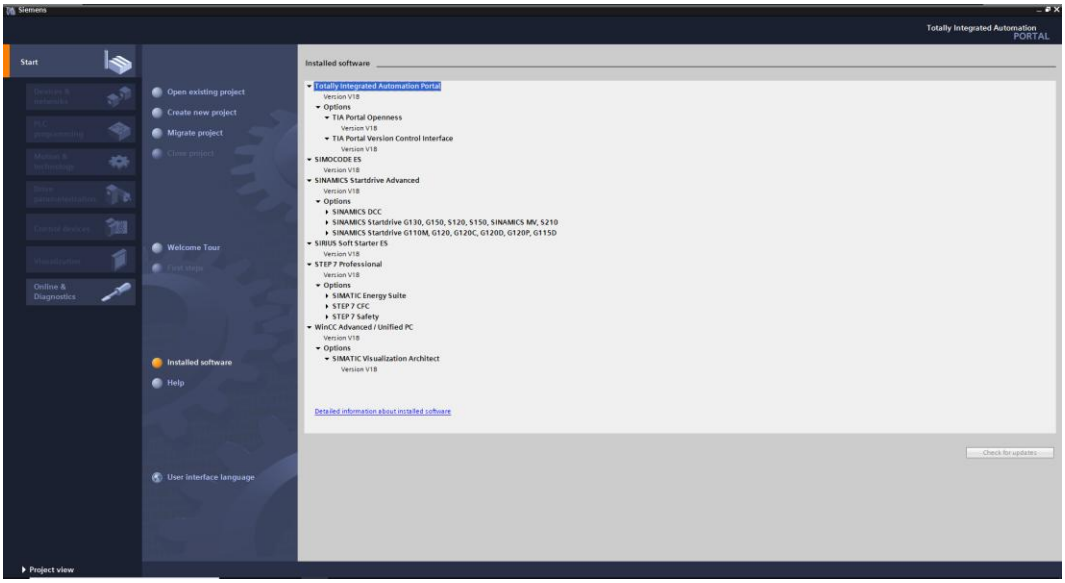
Posteriormente, a linguagem foi renomeada como *JavaScript* e tornou-se essencial para todos os navegadores web, permitindo a execução de scripts no lado do cliente e possibilitando interações dinâmicas sem a necessidade de um servidor. Isso viabilizou conexões assíncronas e alterações diretas no conteúdo exibido.

Reconhecendo a crescente popularidade dessa nova linguagem de programação, a *Netscape* submeteu o *JavaScript à ECMA International* para estabelecer uma padronização. Esse esforço visava permitir que outros navegadores implementassem a linguagem com base no trabalho desenvolvido pela *Netscape*. A primeira versão padronizada, chamada *ECMAScript*, foi lançada em junho de 1997 e continua a receber atualizações e aprimoramentos até os dias atuais (JavaScript, 2024).

2.4. Software TIA Portal

TIA Portal ou portal de automação totalmente integrado, é uma plataforma de programação da empresa *Siemens*, que tem como objetivo integrar todos os equipamentos disponíveis da *Siemens*, sejam eles controladores, HMIs, inversores, sensores entre outros (*Siemens*, 2024).

Figura 2 - Software TIA Portal V18.

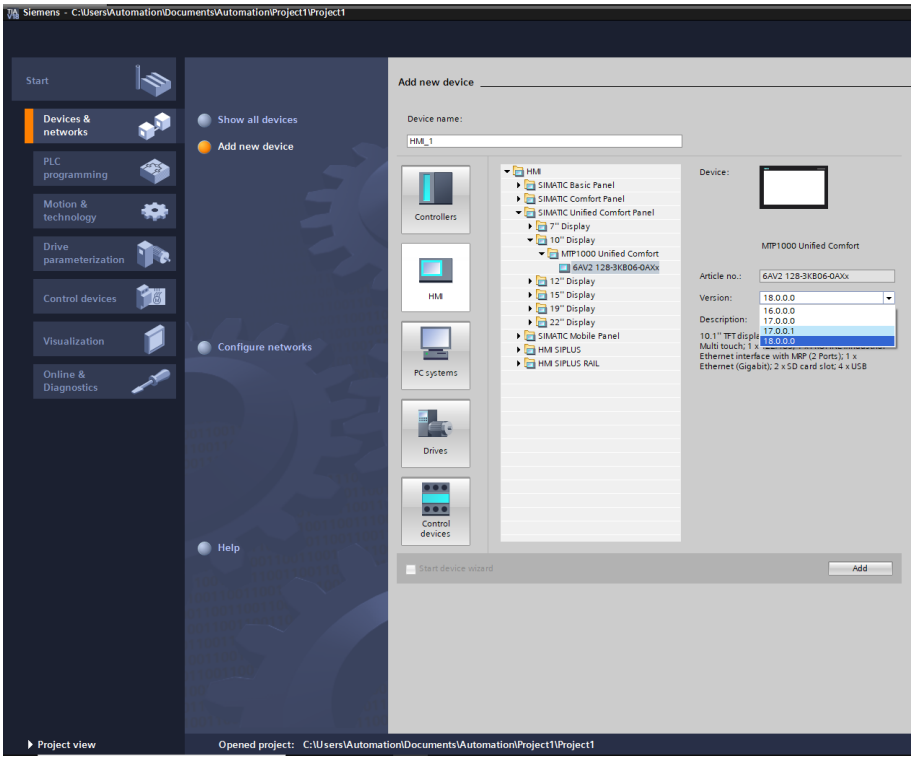


Fonte: Autoria própria.

Este software possui diversas versões, sendo fundamental verificar a compatibilidade da aplicação desenvolvida com os equipamentos a serem utilizados, como HMIs ou controladores. Para garantir essa compatibilidade, é necessário consultar as pastas de equipamentos disponíveis para adicionar ao projeto, conforme ilustrado na Figura 2. No caso deste projeto específico, a versão 18 do software foi escolhida devido à correspondência com a versão de *firmware* da HMI.

É importante destacar que cada equipamento a ser adicionado ao projeto possui um número de série de identificação, geralmente localizado na embalagem ou na parte traseira do dispositivo. Em muitos casos, o projeto especifica os números de identificação exatos ou um intervalo correspondente, como exemplificado na Figura 3, que mostra a adição da HMI ao projeto.

Figura 3 - Seleção de equipamento no projeto.



Fonte: Autoria própria.

2.5. Linguagem *Ladder*

Os primeiros CLPs (Controladores Lógicos Programáveis) começaram a surgir a partir da década 60 com o objetivo de diminuir os grandes painéis de controle baseados em relés. Sua programação era fundamentada na linguagem chamada *Ladder*, uma ferramenta gráfica para

coletar todos os dados de seus módulos de entrada e – dependendo do nível da logica criada – gerar uma resposta em seus módulos de saída (Kalatec, 2025).

Suas funções de programação são intuitivas e simplificadas, baseando-se na lógica de relés e apresentando grande semelhança com circuitos elétricos. Um exemplo básico pode ser ilustrado por um contato de entrada que aciona uma bobina de saída como resposta.

Cada lógica desenvolvida pode conter diversas linhas de programação e segue uma sequência específica de verificação, que varia conforme o fabricante do controlador. No entanto, o padrão mais comum é a leitura da esquerda para a direita e de cima para baixo.

Neste projeto, toda a lógica do código do controlador do processo de limpeza foi criada por outra equipe de automação. Porém, como a interface homem-máquina necessita de uma conexão física com o processo para trocar as informações, foi criado um padrão de encapsulamento dos dados do processo de limpeza.

Esse mesmo padrão foi implementado no desenvolvimento da interface, para que não haja problemas de comunicação e controle do processo.

3. METODOLOGIA

3.1. Atualização da HMI

Inicialmente, foi necessário realizar uma atualização de *firmware* da HMI para a versão V18, que era a mais recente disponível durante o desenvolvimento do trabalho. Esse procedimento foi realizado utilizando um *pendrive* conectado na parte traseira do equipamento.

O *pendrive* deve conter exclusivamente o arquivo de *firmware* apropriado para o equipamento. Após energizar a HMI, é possível acessar os arquivos armazenados no *pendrive* por meio da janela "Update OS", ilustrada na Figura 181 do Apêndice I.

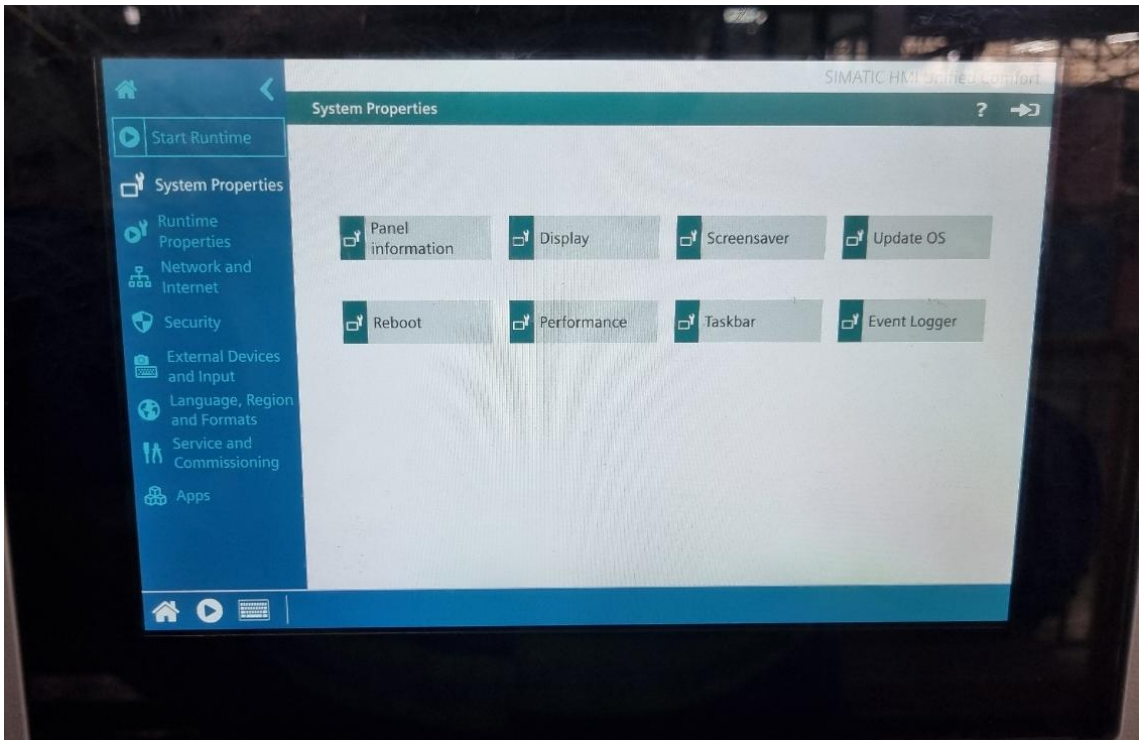
Nesta janela da HMI, é possível visualizar a versão atual de *firmware* e – uma vez apontado o *pendrive* na mídia externa – pode-se selecionar o arquivo para realizar a atualização de *firmware*; vale ressaltar que é muito importante garantir que o equipamento esteja energizado durante todo o procedimento de atualização de *firmware*, pois qualquer queda de energia durante, pode comprometer o equipamento.

Na janela da HMI, é possível verificar a versão atual do *firmware* e, ao selecionar o *pendrive* como mídia externa, escolher o arquivo adequado para realizar a atualização. É essencial garantir que o equipamento permaneça energizado durante todo o procedimento, pois qualquer interrupção no fornecimento de energia pode comprometer o equipamento.

3.2. Janelas de acesso da HMI

Após a atualização do *firmware* da HMI, é necessário navegar pelas janelas de configuração para definir o IP de comunicação e nomear o equipamento, facilitando sua identificação no processo de descarregamento do projeto para o dispositivo.

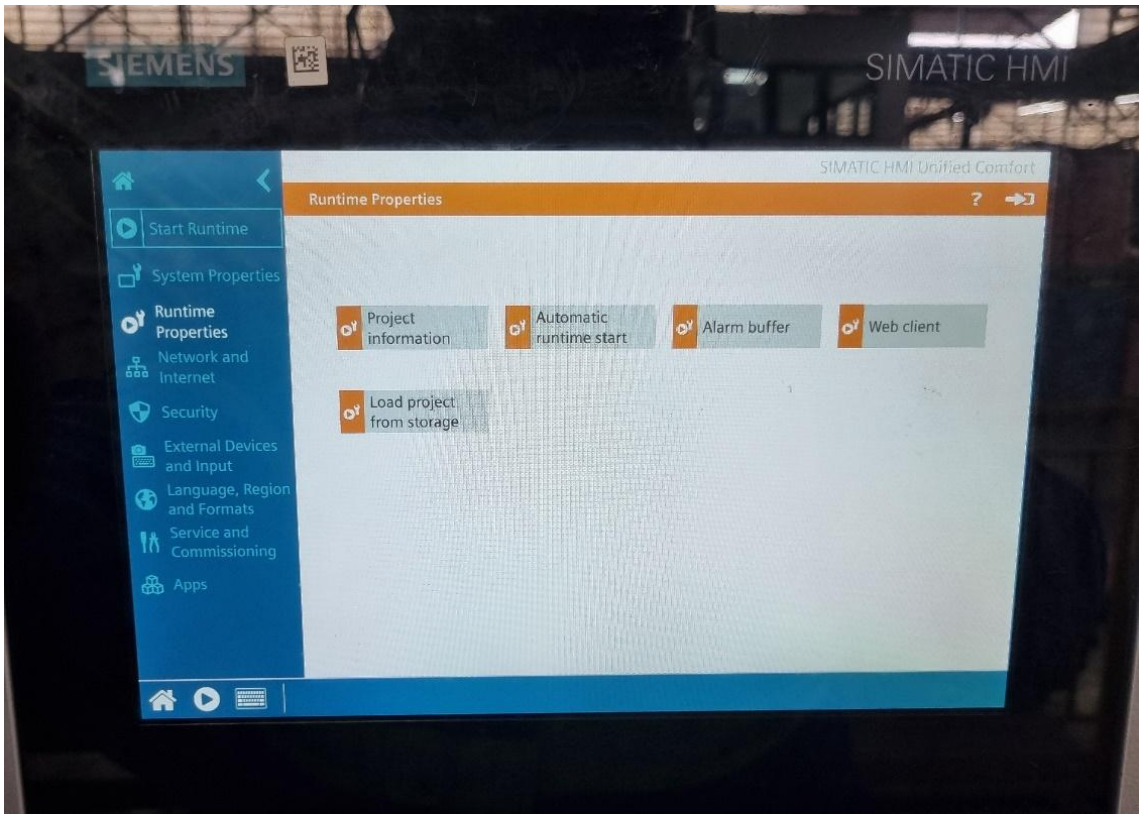
Figura 4 - Janela de acesso as propriedades do sistema da HMI.



Fonte: Autoria própria.

Na Figura 4, são exibidas as janelas de propriedades da HMI, onde é possível acessar informações do sistema, configurações de brilho e orientação da tela, ajustes de tempo de ativação da tela, atualização de *firmware*, configurações de *reboot*, configuração de alarme de performance, configuração dos botões flutuantes e exportação de logs do equipamento. As imagens de visualização para cada aba de configuração estão disponíveis no Apêndice I.

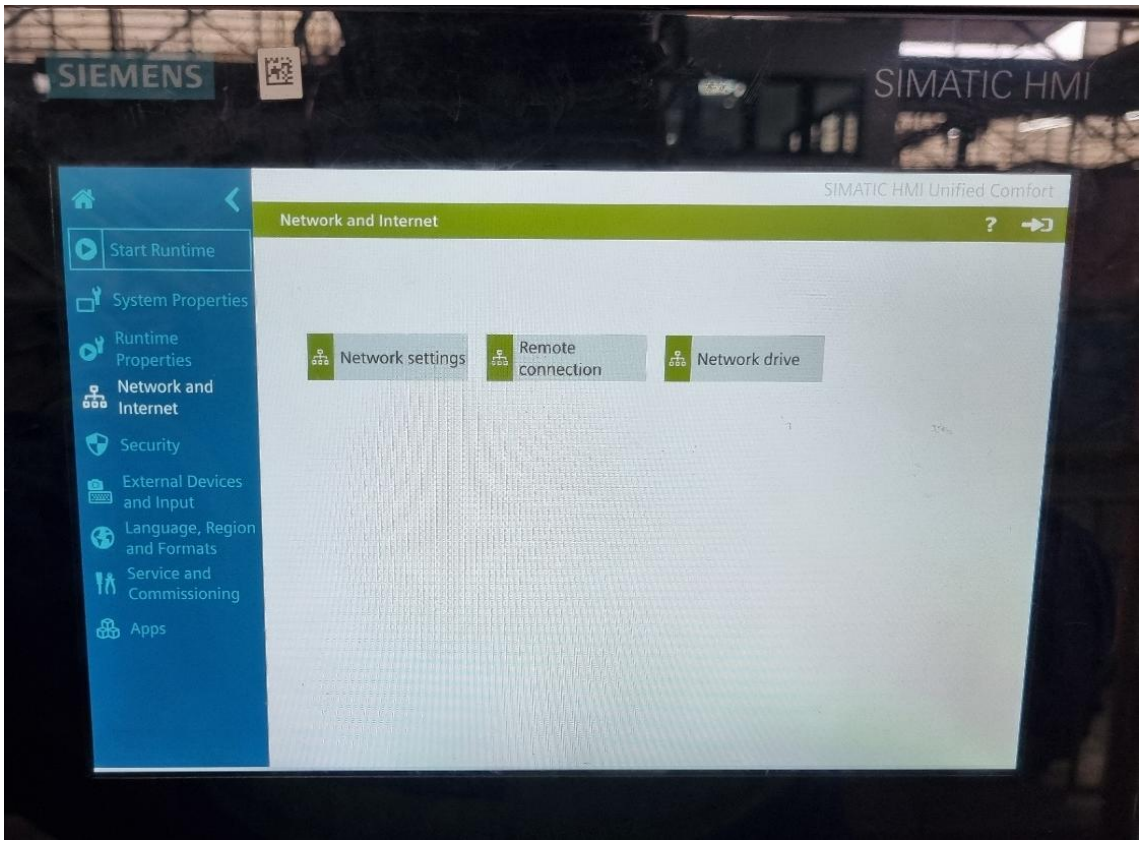
Figura 5 - Janela de acesso as propriedades do *runtime* da HMI.



Fonte: Autoria própria.

Na Figura 5, são apresentadas as janelas relacionadas à aplicação do projeto, onde é possível acessar informações do projeto, configurar o início automático, ajustar o *buffer* de alarme, ativar o acesso *web* e carregar o projeto por meio de memória externa. Todas as visualizações dessas janelas estão disponíveis no Apêndice II.

Figura 6 - Janela de acesso as propriedades de *Network and Internet* da HMI.



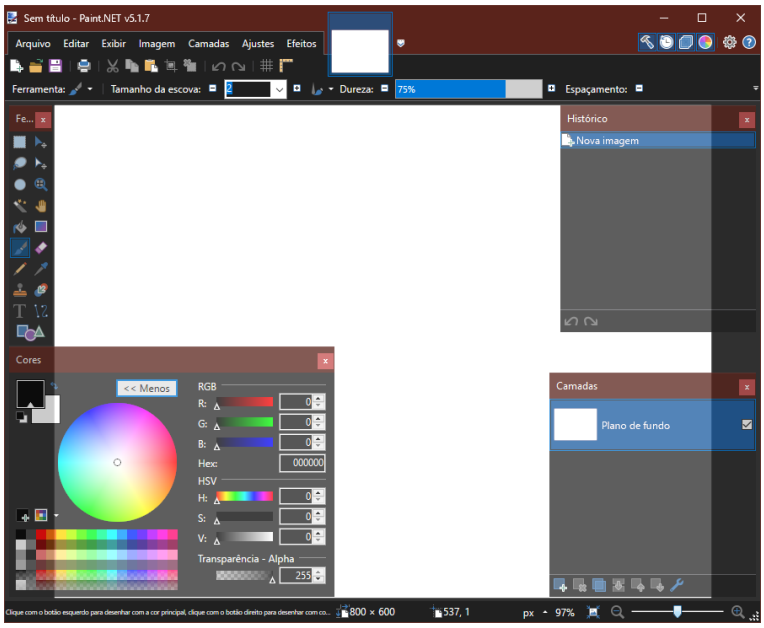
Fonte: Autoria própria.

Na Figura 6, é apresentada a janela de configuração de rede do equipamento, onde se encontram as abas para definir o IP, configurar o acesso remoto aos botões da HMI e ajustar o drive de comunicação. Todas as visualizações dessas janelas estão disponíveis no Apêndice III.

3.3. Criação de imagens para animação

Para o desenvolvimento das animações da HMI, utilizou-se o programa *"Paint.NET"* para a criação e modificação de imagens. Por ter ferramentas muito parecidas com o programa popularmente conhecido como *"Paint"*, sua utilização se tornou mais eficaz em comparação com outros programas de edição de imagem.

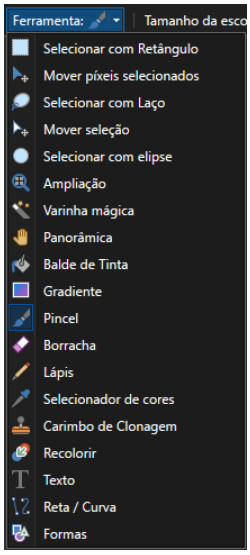
Figura 7 - Programa *"Paint.NET"*.



Fonte: Autoria própria.

As imagens utilizadas na HMI foram editadas no formato ".png" e, com exceção da imagem de fundo, todas as imagens dos objetos tiveram seu plano de fundo removido. Dessa forma, os objetos eram isolados para se adaptarem melhor às animações. Para a edição das imagens, foram utilizadas principalmente as ferramentas demonstradas na Figura 8.

Figura 8 - Conjunto de ferramentas principais de edição de imagem.

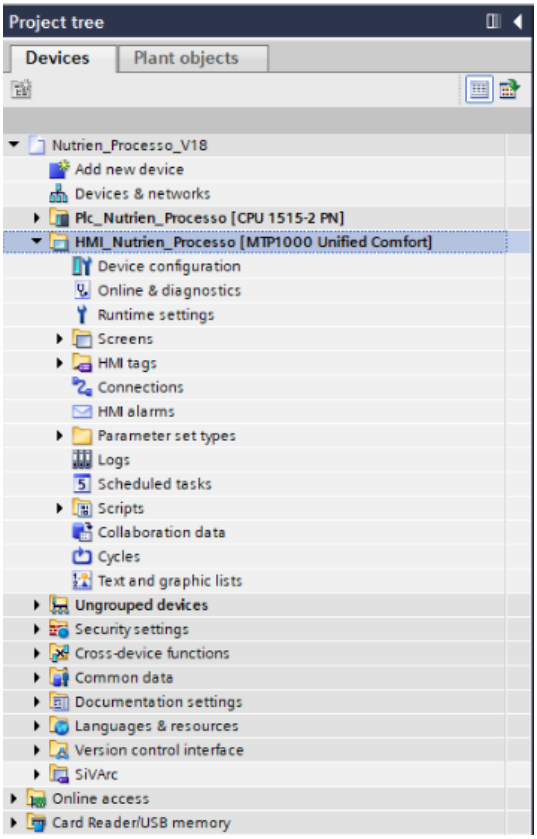


Fonte: Autoria própria.

3.4. Árvore do projeto

Ao navegar pelo programa *"TIA Portal"*, acessa-se a primeira área de trabalho principal para o desenvolvimento da HMI: a árvore do projeto, onde estão listados todos os equipamentos programáveis conectados à rede de comunicação.

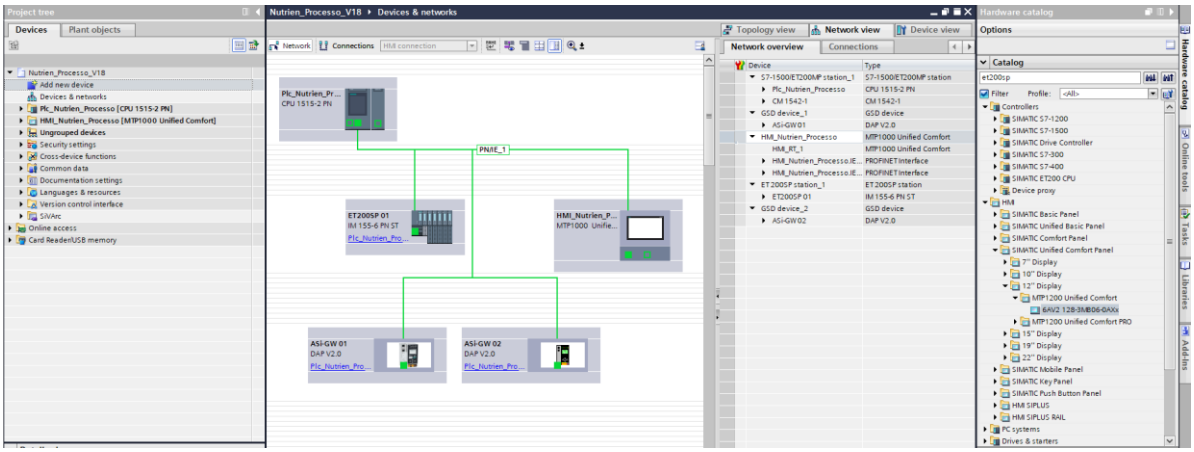
Figura 9 - Visão geral da árvore do projeto.



Fonte: Autoria própria.

Neste projeto, foram integrados os seguintes dispositivos: o PLC, a IHM, a estação remota *Simatic ET200SP* — responsável pelo gerenciamento das entradas e saídas dos sensores — e duas cabeças de rede *AS-i*, destinadas ao controle e monitoramento do estado das válvulas pneumáticas do equipamento. A comunicação entre os dispositivos é realizada por meio do protocolo Profinet, cuja configuração é efetuada na aba *"Devices & Networks"*, conforme ilustrado na Figura 10.

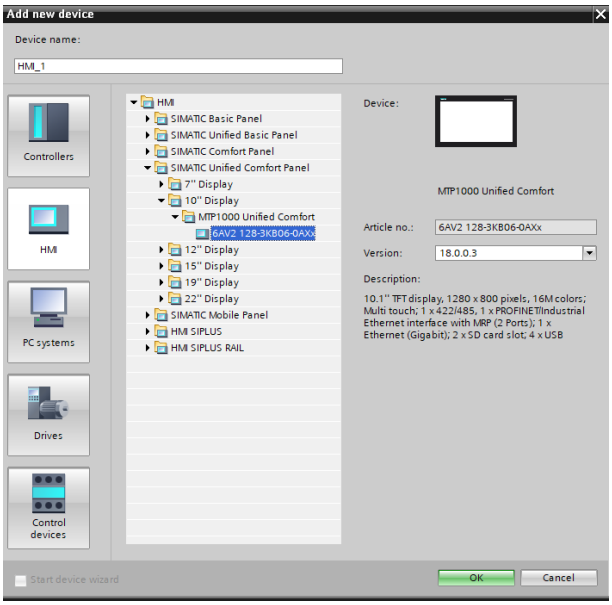
Figura 10 - Janela de configuração de comunicação *"Profinet"*.



Fonte: Autoria própria.

A adição de equipamentos à árvore do projeto pode ser realizada de duas maneiras: pela opção *"Add new device"*, localizada na aba lateral esquerda, a qual abre uma nova janela de seleção (conforme mostrado na Figura 11); ou por meio da aba lateral direita *"Hardware Catalog"*, onde os dispositivos devem ser arrastados diretamente para a área de trabalho.

Figura 11 - Janela de seleção de dispositivos a serem adicionados no projeto.

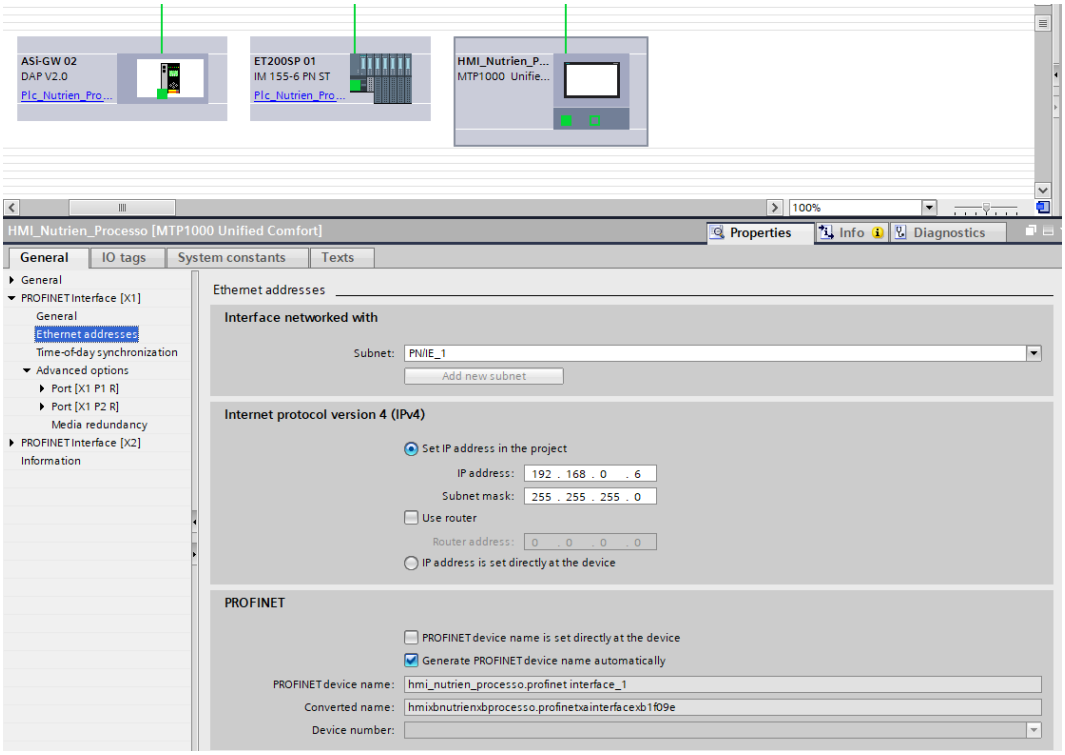


Fonte: Autoria própria.

É fundamental selecionar a versão correta do firmware do dispositivo, que deve corresponder à versão do equipamento físico. Após a inserção do dispositivo apropriado na árvore do projeto, procede-se à inserção dos seus parâmetros de configuração.

Como alguns dispositivos possuem duas portas Ethernet, é essencial configurar corretamente o endereço IP da porta conectada ao PLC. Além disso, deve-se adicionar a interface de comunicação correspondente no projeto, uma vez que essa configuração é necessária para viabilizar a troca de dados entre os equipamentos.

Figura 12 - Aba de configuração do equipamento no projeto.



Fonte: Autoria própria.

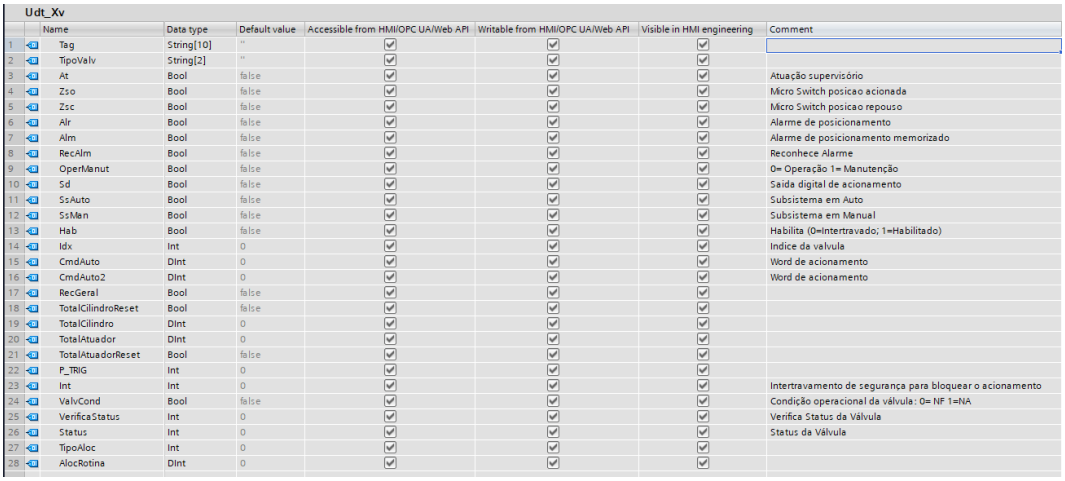
Para adicionar uma nova interface de comunicação, deve-se clicar no botão *"Add new subnet"*, localizado no parâmetro *"Interface networked with"*, dentro da aba de configuração de rede do dispositivo selecionado. A nomenclatura da interface é atribuída automaticamente pelo software; no entanto, é essencial garantir que todos os dispositivos do projeto estejam configurados para utilizar a mesma interface de rede.

3.5. Tipos de dados do PLC

O foco deste documento não é detalhar o desenvolvimento das linhas de código do PLC, uma vez que esse trabalho foi realizado por outro colaborador. Contudo, é importante destacar que houve uma padronização na nomenclatura dos equipamentos controlados pelo PLC, a qual também foi implementada na HMI.

Toda memória criada no PLC possui um tipo de dado associado, que determina seu uso no programa, como inteiro, *boolean* ou *float*. No entanto, os PLCs mais modernos permitem a criação de tipos de dados personalizados (*data types*), que podem agrupar diversas informações compostas por diferentes tipos de dados, conforme ilustrado na Figura 13.

Figura 13 - Tipo de dados utilizados para válvulas.



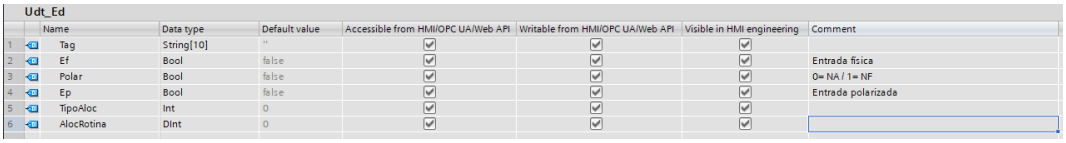
Udt_Xv

	Name	Data type	Default value	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Comment
1	Tag	String[10]	''	☑	☑	☑	
2	TipoValv	String[2]	''	☑	☑	☑	
3	At	Bool	false	☑	☑	☑	Atuação supervisório
4	Zso	Bool	false	☑	☑	☑	Micro Switch posicao acionada
5	Zsc	Bool	false	☑	☑	☑	Micro Switch posicao repouso
6	Alr	Bool	false	☑	☑	☑	Alarme de posicionamento
7	Alm	Bool	false	☑	☑	☑	Alarme de posicionamento memorizado
8	RecAlm	Bool	false	☑	☑	☑	Reconhece Alarme
9	OperManut	Bool	false	☑	☑	☑	0= Operação 1= Manutenção
10	Sd	Bool	false	☑	☑	☑	Saida digital de acionamento
11	SsAuto	Bool	false	☑	☑	☑	Subsistema em Auto
12	SsMan	Bool	false	☑	☑	☑	Subsistema em Manual
13	Hab	Bool	false	☑	☑	☑	Habilita (0=Intertravado; 1=Habilitado)
14	Idx	Int	0	☑	☑	☑	Indice da valvula
15	CmdAuto	DInt	0	☑	☑	☑	Word de acionamento
16	CmdAuto2	DInt	0	☑	☑	☑	Word de acionamento
17	RecGeral	Bool	false	☑	☑	☑	
18	TotalCilindroReset	Bool	false	☑	☑	☑	
19	TotalCilindro	DInt	0	☑	☑	☑	
20	TotalAtuador	DInt	0	☑	☑	☑	
21	TotalAtuadorReset	Bool	false	☑	☑	☑	
22	P_TRIG	Int	0	☑	☑	☑	
23	Int	Int	0	☑	☑	☑	Intertravamento de segurança para bloquear o acionamento
24	ValvCond	Bool	false	☑	☑	☑	Condição operacional da válvula: 0= NF 1=NA
25	VerificaStatus	Int	0	☑	☑	☑	Verifica Status da Válvula
26	Status	Int	0	☑	☑	☑	Status da Válvula
27	TipoAloc	Int	0	☑	☑	☑	
28	AlocRotina	DInt	0	☑	☑	☑	

Fonte: Autoria própria.

Por se tratar de um PLC *Siemens*, algumas configurações específicas devem ser habilitadas para permitir o acesso à memória para leitura e escrita por equipamentos externos. Além das válvulas, também foram utilizados sensores digitais, para os quais foi criado um tipo de dado específico.

Figura 14 - Tipo de dados utilizados para sensores digitais



Udt_Ed

	Name	Data type	Default value	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Comment
1	Tag	String[10]	''	☑	☑	☑	
2	Ef	Bool	false	☑	☑	☑	Entrada física
3	Polar	Bool	false	☑	☑	☑	0= NA / 1= NF
4	Ep	Bool	false	☑	☑	☑	Entrada polarizada
5	TipoAloc	Int	0	☑	☑	☑	
6	AlocRotina	DInt	0	☑	☑	☑	

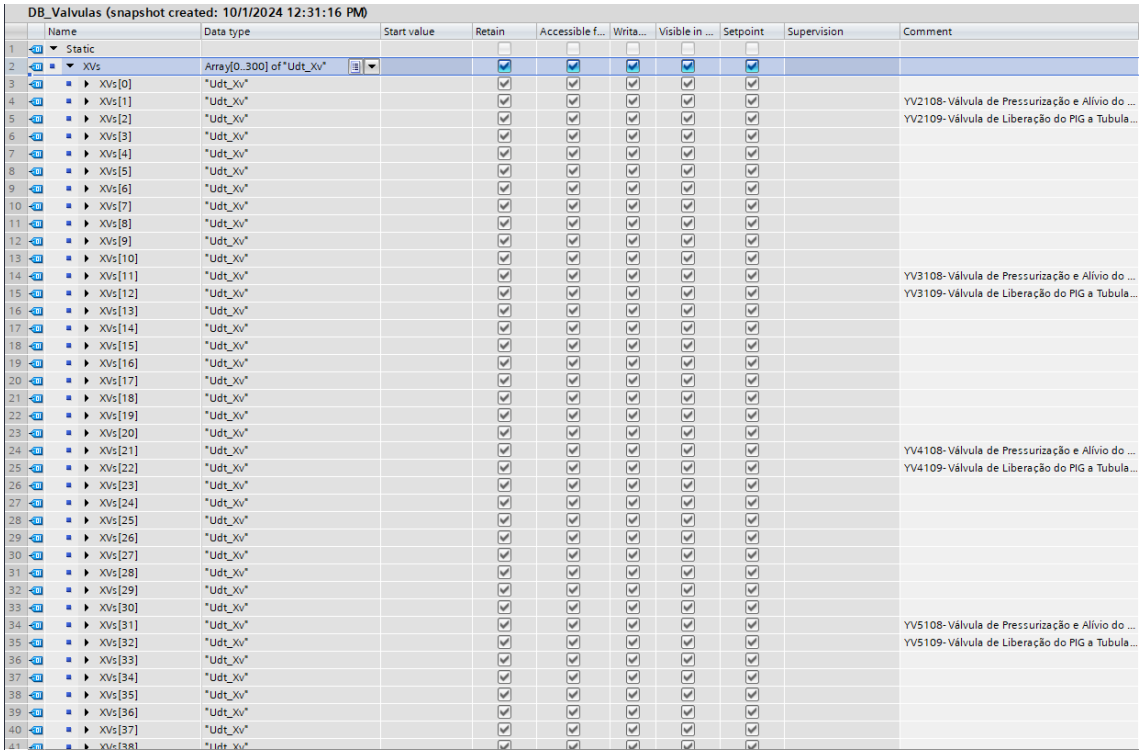
Fonte: Autoria própria.

A estrutura destes tipos de dados foi construída prevendo seu uso para todo e qualquer tipo de válvula ou sensor digital, então é possível que não seja utilizado todos os espaços de memória do tipo de dado.

Após a criação dos tipos de dados, é necessário definir as memórias que serão utilizadas no código do PLC, associando-as a esses novos tipos, como exemplificado na Figura 15. Dentro do bloco de dados (DB) criado, é fundamental habilitar a configuração "*Retain*", pois no caso de falha de energia que cause o reinício do PLC, todas as informações de parametrização de

memória seriam perdidas — restauradas para uma memória limpa — caso essa opção não esteja ativada.

Figura 15 - DB das válvulas.



DB_Valvulas (snapshot created: 10/1/2024 12:31:16 PM)

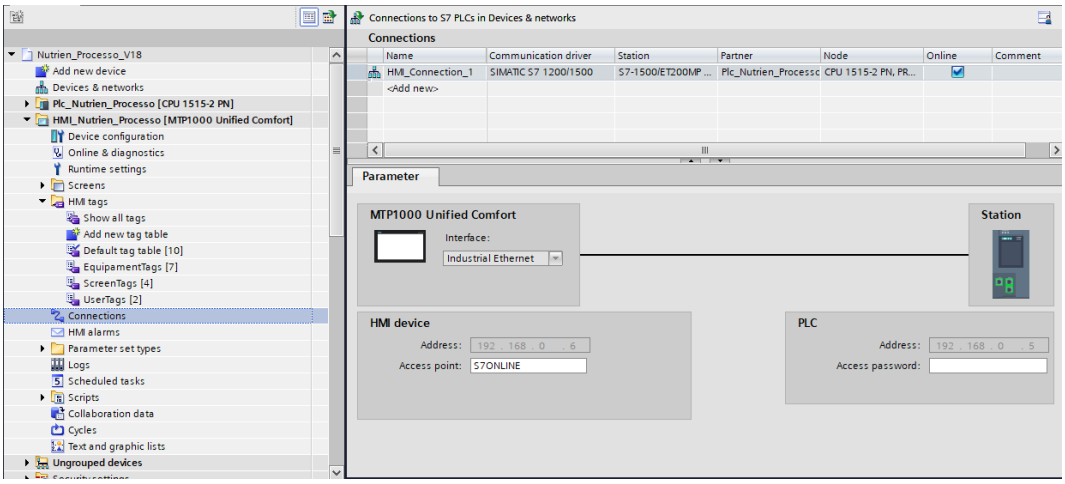
	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
1	Static									
2	XVs	Array[0..300] of "Udt_Xv"								
3	XVs[0]	"Udt_Xv"								
4	XVs[1]	"Udt_Xv"								YV2108- Válvula de Pressurização e Alívio do ...
5	XVs[2]	"Udt_Xv"								YV2109- Válvula de Liberação do PIG a Tubula...
6	XVs[3]	"Udt_Xv"								
7	XVs[4]	"Udt_Xv"								
8	XVs[5]	"Udt_Xv"								
9	XVs[6]	"Udt_Xv"								
10	XVs[7]	"Udt_Xv"								
11	XVs[8]	"Udt_Xv"								
12	XVs[9]	"Udt_Xv"								
13	XVs[10]	"Udt_Xv"								
14	XVs[11]	"Udt_Xv"								YV3108- Válvula de Pressurização e Alívio do ...
15	XVs[12]	"Udt_Xv"								YV3109- Válvula de Liberação do PIG a Tubula...
16	XVs[13]	"Udt_Xv"								
17	XVs[14]	"Udt_Xv"								
18	XVs[15]	"Udt_Xv"								
19	XVs[16]	"Udt_Xv"								
20	XVs[17]	"Udt_Xv"								
21	XVs[18]	"Udt_Xv"								
22	XVs[19]	"Udt_Xv"								
23	XVs[20]	"Udt_Xv"								
24	XVs[21]	"Udt_Xv"								YV4108- Válvula de Pressurização e Alívio do ...
25	XVs[22]	"Udt_Xv"								YV4109- Válvula de Liberação do PIG a Tubula...
26	XVs[23]	"Udt_Xv"								
27	XVs[24]	"Udt_Xv"								
28	XVs[25]	"Udt_Xv"								
29	XVs[26]	"Udt_Xv"								
30	XVs[27]	"Udt_Xv"								
31	XVs[28]	"Udt_Xv"								
32	XVs[29]	"Udt_Xv"								
33	XVs[30]	"Udt_Xv"								
34	XVs[31]	"Udt_Xv"								YV5108- Válvula de Pressurização e Alívio do ...
35	XVs[32]	"Udt_Xv"								YV5109- Válvula de Liberação do PIG a Tubula...
36	XVs[33]	"Udt_Xv"								
37	XVs[34]	"Udt_Xv"								
38	XVs[35]	"Udt_Xv"								
39	XVs[36]	"Udt_Xv"								
40	XVs[37]	"Udt_Xv"								
41	XVs[38]	"Udt_Xv"								

Fonte: Autoria própria.

3.6. Adicionando comunicação da HMI com o PLC

Embora o link de comunicação *Profinet* já tenha sido inserido na árvore de projeto, essa configuração inicial permite apenas o reconhecimento mútuo dos dispositivos na mesma rede de dados. Para viabilizar a troca efetiva de informações entre a HMI e o PLC, é imprescindível configurar explicitamente a conexão entre ambos. É essa configuração que garante à HMI permissão total de leitura e escrita nas áreas de memória do PLC.

Figura 16 - Configuração de conexão da HMI com o PLC.

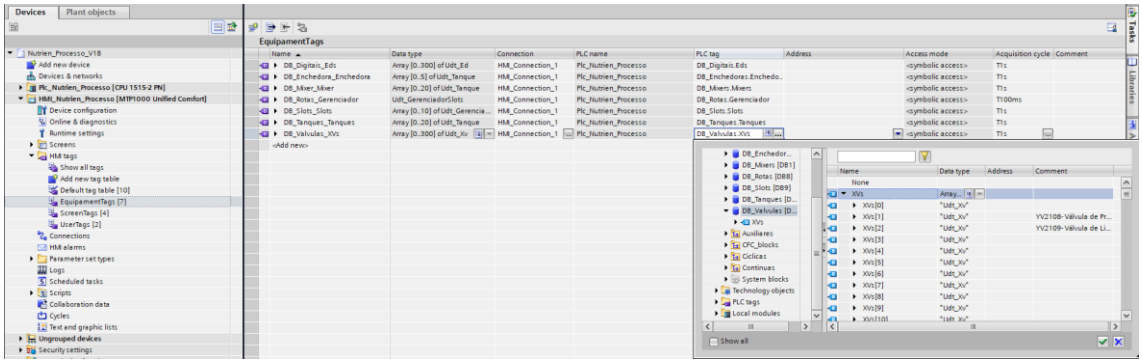


Fonte: Autoria própria.

Conforme ilustrado na Figura 16, a configuração da conexão é realizada por meio da aba *"Connections"*, localizada na pasta da HMI dentro do projeto. Como padrão, essa aba não apresenta conexões previamente definidas. Portanto, é necessário acionar o botão *"<Add new>"* para criar uma nova conexão, selecionando os dispositivos envolvidos — HMI e PLC — e configurando os respectivos endereços IP, que devem corresponder à interface física de rede à qual estarão conectados via cabo.

Uma vez criado o *link* de conexão, pode-se adicionar as memórias do PLC que serão lidas e/ou escritas na HMI. Como mostra a Figura 17, dentro das pastas da HMI, há uma relacionada a memória local da HMI para animação ou memória vinculada ao PLC adicionada: *"HMI tags"*.

Figura 17 - Janela de Tags utilizadas na HMI.



Fonte: Autoria própria.

Dentro dessa pasta, é possível criar subgrupos para organizar as memórias utilizadas na HMI, como *"EquipmentTags"* ou *"ScreenTags"*. Dentro desses subgrupos, podem ser adicionadas memórias vinculadas ao PLC. O nome pode ser igual ou diferente ao da variável no PLC; porém, para adicionar as DBs, é necessário preencher corretamente as informações na coluna *"PLC tag"*.

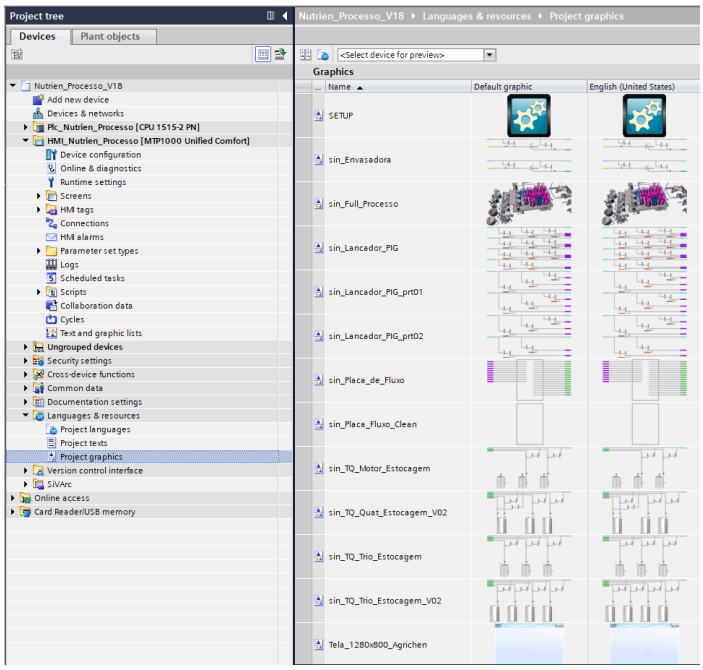
Vale a pena observar que uma vez inserido as informações corretamente, as outras colunas se preenchem automaticamente, e na coluna de *"Connection"* sempre irá aparecer o nome da configuração de conexão da HMI com o PLC; ressaltando a importância de sua criação antes de tentar vincular as memórias do PLC.

3.7. Configuração das telas de navegação do projeto da HMI

3.7.1. Adição de propriedades das telas principais

Inicialmente, todas as imagens utilizadas nas telas e animações devem ser adicionadas ao projeto, mais especificamente, como mostra a Figura 18, na aba *"Project graphics"*, localizada na pasta de configuração *"Languages & resources"*. Ressalta-se que o formato mais recomendado para as imagens é o ".png", devido à sua facilidade em exibir a imagem centralizada e com fundo transparente.

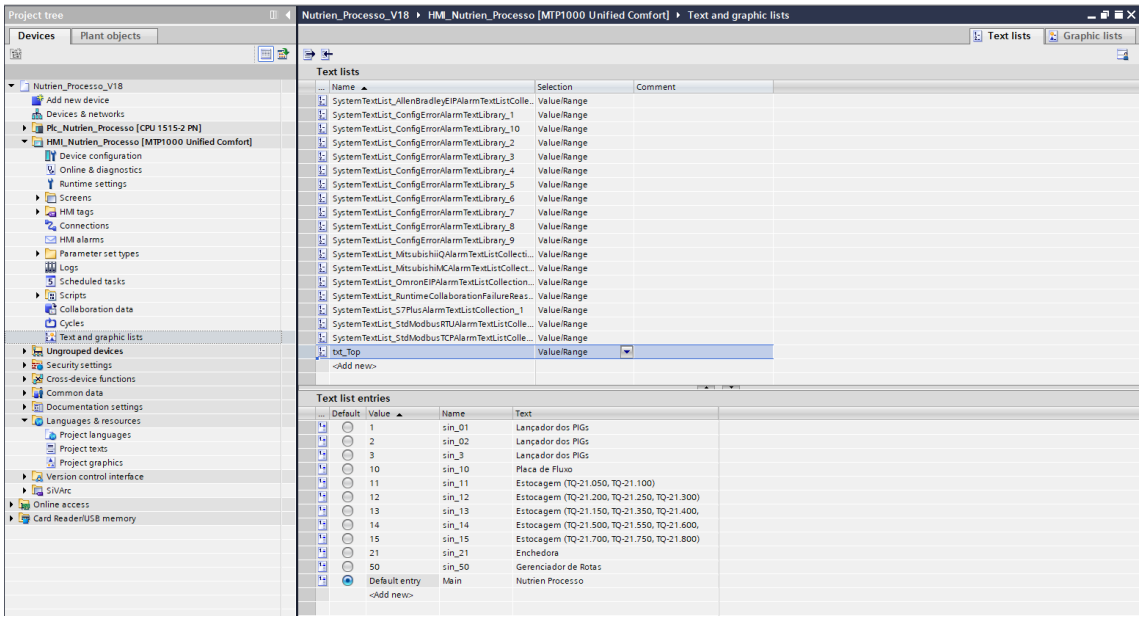
Figura 18 - Janela de adição das imagens do projeto.



Fonte: Autoria própria.

Para a sincronização das telas desenvolvidas, foram criadas listas de textos e imagens a serem utilizadas no projeto. A Figura 19 apresenta a lista de textos definida para os títulos das telas, ou seja, à medida que cada tela é acionada, o texto correspondente é automaticamente exibido como título.

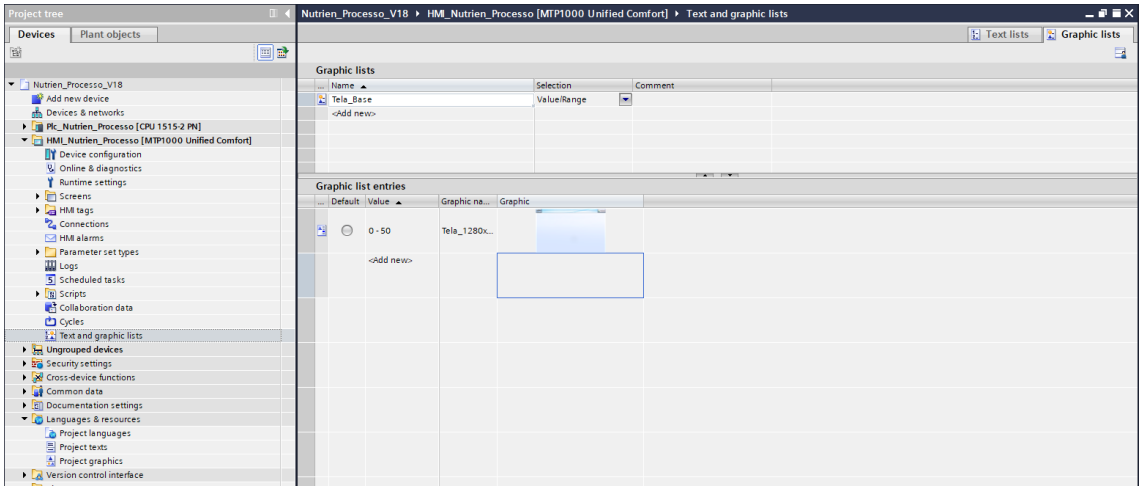
Figura 19 - Janela da lista de texto do projeto.



Fonte: Autoria própria.

A Figura 20 apresenta a lista de imagens utilizada como fundo padrão para todas as telas do projeto. Seguindo a mesma lógica da lista anterior, à medida que cada tela é chamada, a imagem de fundo correspondente é exibida automaticamente.

Figura 20 - Janela da lista de imagens do projeto.

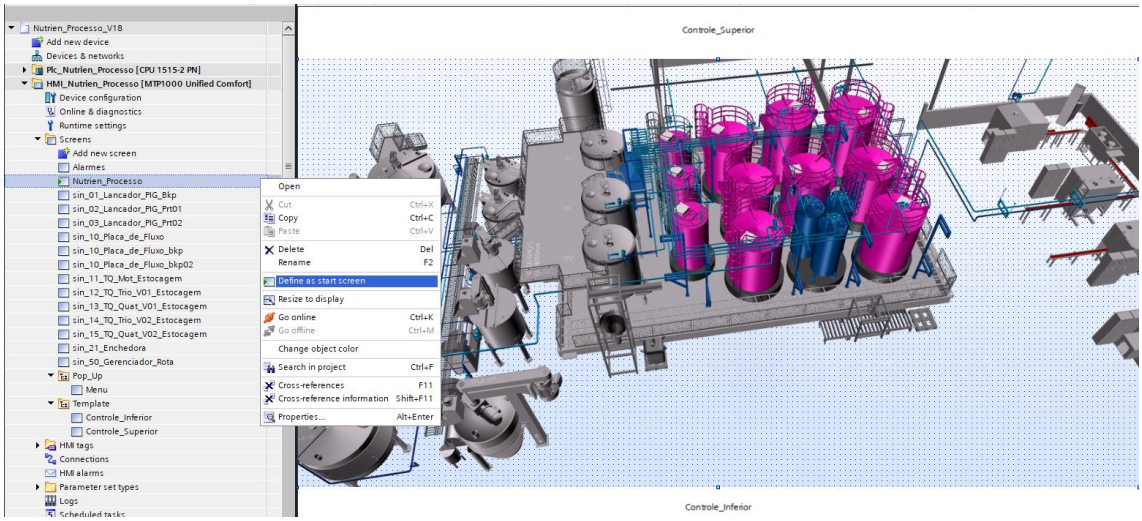


Fonte: Autoria própria.

3.7.2. Configurando os *"Templates"* principais

As telas de visualização do projeto são criadas na pasta *"Screens"* da HMI, onde é possível configurar uma janela específica como tela inicial do projeto, só precisa clicar com o botão direito sobre a tela desejada, conforme ilustrado na Figura 21.

Figura 21 - Configuração da tela inicial do projeto da HMI.

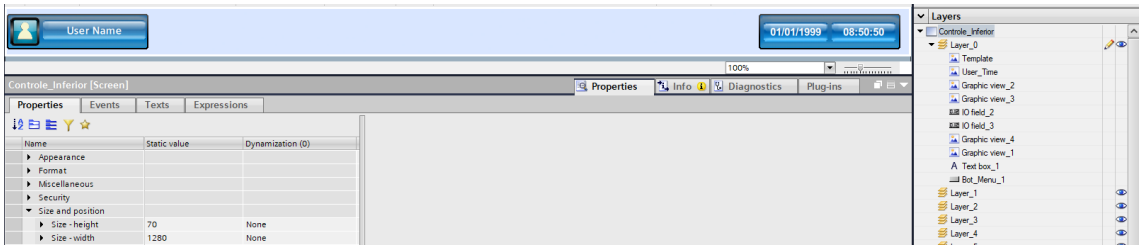


Fonte: Autoria própria.

Dentro da pasta "Screens", podem ser criadas subpastas para organizar as telas, como exemplificado na Figura 21, com os subgrupos *"Pop_Up"* e *"Template"*. Todas as telas principais do projeto utilizam como padrão a tela de fundo e algumas configurações de animação. Além disso, todas as telas principais do projeto possuem uma área padrão de visualização superior e inferior.

Para construir o padrão de tela inferior, foi adicionado uma tela normal da HMI, porém com configurações de tamanho reduzido, para que esse tamanho seja sobreposto como rodapé de todas as telas do projeto. Como mostra a Figura 22.

Figura 22 - Tela de controle inferior do projeto.

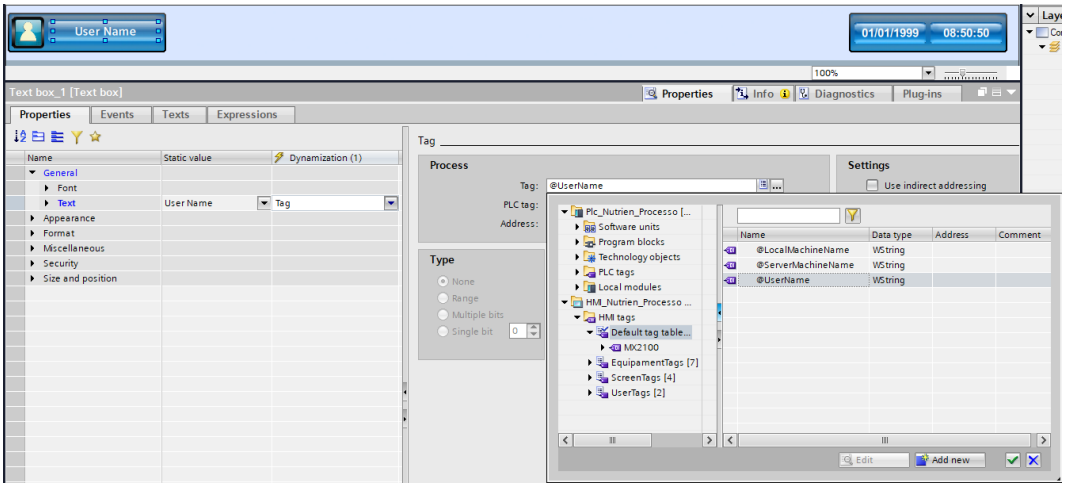


Fonte: Autoria própria.

O texto do nome do usuário foi relacionado a *tag "@UserName"*, que é uma memória local da HMI, no qual apresenta o usuário atual que está acessando o projeto, por exemplo, operador ou manutenção. Como esta apresentada na figura 23.

O texto referente ao nome do usuário está vinculado à *tag "@UserName"*, que corresponde a uma memória local da HMI e exibe o usuário atualmente acessando o projeto, como, por exemplo, operador ou manutenção, conforme apresentado na Figura 23.

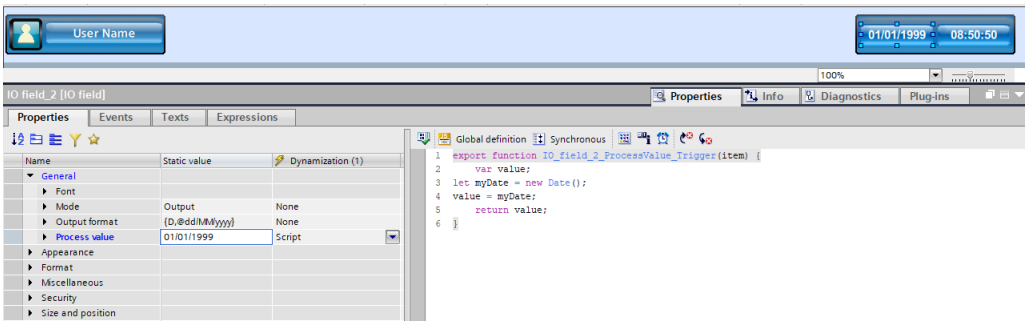
Figura 23 - Configuração de nome de usuário da HMI.



Fonte: Autoria própria.

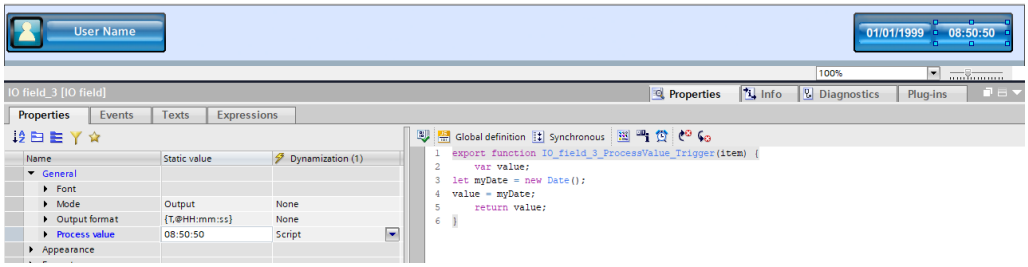
Para apresentar a data e o horário da HMI, foi utilizado uma função do código de *JavaScript*, como mostra as Figuras 24 e 25. E para apresentar o resultado, foi utilizado o bloco *"IO field"*, que tem a função de inserir ou apresentar valores em formatos configuráveis. Para estes casos, foi utilizado o formato "{D,@dd/MM/yyyy}" para data e o formato "{T,@HH:mm:ss}" para o horário.

Figura 24 - Configuração da data da HMI.



Fonte: Autoria própria.

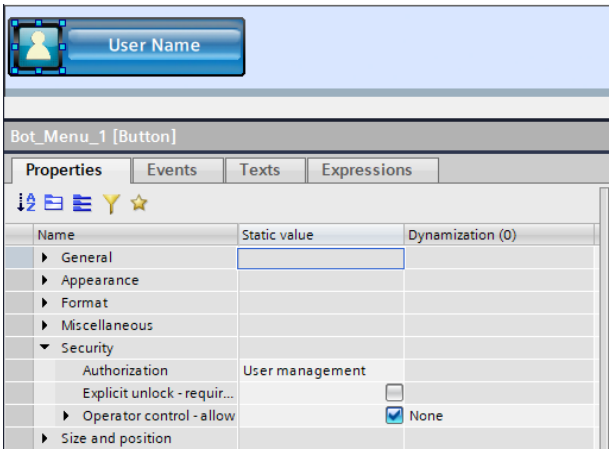
Figura 25 - Configuração do horário da HMI.



Fonte: Autoria própria.

Foi criado um botão invisível ao lado do texto do nome do usuário para a função de desligar (ou sair) do projeto. Para evitar acionamentos indevidos por qualquer usuário, o acesso a esse botão foi restrito apenas a usuários com nível de administrador, como, por exemplo, manutenção. Conforme mostrado na Figura 26, essa configuração foi realizada na *opção "User management"*, dentro de *"Authorization"*.

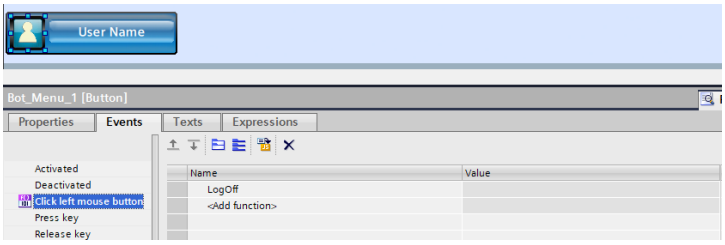
Figura 26 - Configuração de nível de usuário para o botão de sair.



Fonte: Autoria própria.

Na aba de eventos do botão, para adicionar uma animação, foi inserido a função de *"LogOff"*, como mostra a Figura 27.

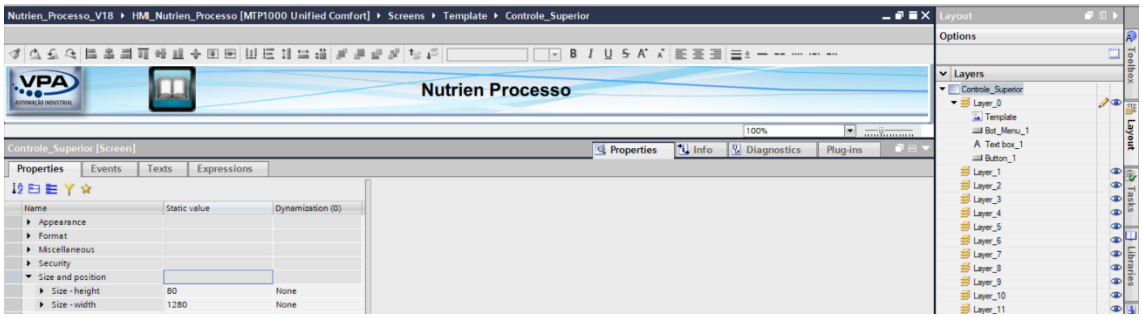
Figura 27 - Configuração da animação para o botão de sair.



Fonte: Autoria própria.

Para construir o padrão de tela superior, foi adicionada uma tela normal da HMI, porém da mesma forma da tela inferior, foi reduzido o seu tamanho para sobrepor apenas o cabeçalho das telas do projeto como mostrado na Figura 28.

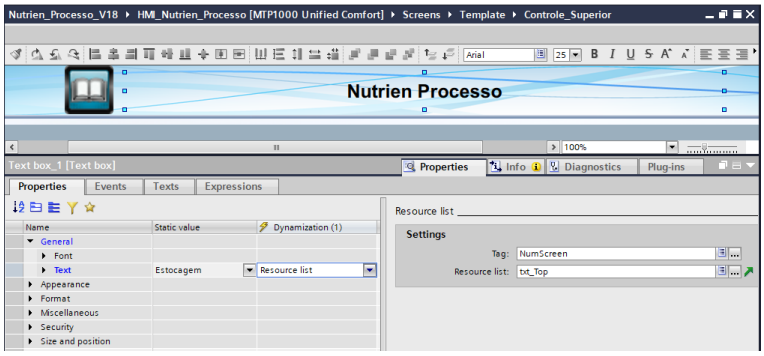
Figura 28 - Tela de controle superior do projeto.



Fonte: Autoria própria.

Para inserir o recurso da lista de textos no título da tela, a lista foi selecionada nos parâmetros do texto e vinculada à memória local da HMI responsável por identificar a tela aberta, conforme apresentado na Figura 29. A memória local da HMI "NumScreen" será explicada posteriormente.

Figura 29 - Configuração do recurso da lista de texto.

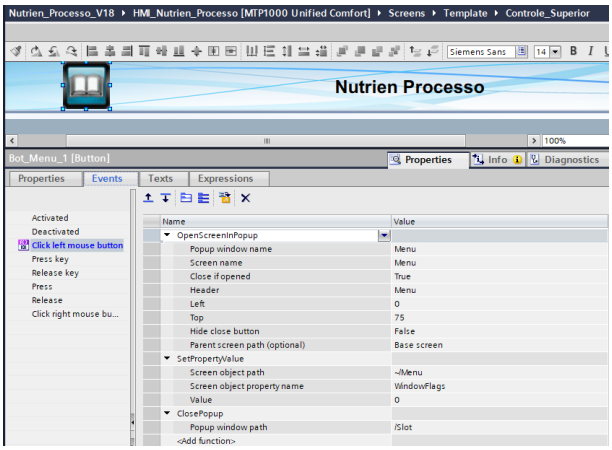


Fonte: Autoria própria.

3.7.3. Configuração do menu de telas do projeto

Além disso, no controle da tela superior há um botão com a função de abrir o menu de botões para a navegação entre as demais telas do projeto. A tela de menu será apresentada adiante, porém sua ativação está configurada conforme ilustrado na Figura 30, utilizando o recurso *"OpenScreenInPopup"* da HMI.

Figura 30 - Configuração da chamada da tela de menu do projeto.

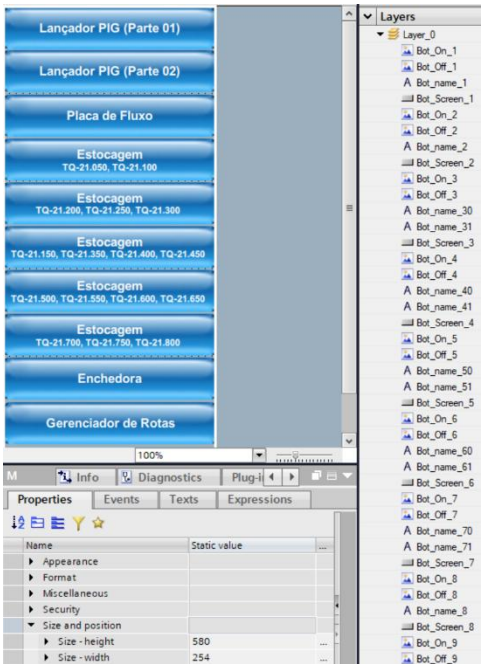


Fonte: Autoria própria.

Para a navegação entre as telas do projeto, foi desenvolvida uma tela de menu, adicionada na aba *"Pop_Up"* do projeto. Sua visualização na HMI ocorre sempre como uma tela reduzida sobreposta à tela atual, funcionando como uma animação de sobreposição.

Seu tamanho foi calculado para apresentar apenas os botões de chamadas das telas, como se mostra na Figura 31. Cada botão tem um conjunto de objetos de animação: imagem de botão ligado, imagem de botão desligado, texto contendo o nome da tela e um botão que irá realizar a chamada para a nova tela de operação.

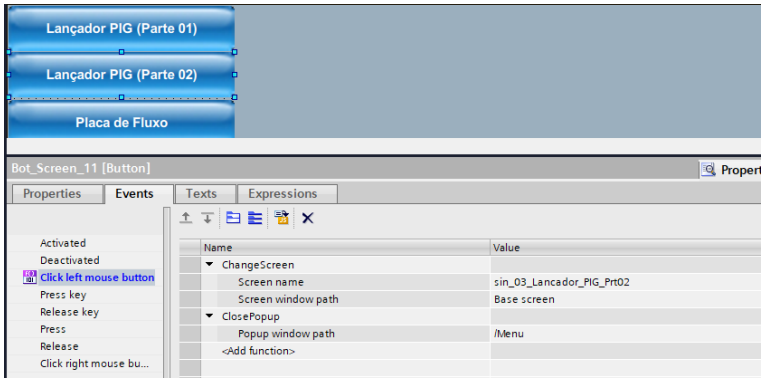
Figura 31 - Tela de Menu.



Fonte: Autoria própria.

O texto de cada botão não possui animação, apenas exibe as informações do título da tela associada ao botão. O botão — configurado para ser invisível — possui a animação de abertura da tela do projeto e de fechamento da tela de menu, garantindo uma visualização limpa da tela aberta, conforme apresentado na Figura 32.

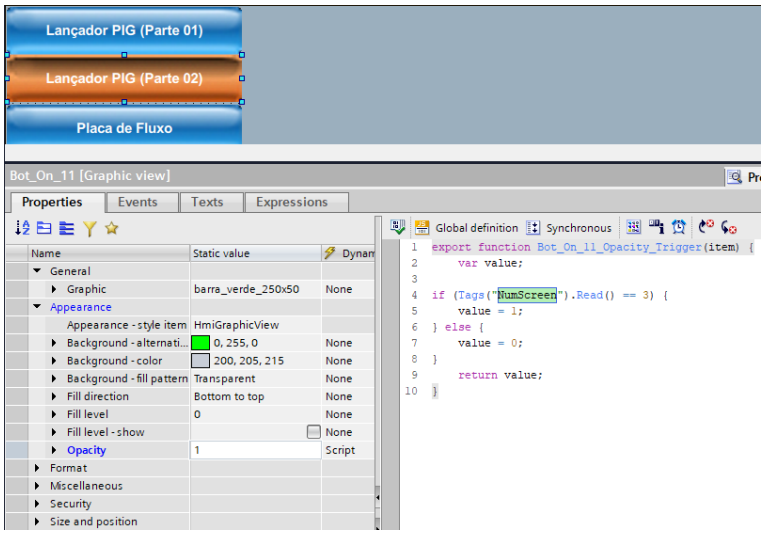
Figura 32 - Configuração da animação dos botões da tela de menu.



Fonte: Autoria própria.

Para a animação do botão ligado e desligado, foram adicionadas imagens correspondentes às duas condições na mesma posição, conforme ilustrado na Figura 33.

Figura 33 - Configuração da animação das imagens dos botões.



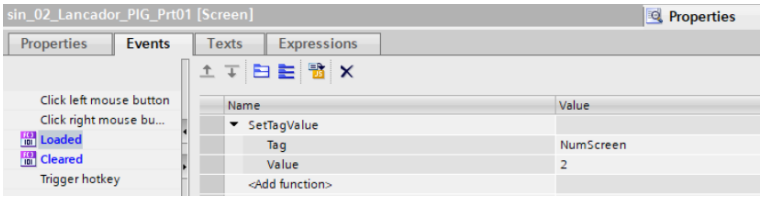
Fonte: Autoria própria.

Através de uma memória local "*NumScreen*", configurada para gerar um número dependo de qual tela estiver aberta, programará qual imagem do botão será apresentada na tela de menu. Esse código foi implementado em todas as imagens de botão, com a diferença que, quando o número da tela for igual ao programado no botão, este apresentará a imagem de cor laranja, indicando que a tela atual foi chamada por este botão.

3.7.4. Configurando transição das telas principais do projeto

A memória local *"NumScreen"* foi criada para numerar todas as telas de comando. Ela foi implementada na animação de abertura das telas principais do projeto, conforme mostrado na Figura 34.

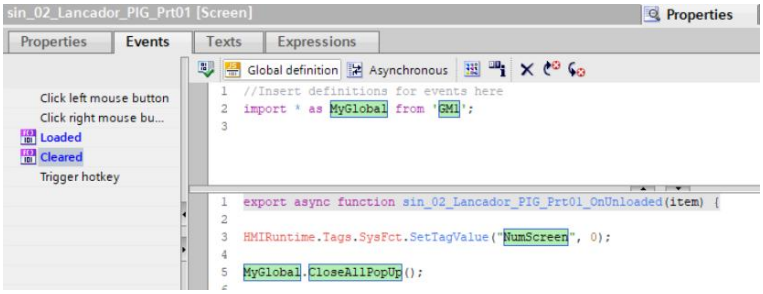
Figura 34 - Configuração da memória local "NumScreen".



Fonte: Autoria própria.

A animação de fechamento da tela de comando, apresentada na Figura 34, foi implementada devido a um problema identificado durante os diversos testes do projeto. O problema consistia no estouro da memória *buffer* causado pelas chamadas consecutivas de telas sobrepostas. Para solucionar essa questão, foi programada uma limpeza forçada sempre que uma nova tela de comando fosse chamada, conforme demonstrado na Figura 35.

Figura 35 - Configuração da animação de limpeza da tela de comando.

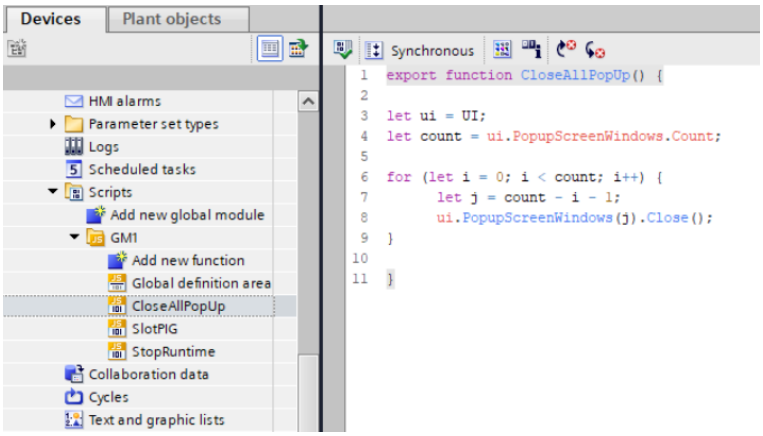


Fonte: Autoria própria.

Em vez de utilizar funções prontas do programa, optou-se pelo desenvolvimento de códigos por meio de *scripts*. Para iniciar o uso desses códigos, é necessário importar a pasta de scripts — neste projeto criada como *"GM1"* — dentro da janela de eventos globais da tela de comando, denominada *"Global definition"*.

Em seguida, para utilizar o código específico dentro da pasta de *scripts*, escreve-se o nome do script no seguinte formato: *"MyGlobal.CloseAllPopUp();"*, conforme mostrado na Figura 35. A pasta dos *scripts* e o código utilizado na tela de comando são apresentados na Figura 36.

Figura 36 - Script para fechar todas as telas e sub telas de comando.

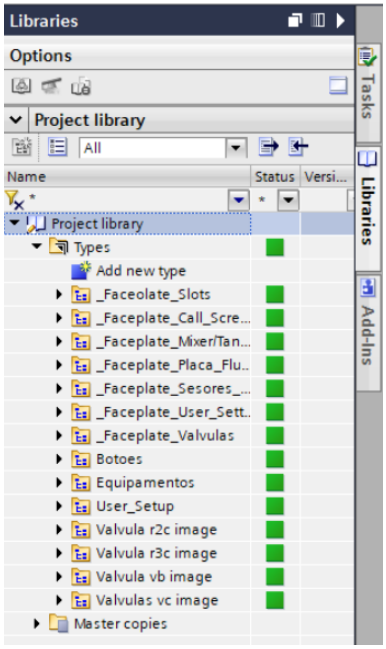


Fonte: Autoria própria.

3.8. Configuração de objetos de animação - ‘Faceplates”

Faceplates são sub-telas ou objetos de animação dentro das telas de comando. Nas abas laterais direitas, há a opção da biblioteca de objetos desenvolvidos — *Libraries* — conforme apresentado na Figura 37.

Figura 37 - Biblioteca de objetos desenvolvidos do projeto.



Fonte: Autoria própria.

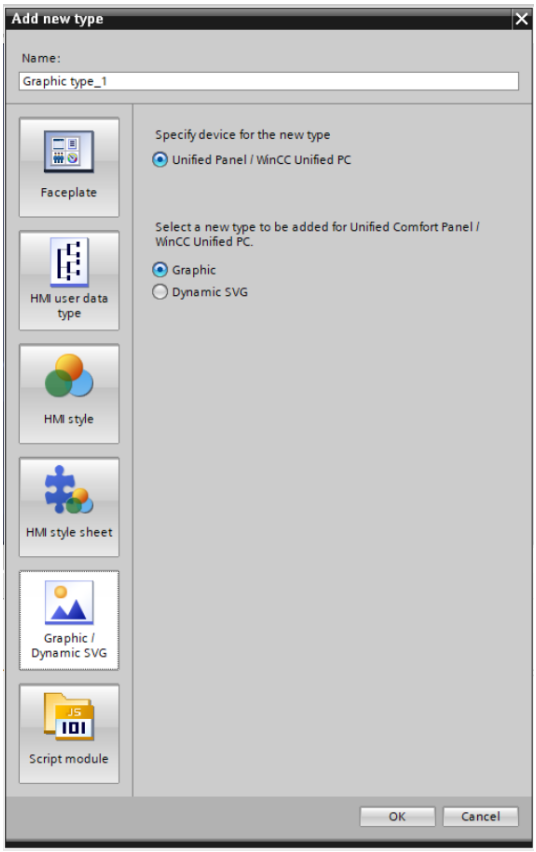
Foram desenvolvidos muitos objetos de animação no projeto, a tal ponto que foi necessário separá-los em pastas.

3.8.1. Adição de objetos na biblioteca

Infelizmente, mesmo adicionando imagens para animação dos objetos na pasta *"Project graphics"*, a opção de imagens não permite utilizar essa pasta como referência para as animações. Por isso, foi necessário adicioná-las diretamente nas bibliotecas de trabalho, podendo ser encapsuladas em subpastas para melhor organização.

Para adicionar as imagens, é necessário pressionar a opção *"Add new type"* para abrir a janela de configuração mostrada na Figura 38 e selecionar a opção de *"Graphic / Dynamic SVG"*.

Figura 38 - Janela de adição de objeto de imagem.

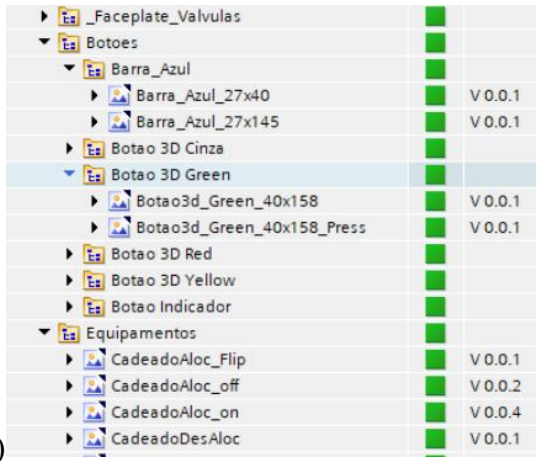


Fonte: Autoria própria.

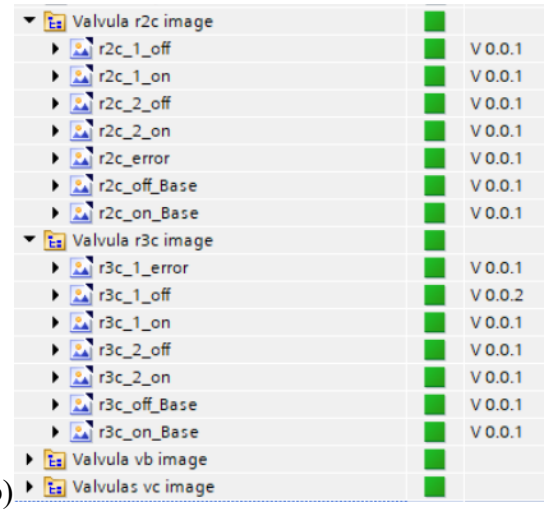
Neste projeto, foram utilizadas muitas imagens de animação, como válvulas abertas e fechadas, objetos travados ou destravados, e botões em seus estados normal e pressionados de diferentes cores.

Figura 39 - Objetos de imagem da biblioteca utilizados.

a)



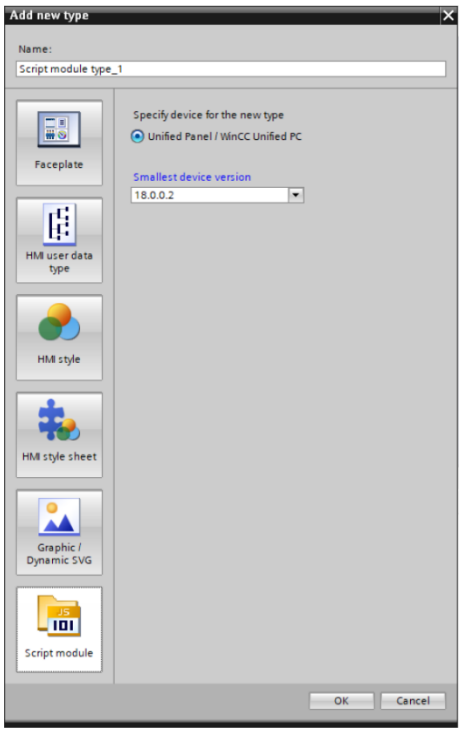
b)



Fonte: Autoria própria.

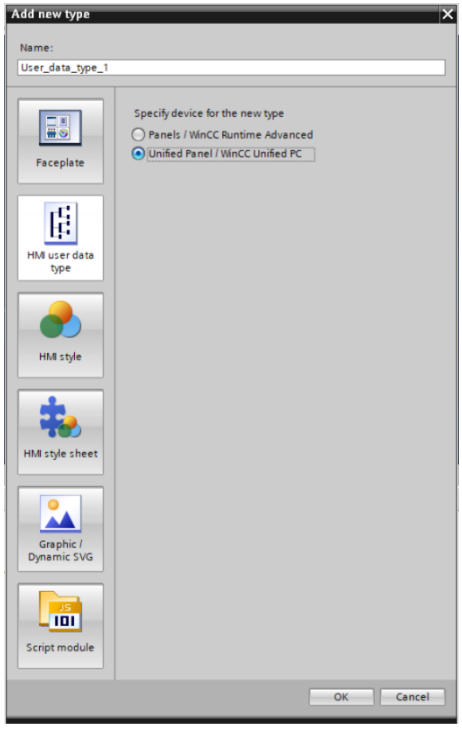
Além das imagens, foram utilizadas as opções de *"Script module"* para uso de código desenvolvidos dentro dos objetos; *"*HMI *user data type"* para a utilização de tipos de dados sincronizado com o PLC, como dados de válvulas ou sensores digitais; e *"Faceplate"* para a criação dos próprios objetos.

Figura 40 - Janela de adição de Script.



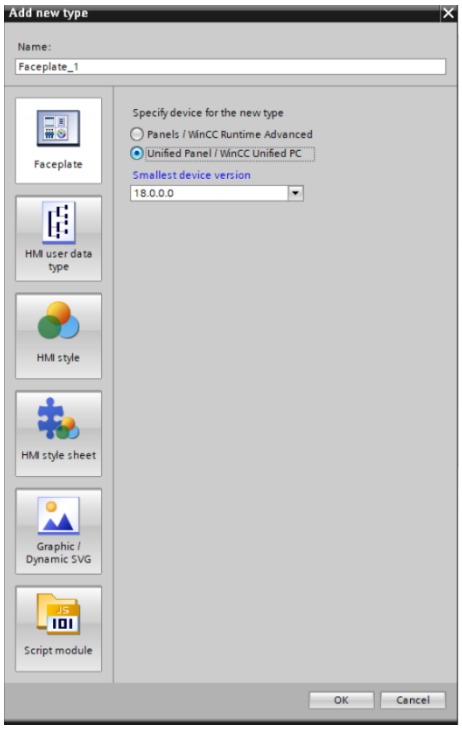
Fonte: Autoria própria.

Figura 41 - Janela de adição de tipo de dado.



Fonte: Autoria própria.

Figura 42 - Janela de adição de sub telas ou objetos de animação.

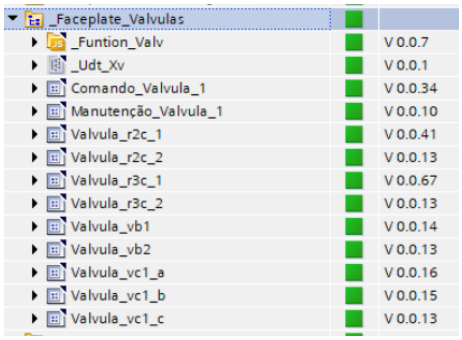


Fonte: Autoria própria.

3.8.2. Desenvolvimento dos objetos de animação das válvulas

Pondo estas opções, neste projeto foram utilizados todos estes objetos da biblioteca para o desenvolvimento dos objetos de animação de válvulas e suas sub telas de controle. Como foram criadas muitas válvulas, seus objetos foram criados de forma parametrizada utilizando o vetor do dado de válvula – anteriormente introduzido. Como apresentada na Figura 43.

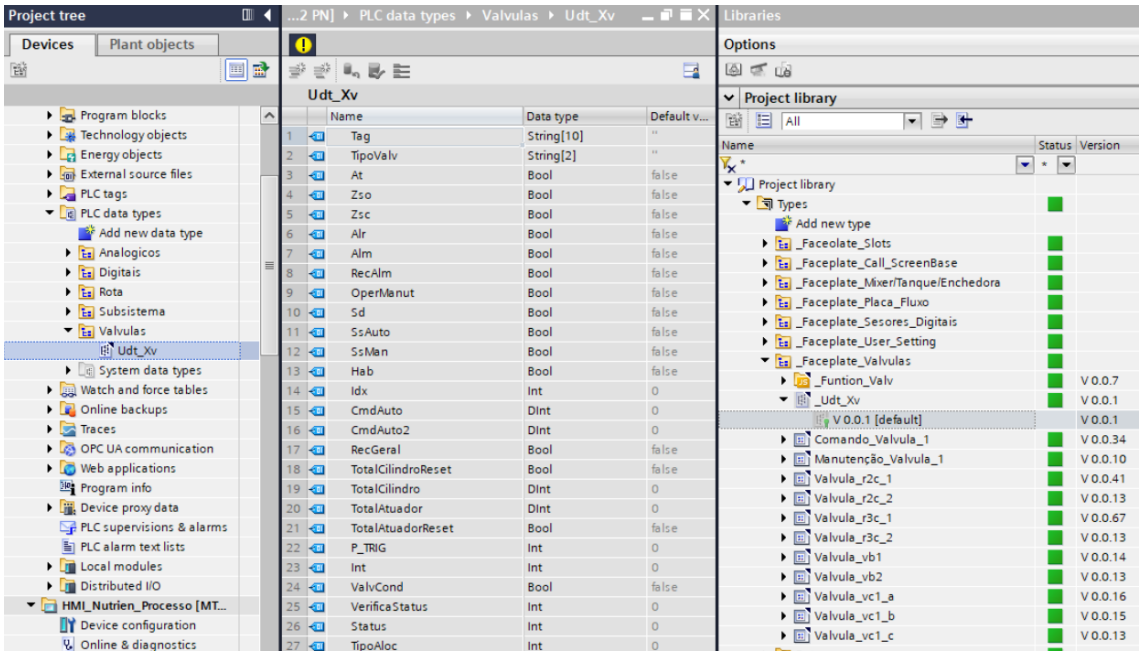
Figura 43 - Pasta da biblioteca para as válvulas.



Fonte: Autoria própria.

Para adicionar o tipo de dado das válvulas, em vez de criar cada dado por si, é mais rápido arrastar o tipo de dado da válvula da pasta do PLC para dentro da biblioteca da HMI, desta forma se consegue sincronizar os dados; como mostra a Figura 44.

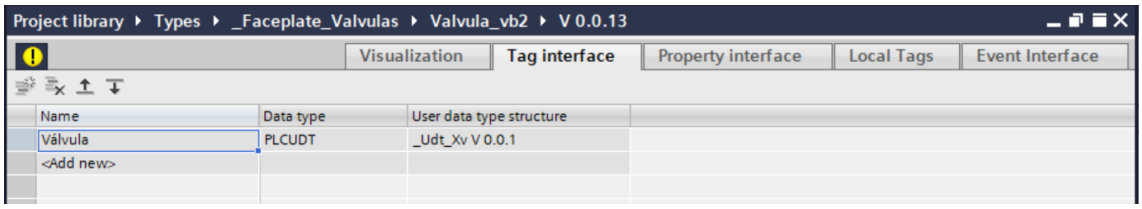
Figura 44 - Tipo de dados utilizados nas animações das válvulas.



Fonte: Autoria própria.

Antes de criar qualquer tipo de animação das válvulas, é necessário vincular o tipo de dados apresentado dentro do objeto das válvulas. Desta forma, pode-se desenvolver as animações utilizando as memórias disponíveis. Como se estivesse criando uma válvula genérica, adiciona-se na aba de *"Tag Interface"* o tipo de dado da válvula, conforme mostrado na Figura 45.

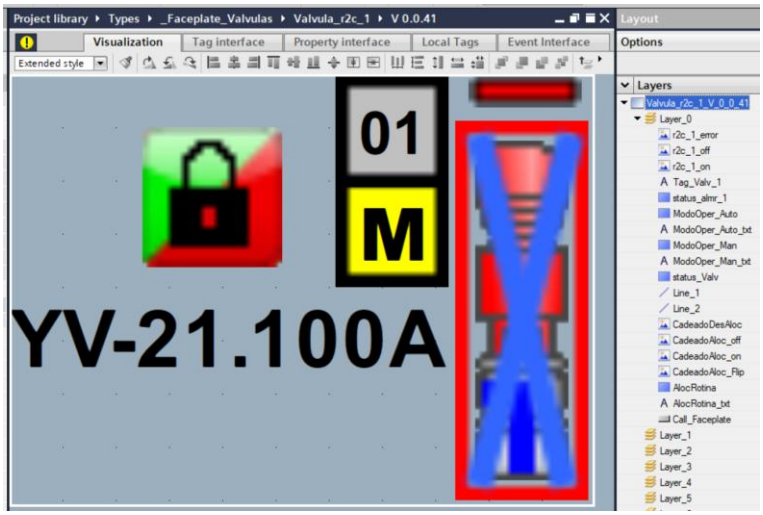
Figura 45 - Adição de memória de válvula geral.



Fonte: Autoria própria.

Todas as válvulas utilizadas neste projeto possuem as mesmas animações e características de função, tendo apenas a única diferença do desenho que as representam. Como demonstrada na Figura 46. Foram desenvolvidos objetos de válvulas para os tipos; borboleta, *Mux* de 3 vias, *Mux* de 2 vias, pistões duplos e suas variações de horizontal, vertical, do lado direito ou esquerdo.

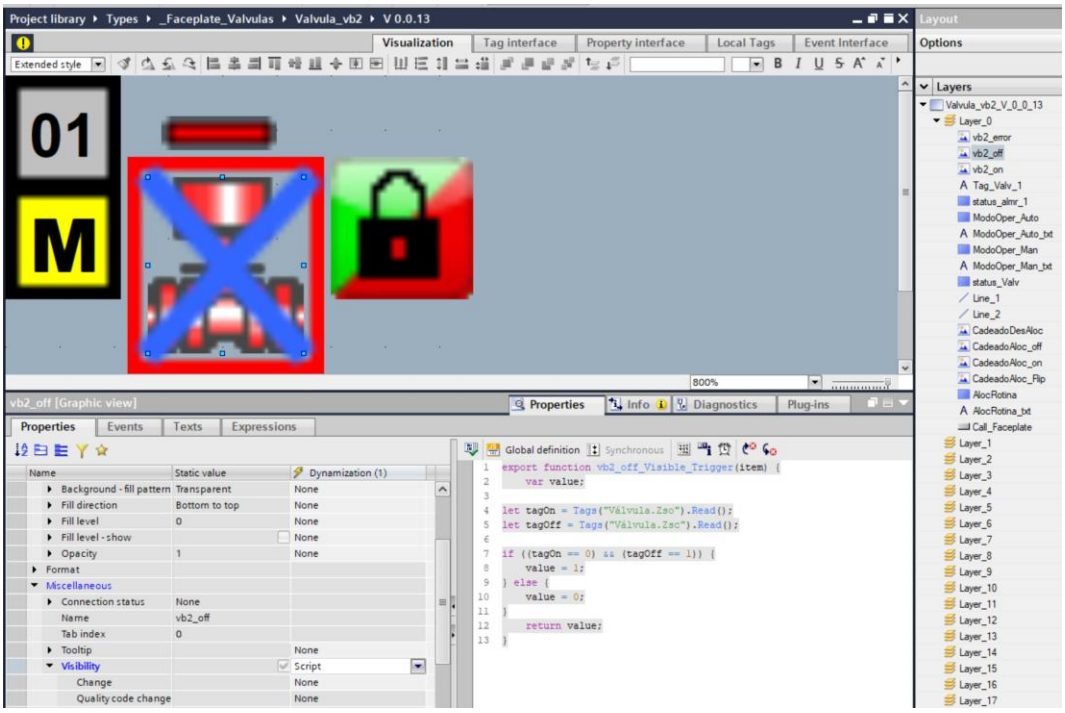
Figura 46 - Objetos utilizados na animação das válvulas.



Fonte: Autoria própria.

A primeira animação apresentada será de representação da válvula aberta, fechada ou em falha (quando não se sabe se está aberta ou fechada); apresentando colorações básicas de vermelho para aberto, verde claro para fechado e cinza para falha. Sua animação de visualização se dá pelo código, na opção de *Script,* demonstrado na Figura 47; em que se utiliza as memórias de aberto e fechado da válvula; "Válvula.*Zso*" e "Válvula.*Zsc*", respectivamente.

Figura 47 - Código de animação dos estados de aberto/fechado das válvulas.

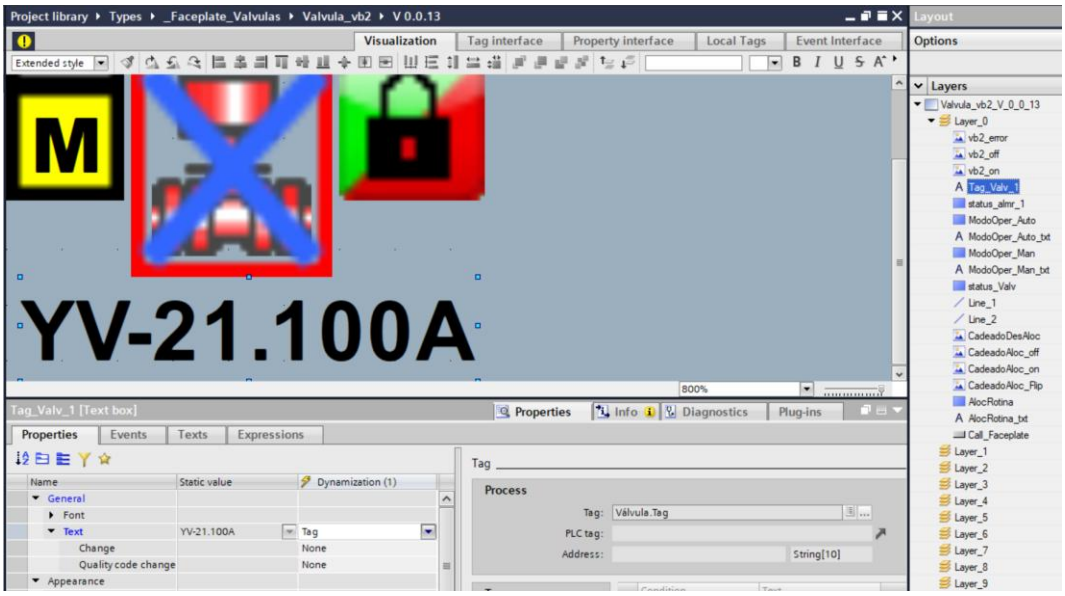


Fonte: Autoria própria.

Cada imagem de estado da válvula, leva um código similar ao apresentado na Figura 47, porém com a diferença de sua ativação tendendo as memórias de aberto e fechado para cada imagem.

O nome de cada válvula também recebe um vínculo de animação que requer uma memória da válvula geral, “Válvula.*Tag*”. Como apresentada na Figura 48.

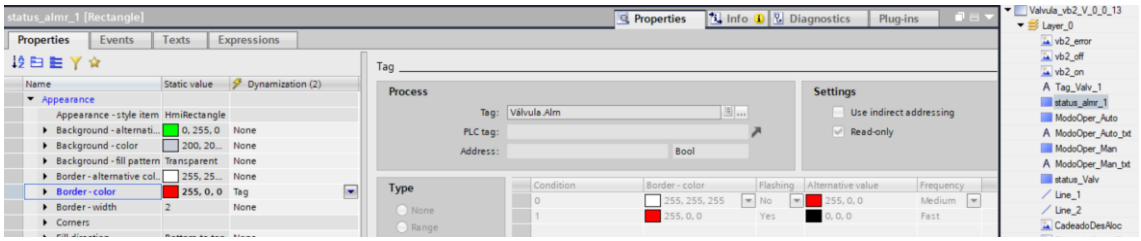
Figura 48 - Configuração do nome das válvulas.



Fonte: Autoria própria.

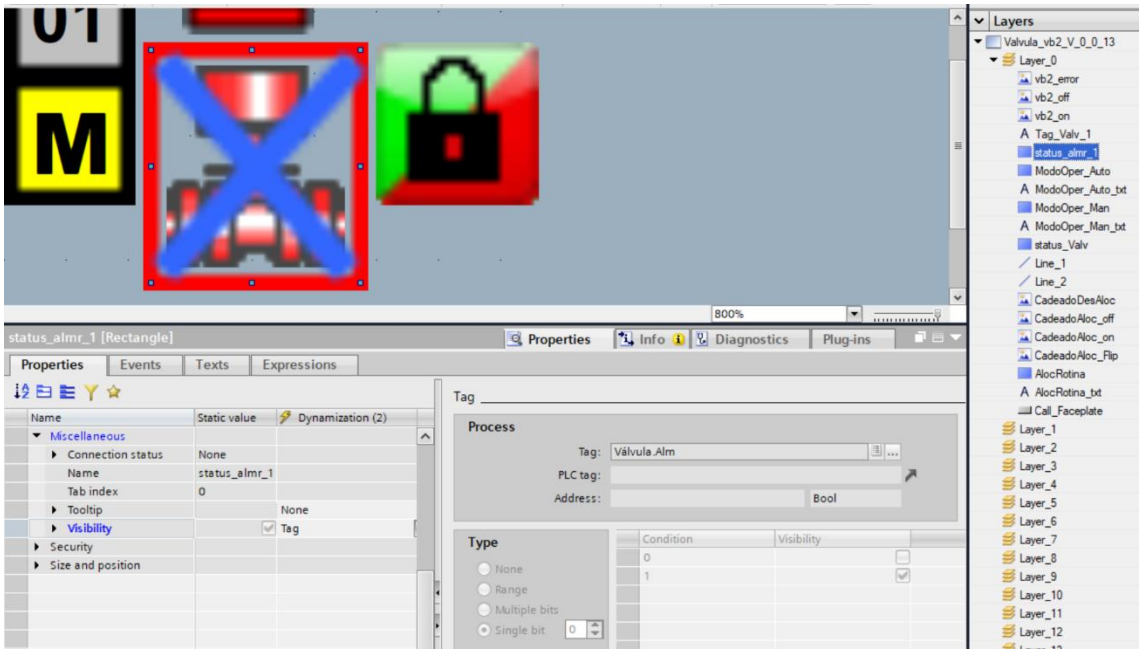
Caso a válvula venha a apresentar algum tipo de falha, sua animação de representação será pela figura de quadrado vermelho. Ela terá dois tipos de animação, uma de *“Visibility”* para mostrar o quadrado; outra para quando ela estiver a mostra, se manterá continuamente piscante. Ambas as animações utilizaram a memória “Válvula.*Alm*”. Como demonstrado na Figura 49 e na Figura 50

Figura 49 - Configuração da função *“flashing”* do alarme da válvula.



Fonte: Autoria própria.

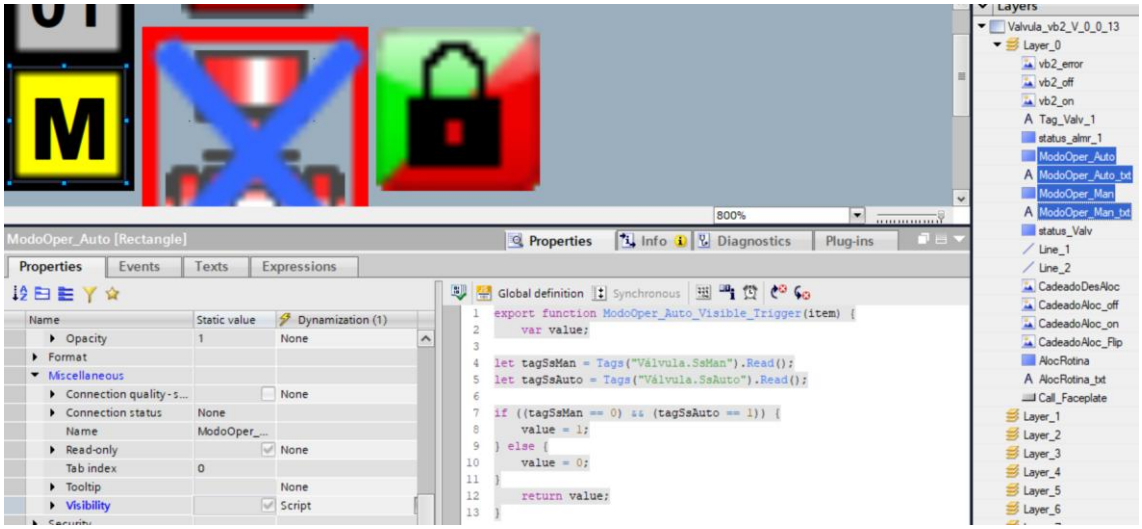
Figura 50 - Configuração de visibilidade do alarme da válvula.



Fonte: Autoria própria.

Para representar o estado de operação da válvula, podendo ser manual ou automático, utilizou-se figuras quadras e um texto com uma letra (“M” ou “A”). No mesmo formato de animação dos estados aberto e fechado das válvulas, foi utilizado as memórias “Válvula.*SsMan*” e “Válvula.*SsAuto*”. Conforme apresentado na Figura 51.

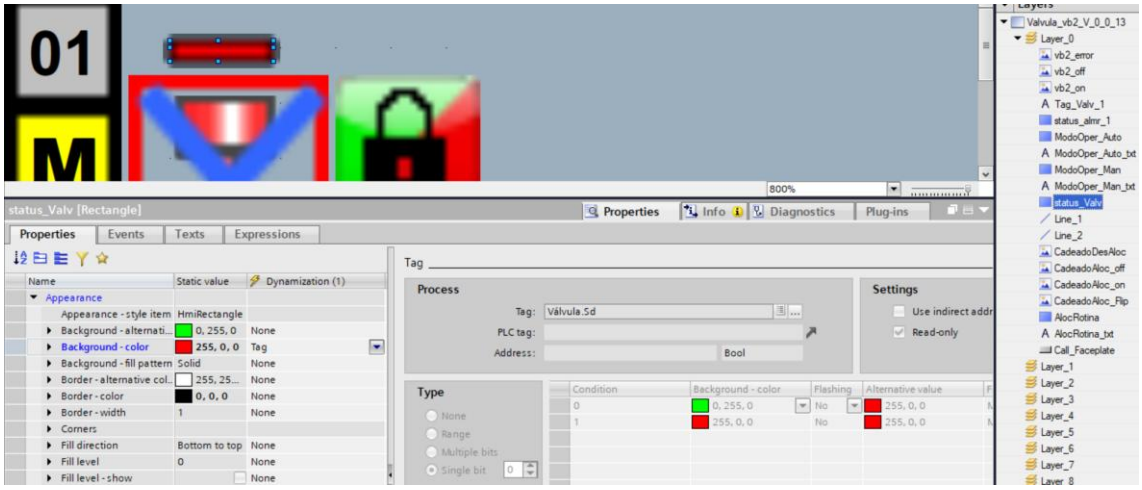
Figura 51 - Código de visibilidade dos estados operacionais das válvulas.



Fonte: Autoria própria.

Da mesma forma que se representa os estados (aberto/fechado) das válvulas, também é muito importante representar o sinal de comando para as válvulas; pois através de todas essas informações gráficas, é que se pode tirar algumas ponderações rápidas. Para esta animação mostrada na Figura 52, utilizou-se a memória “Válvula.Sd”.

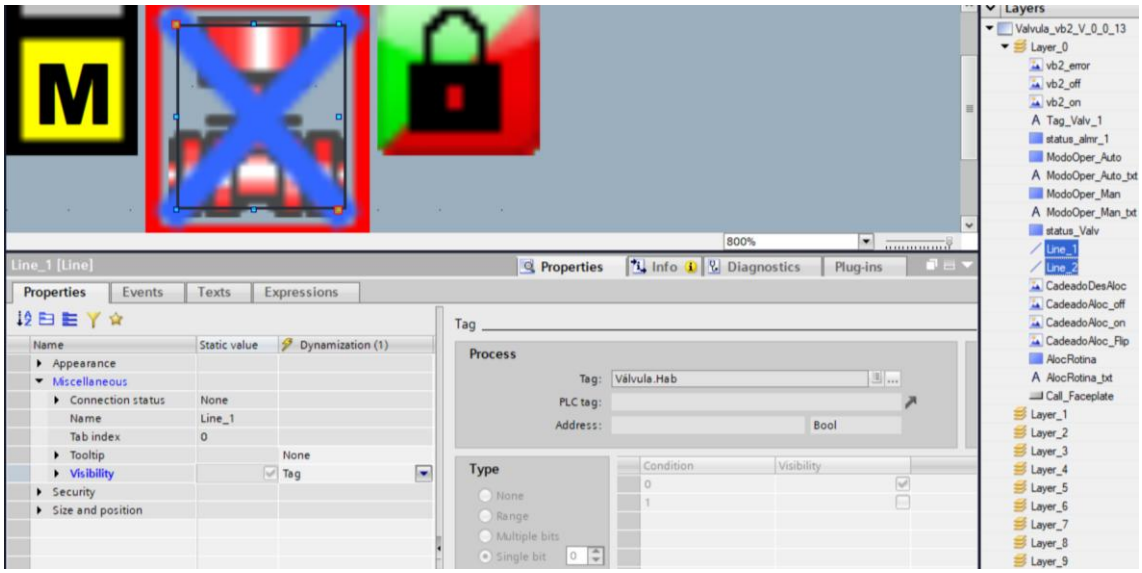
Figura 52 - Animação do sinal de comando das válvulas.



Fonte: Autoria própria.

Além das animações para os estados e sinais de comando das válvulas, também tem a animação de representação do sinal de intertravamento das válvulas, quando em uma situação de emergência ou máquina desligada, a válvula não aceita nenhum sinal de comando. Utilizando a memória “Válvula.*Hab*” na Figura 53.

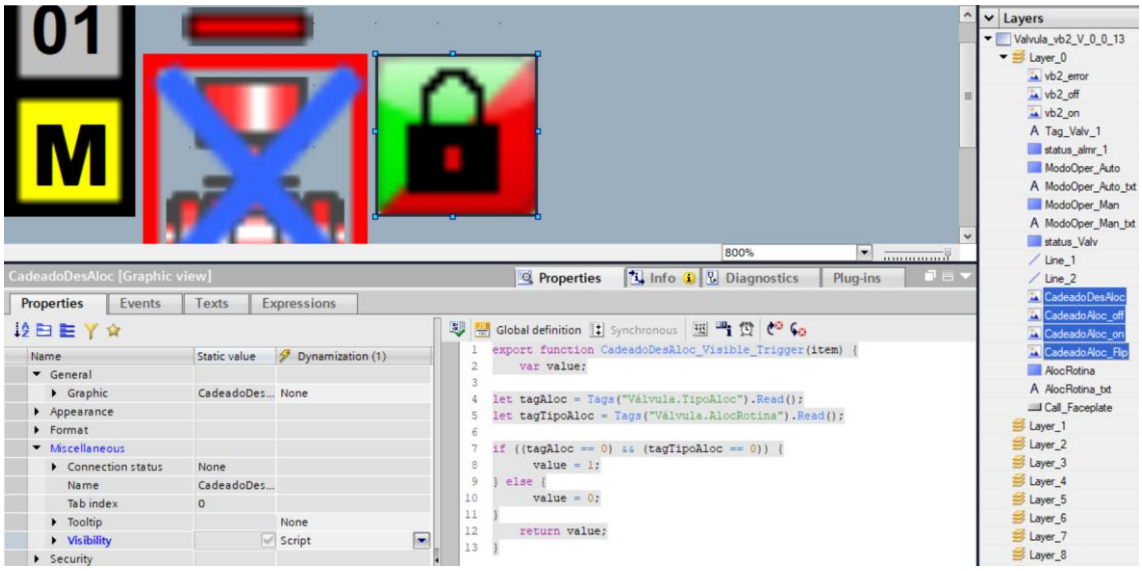
Figura 53 - Animação do sinal de intertravamento das válvulas.



Fonte: Autoria própria.

Além dos sinais de comando, a válvula também pode apresentar um estado de operação vinda de uma *Slot* de operação – ferramenta que opera um conjunto de comandos de diversos equipamentos em sequência para dar origem a um produto ou função da máquina – que tem sua animação vinda de um símbolo de cadeado. Utilizando duas memórias, "Válvula.TipoAloc" e "Válvula.AlocRotina" na Figura 54.

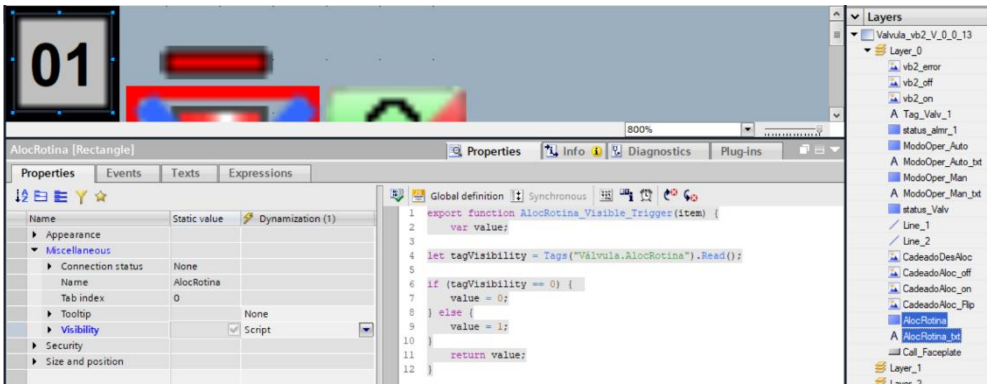
Figura 54 - Animação do sinal de comando vinda de *Slot* de operação para as válvulas.



Fonte: Autoria própria.

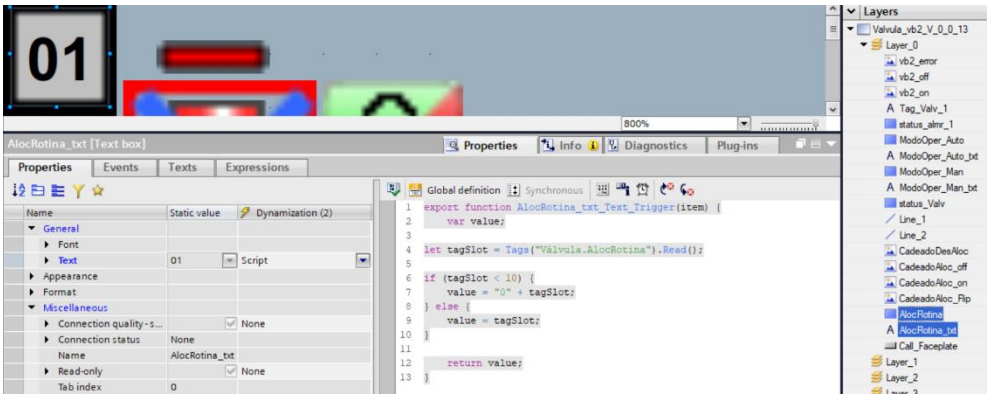
Como foi descrito antes, a válvula pode receber comando de um *Slot* de operação; porém além de animar o comando recebido, também se representará o número do *Slot* de que controlará esta válvula. Através da memória "Válvula.*AlocRotina*" se animará a visibilidade e texto dos objetos da Figura 55 e Figura 56.

Figura 55 - Código de visibilidade do *Slot* de operação da válvula.



Fonte: Autoria própria.

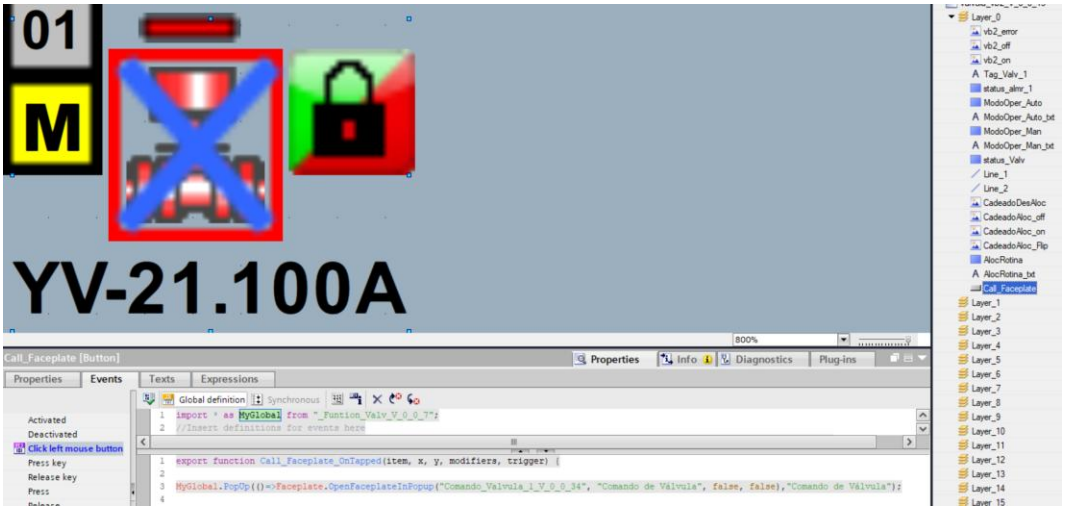
Figura 56 - Código de texto do número do *Slot* de operação da válvula.



Fonte: Autoria própria.

Como última animação da válvula, se desenvolveu um botão invisível para criar uma *faceplate* (subtela*)* para mostrar as mensagens de falha do equipamento e comandos manuais do operador. Segundo a Figura 57, foi importado para dentro da biblioteca de *Script* da válvula, um código que será explicado mais afrente. E na janela de chamada de animações do botão, foi inserido a função de chamada de *faceplate* com o cabeçalho “Comando de Válvula”.

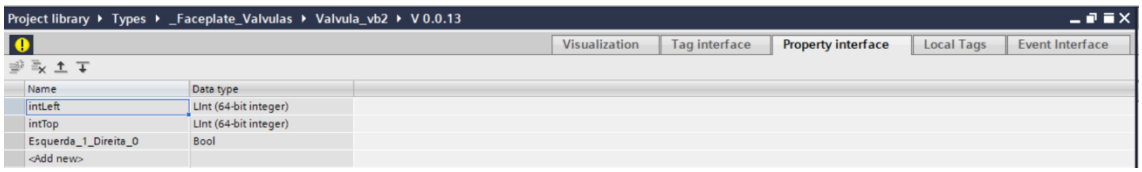
Figura 57 - Código chamada de *faceplate* de comandos da válvula.



Fonte: Autoria própria.

Atuando junto com a função de chamada de *faceplate*, foi criado memórias locais para determinar a posição do comando de válvula. Como mostrado na Figura 58.

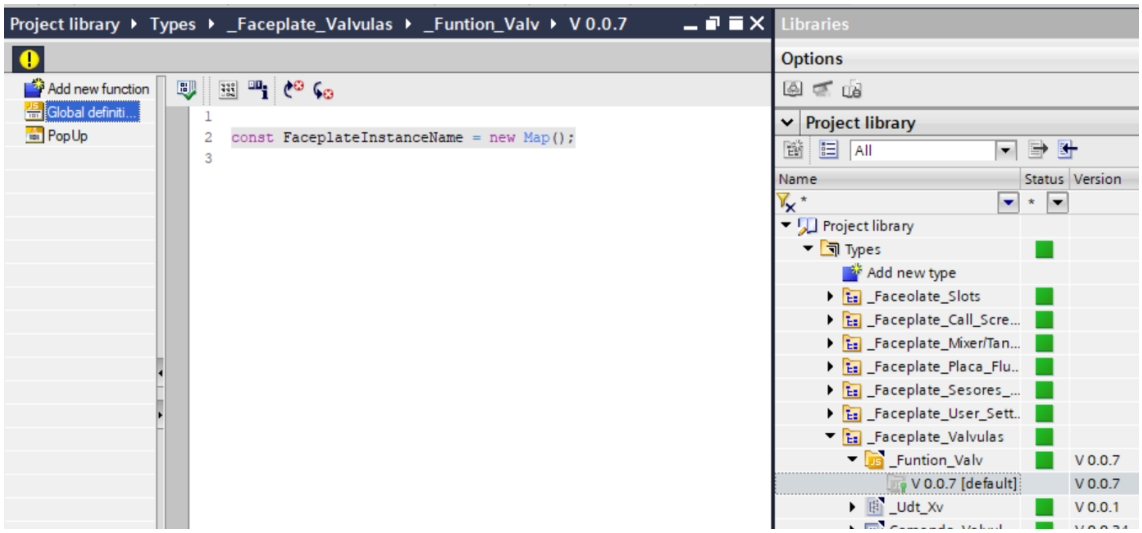
Figura 58 - Memórias locais do objeto da válvula.



Fonte: Autoria própria.

Na aba dentro do objeto de *Script* da biblioteca de válvulas, foi inserido a linha de código da Figura 59; ela tem a função de mapear uma instância de um *faceplate* (subtela) de comando para as válvulas, assim podendo ser chamada por outras válvulas em conjunto com diferentes *faceplates* de comando.

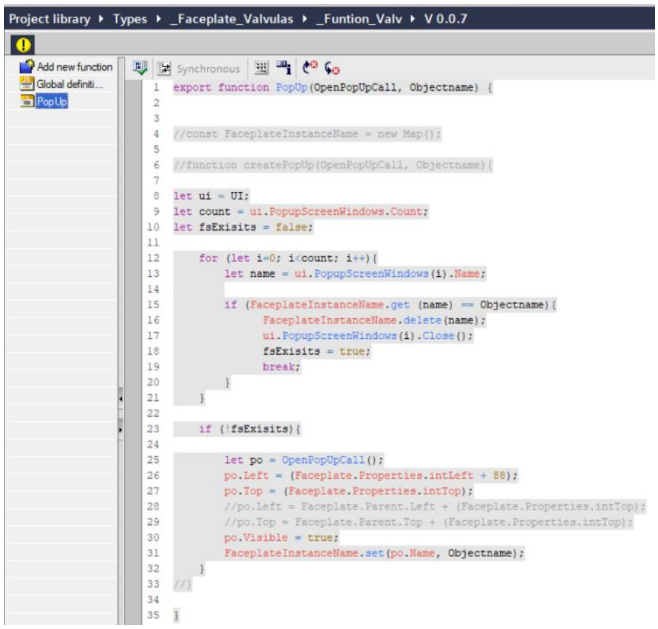
Figura 59 - Código de instância do *faceplate* de comando da válvula.



Fonte: Autoria própria.

Utilizando memórias de comando da raiz da HMI, foi criado o código de chamada de *faceplate* de comando das válvulas, apresentada pela Figura 60. Porém em conjunto com os parâmetros de posição que cada válvula irá oferecer, esse *faceplate* pode-se mostrar em diferentes posições.

Figura 60 - Código da chamada de um *faceplate*.



Fonte: Autoria própria.

3.8.3. Desenvolvimento de um *faceplate* de comando das válvulas

A partir dos parâmetros transmitidos dos objetos das válvulas, pode-se abrir o *faceplate* de comando para operador, como mostrado na Figura 61.

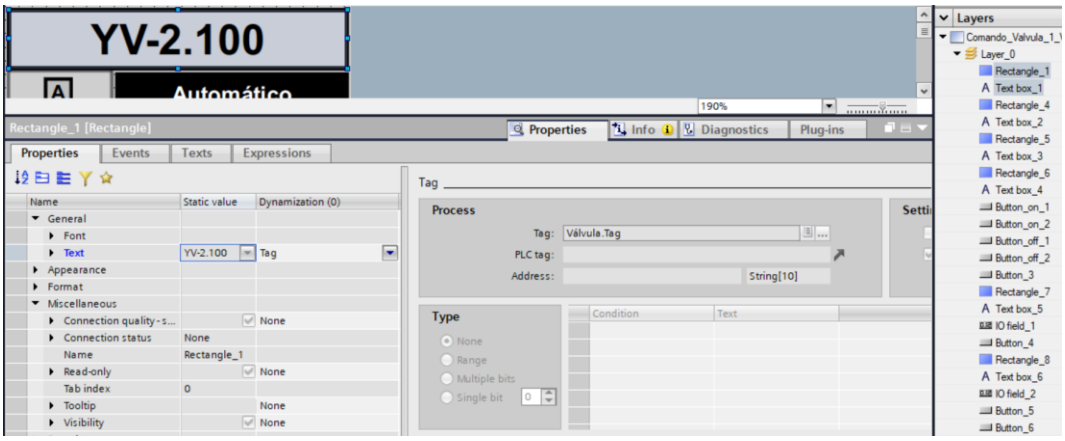
Figura 61 - Objetos de animação para o *faceplate* de comando da válvula.

Fonte: Autoria própria.

Da mesma forma que os objetos de válvulas, é necessário adicionar a memória de válvula geral, na aba de *"Tag Interface"* e na aba de *"Property Interface"*. Reveja a Figura 45 e a Figura 58.

Utilizando a mesma memória de nome da válvula para identificar o *faceplate* de comando, "Válvula.*Tag*". Apresentado na Figura 62.

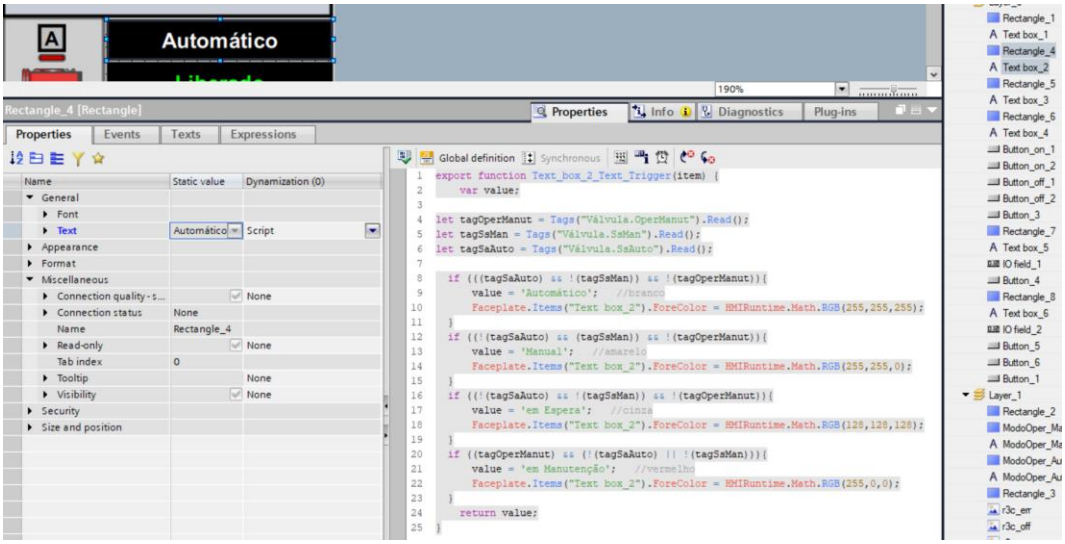
Figura 62 - Configuração do nome da válvula de comando.



Fonte: Autoria própria.

Antes foi apresentado de modo resumida os estados operacionais das válvulas, agora dentro dos *faceplates* de comando, se apresentarão todos os modos de operação incluindo uma animação de coloração para cada estado. Como demonstrada na Figura 63, utilizando as memórias "Válvula.*OperManut*", "Válvula.*SsMan*" e "Válvula.*SsAuto*".

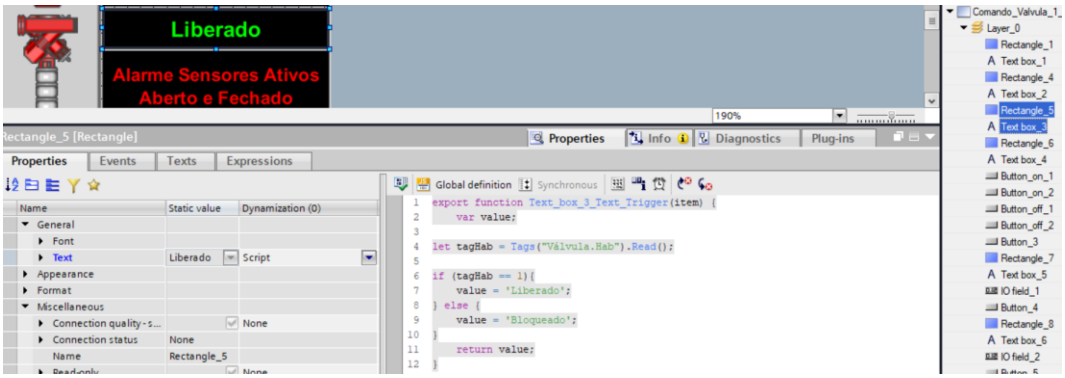
Figura 63 - Código de animação dos estados operacionais da válvula.



Fonte: Autoria própria.

Utilizando a memória “Válvula.*Hab*”, foi animado os textos de “Liberado” ou “Bloqueado” das válvulas. Que estão relacionados ao intertravamento das válvulas, anteriormente explicado. Mostrada pela Figura 64.

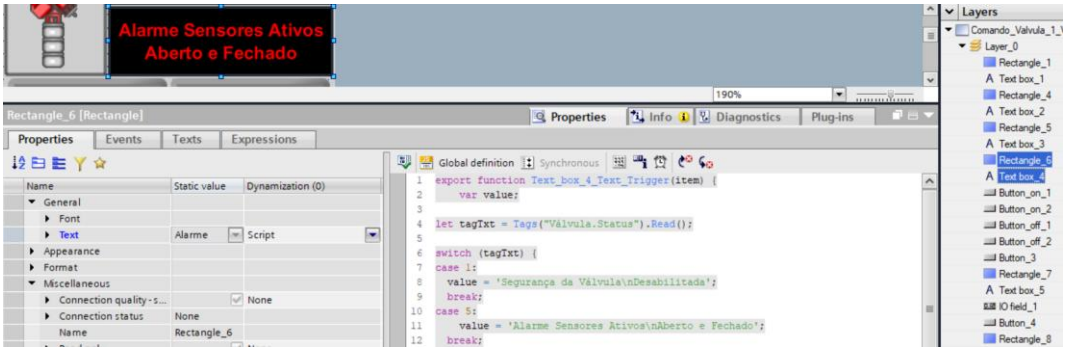
Figura 64 - Código de animação do sinal de intertravamento da válvula.



Fonte: Autoria própria.

Como função nova, diferente do objeto, o *faceplate* de comando apresentará textos de possíveis falhas que estão ocorrendo com o equipamento. A memória utilizada para essa função é “Válvula.*Status*”, e todo o código utilizado nesta animação será apresentado no Apêndice IV. Como demonstrada na Figura 65.

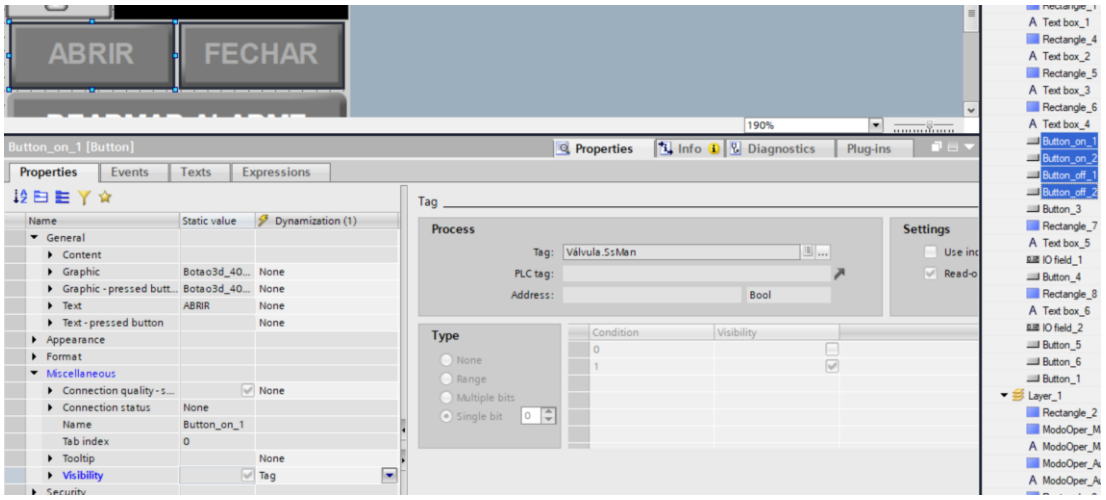
Figura 65 - Código de animação das mensagens de estado da válvula.



Fonte: Autoria própria.

Os botões de comando de abrir e fechar as válvulas, somente funcionaram quando as válvulas em questão estarão no estado operacional manual ou manutenção. Esta condição pode ser trava pelo código do PLC, ou como apresentado na Figura 66, pode ser escondido através da animação de visibilidade com a memória “Válvula.SsMan”.

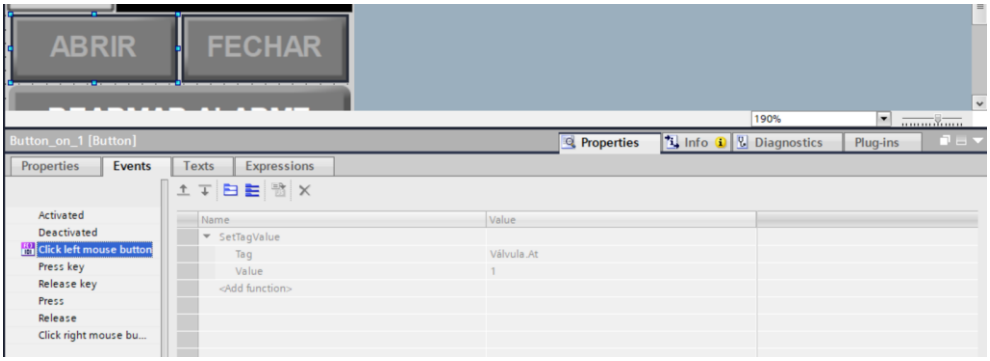
Figura 66 - Animação visibilidade dos botões de comando da válvula.



Fonte: Autoria própria.

Quando as válvulas estão nas condições corretas, os botões poderão acionar a válvula através da memória "Válvula.*At*". Como mostra a Figura 67.

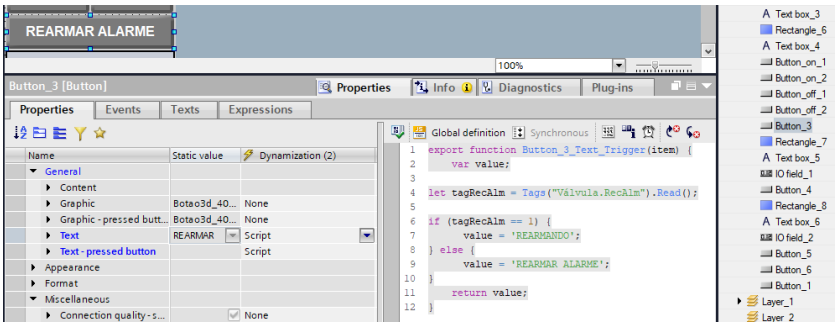
Figura 67 - Sinal de comando da válvula.



Fonte: Autoria própria.

Sempre que ocorrer algum tipo de alarme, pela lógica do PLC, ela terá o formato do tipo de alarme que precisa ser rearmado; esperando esse comando, qualquer comando para este equipamento será intertravado por segurança. Por isso em cada *faceplate* de comando há um botão de rearme, este possui dois tipos de animação; quando os alarmes estiverem sendo rearmado o texto do botão apresentará "REARMANDO", e quando o botão já estiver finalizado ou esperar o próximo comando de rearme, apresentará o texto "REARMAR ALARME". Como vista na Figura 68, com sua lógica de ação.

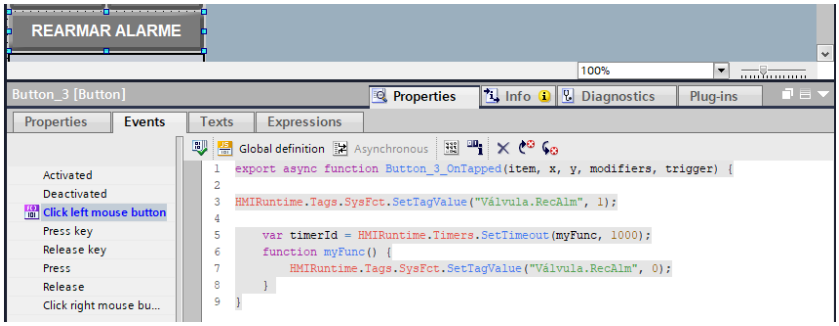
Figura 68 - Animação texto do botão de rearme.



Fonte: Autoria própria.

A segunda animação se dá pela ação do comando do botão, quando pressionado. Como se apresenta na Figura 69, uma vez pressionado, o comando de rearme se mantém ativo por um segundo, e logo após o comando se desativa, para garantir que toda a lógica de rearme dos alarmes seja completado.

Figura 69 - Comando do sinal de rearme.



Fonte: Autoria própria.

Os dois conjuntos de objetos apresentados na Figura 70, mostra informações para o operador do equipamento. Para fins de manutenção preventiva, um contator de quantidade total de comando para acionar a válvula é apresentada utilizando a memória “Válvula.TotalAtuador”; e um contator de quantidade total de retorno para posição inicial da válvula é apresentada pela memória “Válvula.TotalCilindro”.

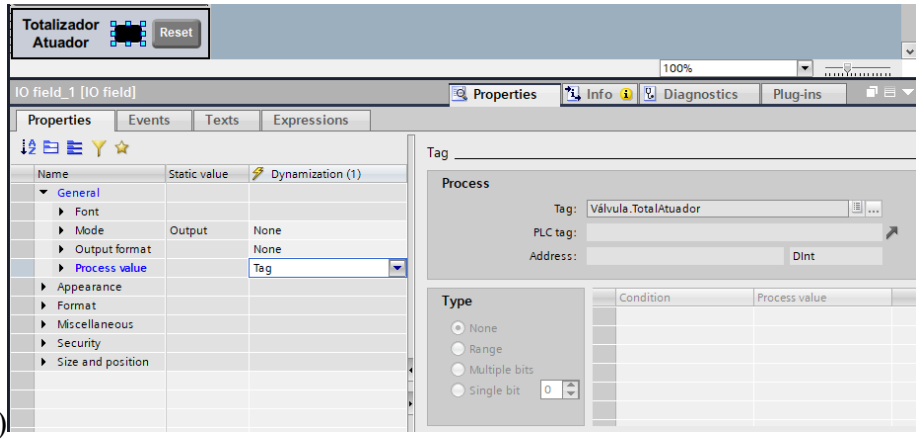
Figura 70 - Objetos utilizados nas informações de manutenção preventiva.

Fonte: Autoria própria.

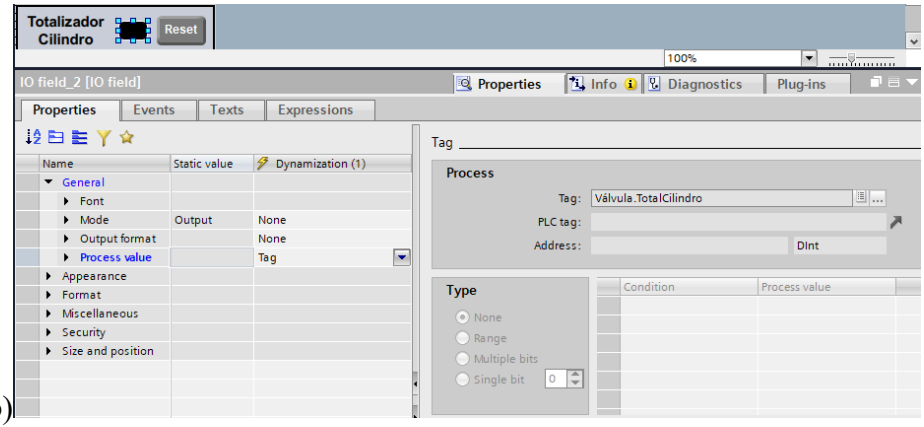
Para mostrar o número dessas memórias, se utiliza o objeto “IO_*field*”, como mostrada na Figura 71.

Figura 71 - Animação das informações dos totalizadores de a) Atuador e b) Cilindro.

a)



b)

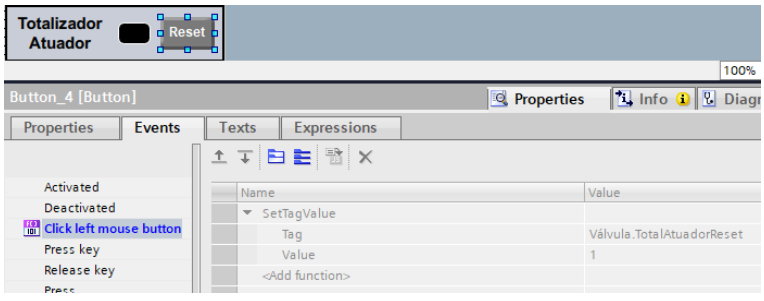


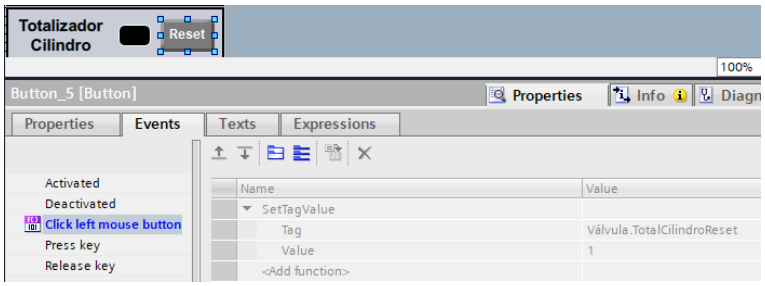
Fonte: Autoria própria.

Caso seja feita a substituição física desses equipamentos, é importante zerar as informações para a nova contagem. Para o totalizador de atuador se utiliza a memória “Válvula.TotalAtuadorReset” e para o totalizador do cilindro, a memória “Válvula.TotalCilindroReset”. Como mostrada na Figura 72.

Figura 72 - Animação de zerar as informações dos totalizadores de a) Atuador e b) Cilindro.

a)

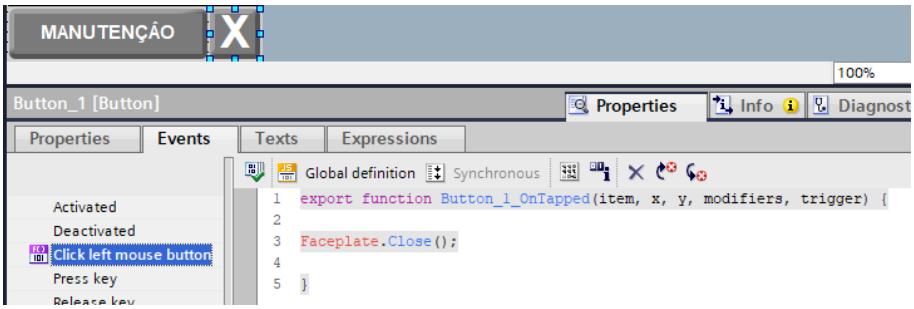


b) 

Fonte: Autoria própria.

Quando há a necessidade de fechar o *faceplate* de comando do equipamento, o botão com o texto de "X" pode ser pressionado; nela há a lógica para fechar o *faceplate*, como se mostra na Figura 73.

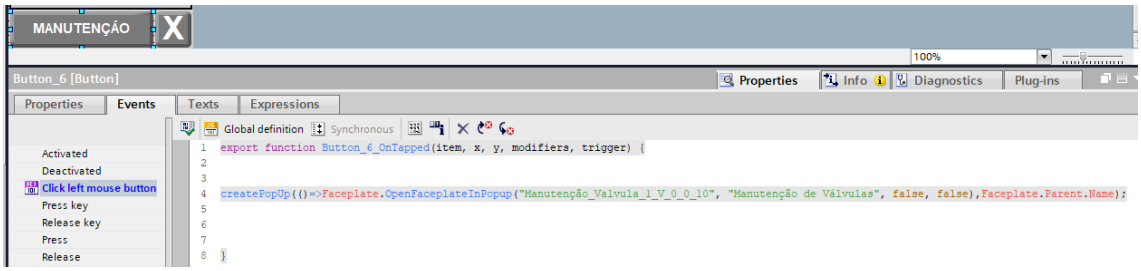
Figura 73 - Animação para fechar o *faceplate* de comando.



Fonte: Autoria própria.

Agora para abrir o *faceplate* de manutenção, onde há mais informações e configurações pertinentes ao pessoal de supervisor ou manutenção. O botão com o texto "MANUTENÇÃO" deve ser pressionado, como mostrada na Figura 74. Com a mesma lógica de criação de chamada de um *faceplate* (subtela) de comando, foi utilizada para abrir o *faceplate* de manutenção; porém com uma modificação para que os parâmetros de posição de tela sejam passados também.

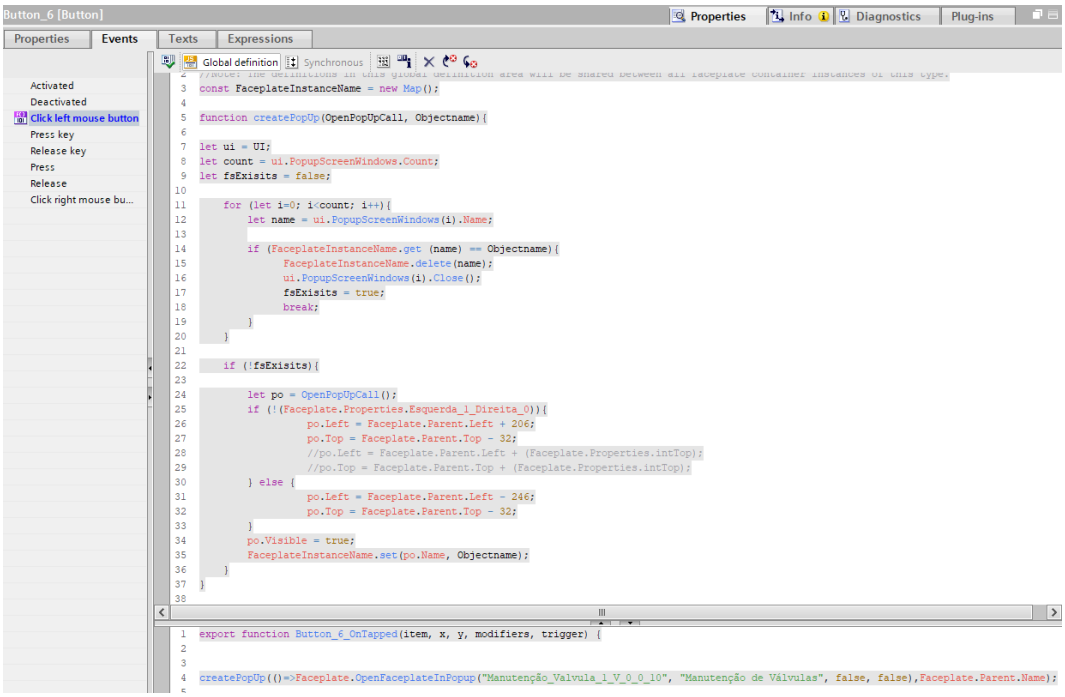
Figura 74 - Animação de chamada do *faceplate* de manutenção.



Fonte: Autoria própria.

Para melhor adequação na visualização dos *faceplates* de comando e manutenção. Foi utilizado alguns parâmetros físicos para que o segundo *faceplate* se apresentasse do lado direito ou esquerdo do *faceplate* de comando. Como mostra a Figura 75.

Figura 75 - Código da animação de chamada do *faceplate* de manutenção.



Fonte: Autoria própria.

Nesse *faceplate* de comando há também algumas animações de estado do equipamento; por exemplo o símbolo resumido do estado da válvula, automático (“A”) ou manual (“M”). Como mostra a Figura 76, e sua configuração de animação pode ser vista igual a Figura 51.

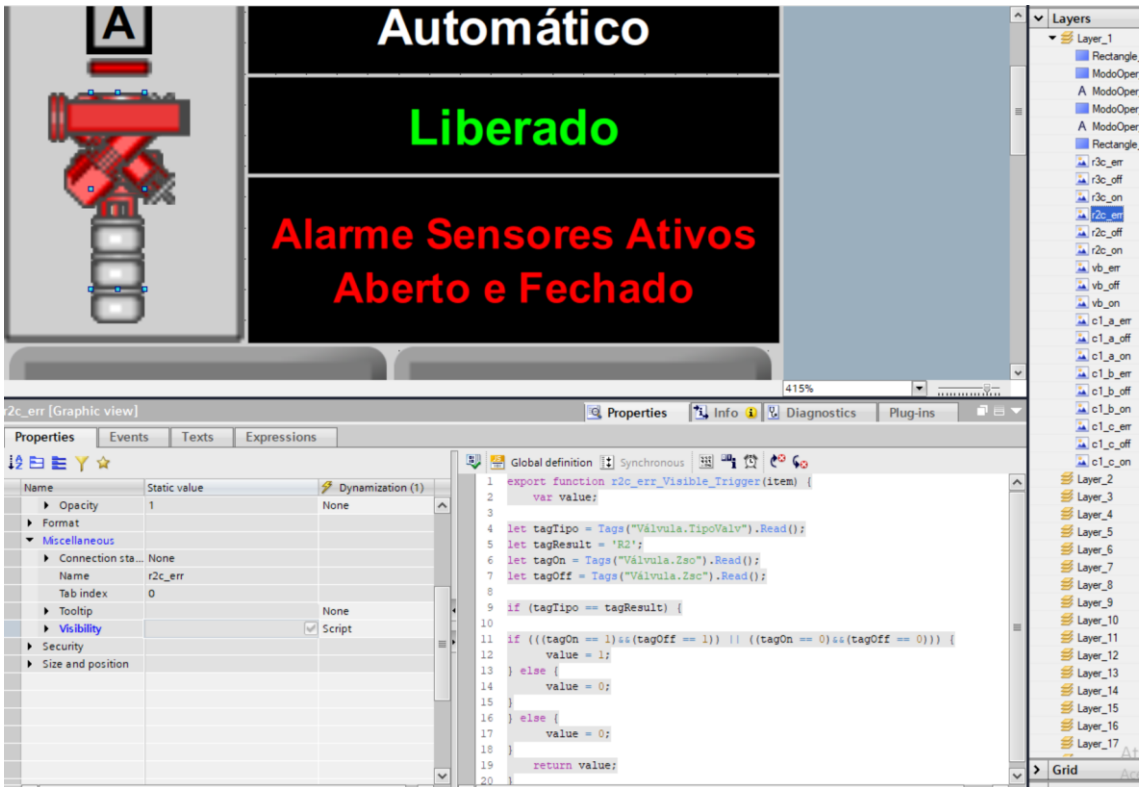
Figura 76 - Visualização resumida do equipamento no *faceplate* de comando.

Fonte: Autoria própria.

Da mesma forma que o objeto da válvula, há uma barra com animação para representar o comando da válvula utilizando a memória “Válvula.*Sd*”; o código desta animação pode ser visto na Figura 52.

Agora para a representação da imagem do equipamento, foi utilizada diversas imagens de tipos de válvulas. Ainda utilizando as memórias “Válvula.*Zsc*” e “Válvula.*Zso*” para representar a válvula desacionado e acionada, respectivamente. As diversas imagens das válvulas foram filtradas com o uso da memória “Válvula.TipoValv” para a melhor representação da válvula, do qual foi utilizada para abrir o *faceplate* de comando. Como se mostra na Figura 77.

Figura 77 - Código de animação das diversas imagens de válvulas no *faceplate*.



Fonte: Autoria própria.

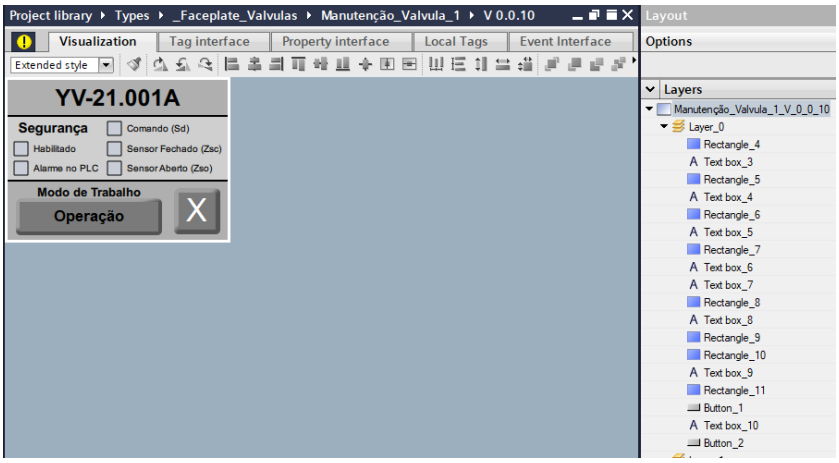
Seguindo a seguinte lista de tipo de válvula com a memória da válvula;

- “R3” – Válvula do tipo *Mux* de 3 vias (GEA, 2024);
- “R2” – Válvula do tipo *Mux* de 2 vias;
- “VB” – Válvula do tipo borboleta;
- “CA” – Válvula do tipo A para o lançamento da pastilha PIG;
- “CB” – Válvula do tipo B para o lançamento da pastilha PIG;
- “CC” – Válvula do tipo C para o lançamento da pastilha PIG.

3.8.4. Desenvolvimento do *faceplate* de manutenção das válvulas.

O *faceplate* de manutenção possui algumas informações a mais para o equipamento e um botão para ativar o modo de manutenção do equipamento; como mostra na Figura 78.

Figura 78 - Objetos utilizados no *faceplate* de manutenção.

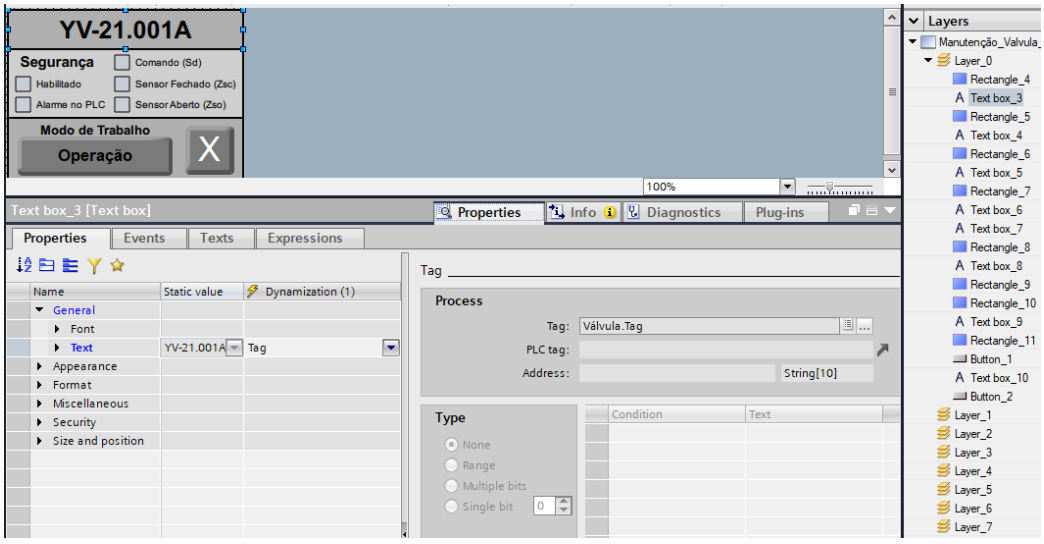


Fonte: Autoria própria.

Da mesma forma que *faceplate* de comando, é necessário adicionar a memória de válvula geral, na aba de *"Tag Interface"* e na aba de *"Property Interface"*. Reveja a Figura 45 e a Figura 58.

Utilizando a mesma memória de nome da válvula para identificar o *faceplate* de manutenção, "Válvula.*Tag*". Apresentado na Figura 79.

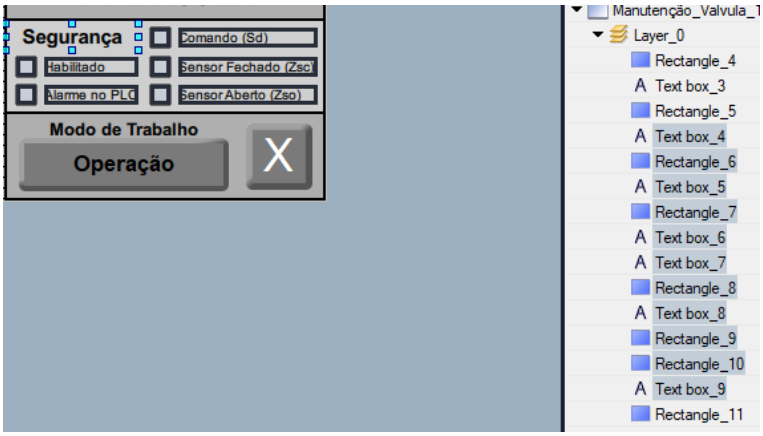
Figura 79 - Animação para apresentar o nome do equipamento.



Fonte: Autoria própria.

A Figura 80 apresenta os objetos utilizados nas informações adicionais dos equipamentos, sendo estes o sinal de intertravamento (“Válvula.*Hab*”), o sinal de alarme do PLC (“Válvula.*Alm*”), o sinal de comando para a válvula (“Válvula.*Sd*”), o sinal de posição da válvula no estado desacionado (“Válvula.*Zsc*”), e o sinal de posição da válvula no estado acionado (“Válvula.*Zso*”).

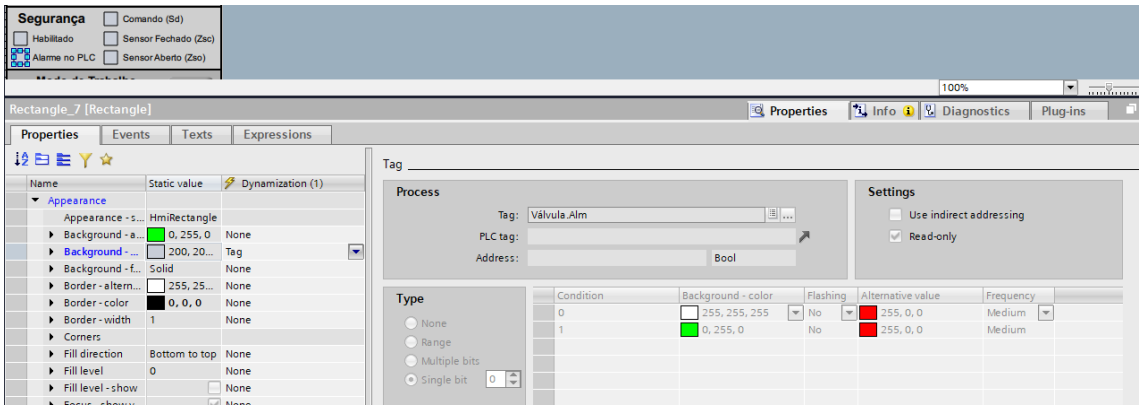
Figura 80 - Objetos utilizados na apresentação das informações adicionais.



Fonte: Autoria própria.

Todas as informações são representadas pela animação dos objetos quadrados menores do *faceplate*. E todas as animações seguem o mesmo tipo de configuração, apenas alterando a memória da válvula para sua ativação. Como mostrada na Figura 81.

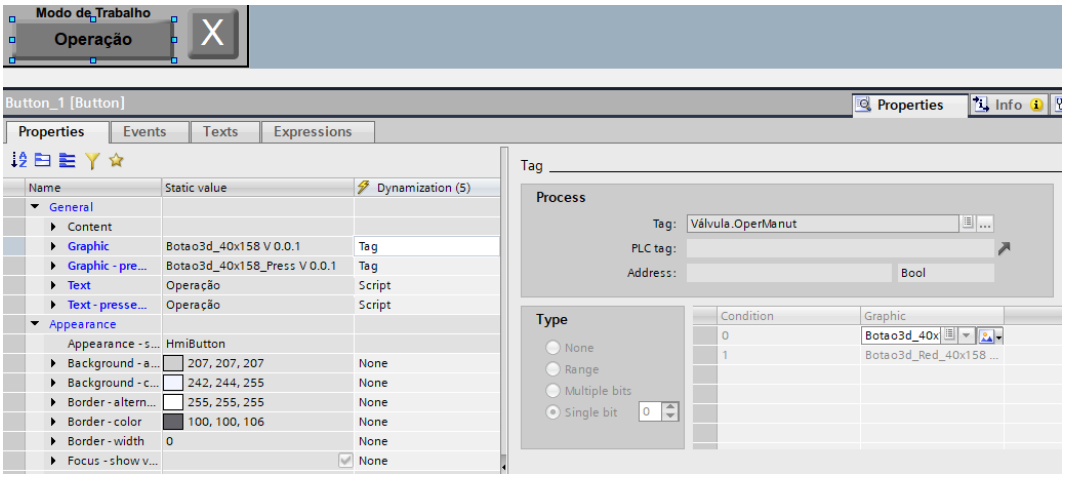
Figura 81 - Configuração da animação da coloração do objeto quadrado das informações adicionais.



Fonte: Autoria própria.

O botão de alteração do modo de trabalho para manutenção, possui 4 tipos de animações, porém todas utilizando a mesma memória de ativação “Válvula.*OperManut*”. A primeira animação se dá pela coloração do botão, utilizando a função de inserir uma figura no objeto do botão. Neste objeto foi desenvolvido imagens de botão normal e pressionado nas cores cinza e vermelho; e estes foram inseridos nos parâmetros apresentados na Figura 82.

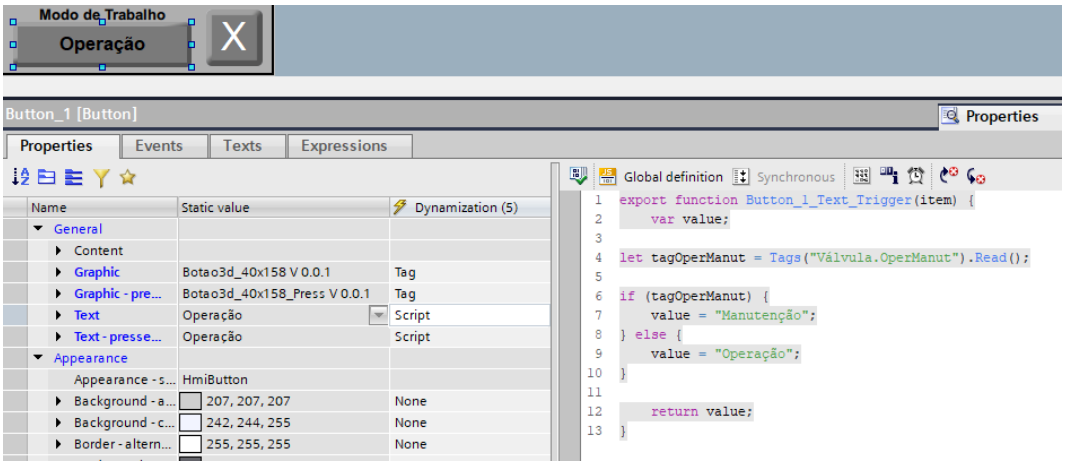
Figura 82 - Configuração dos parâmetros das figuras dos estados do botão de manutenção.



Fonte: Autoria própria.

Para representar o estado atual do modo de trabalho do equipamento, foi utilizado a animação de texto para indicar o estado da memória “Válvula.*OperManut*”. Para o estado “0” da memória o texto apresenta “Operação” e para o estado “1” da memória, o texto apresenta “Manuteção”. Como demonstrada na Figura 83.

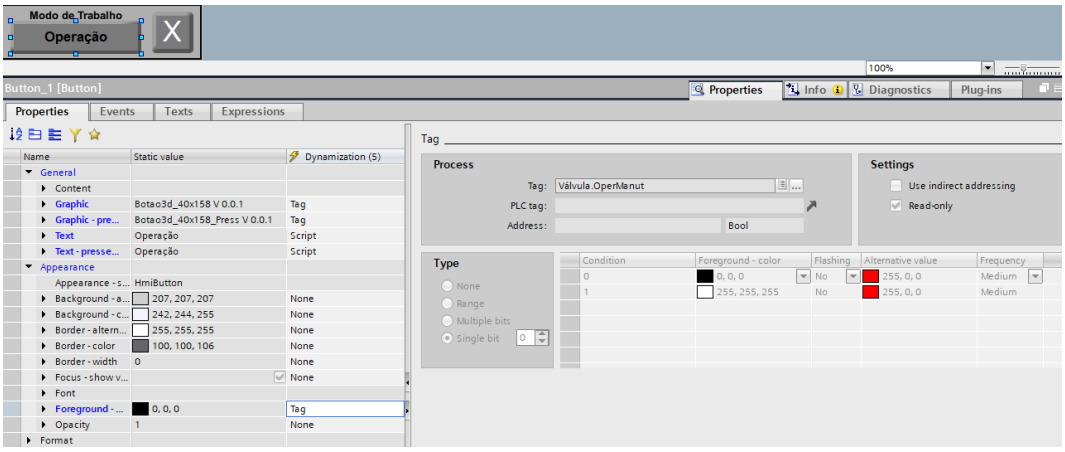
Figura 83 - Configuração da animação de texto do botão de manutenção.



Fonte: Autoria própria.

Como a coloração do texto apresentado era um problema para os diferentes tipos de botões apresentados, foi utilizado a mesma condição para alterar sua coloração; como demonstrada na Figura 84.

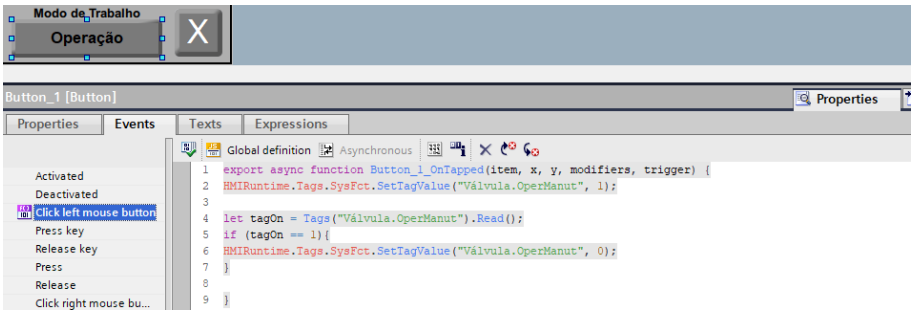
Figura 84 - Configuração da coloração do texto do botão de manutenção.



Fonte: Autoria própria.

E por fim, o comando para ativar e desativar a memória de manutenção se dá pelo código apresentado na Figura 85. Onde ocorre uma avaliação do estado atual da memória, para que o comando seja enviado para inverter o estado.

Figura 85 - Configuração da coloração do texto do botão de manutenção.



Fonte: Autoria própria.

E para fechar o *faceplate* de manutenção, existe o mesmo botão de ação identificada por "X"; cujo código da animação segue o mesmo formato do *faceplate* de comando, reveja a Figura 73.

3.8.5. Desenvolvimento dos objetos de animação dos sensores digitais

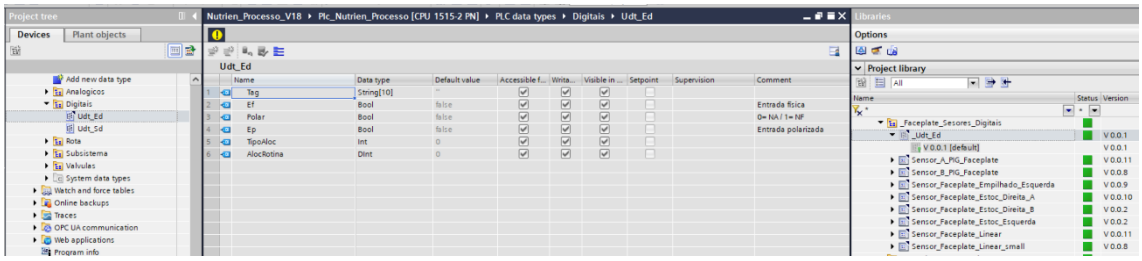
Da mesma forma que o objeto de válvula, neste projeto foi desenvolvido os objetos dos sensores digitais em suas diversas posições e compartimentalizados em pasta dento da biblioteca, como mostrada na Figura 86.

Figura 86 - Pasta da biblioteca para os sensores digitais.

Fonte: Autoria própria.

Para adicionar o tipo de dado dos sensores digitais, em vez de criar cada dado por si, é mais rápido arrastar o tipo de dado do sensor digital da pasta do PLC para dentro da biblioteca da HMI, desta forma se consegue sincronizar os dados; como mostra a Figura 87.

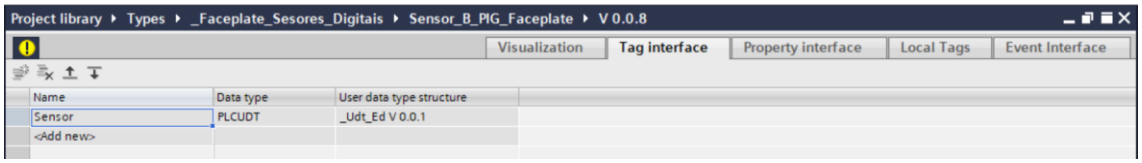
Figura 87 - Tipo de dados utilizados nas animações dos sensores digitais.



Fonte: Autoria própria.

Após criar um objeto novo para representar os sensores digitais, toda animação desenvolvida deve utilizar um padrão de memória para poder trabalhar depois com o vetor de equipamento. Assim como os objetos das válvulas, foi inserida uma memória padrão para os sensores digitais utilizando o tipo de dado de entradas digitais; como mostra a Figura 88.

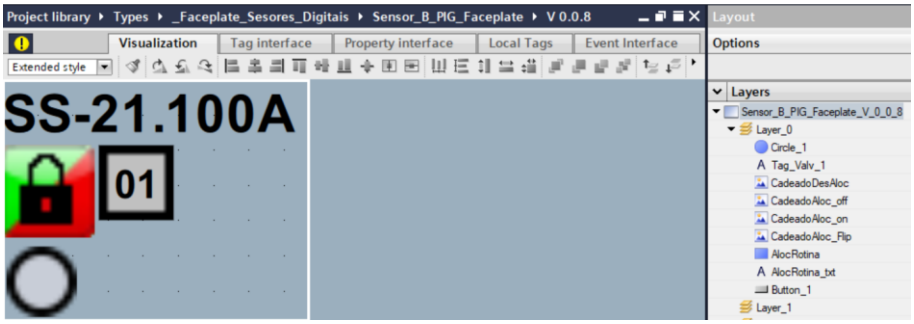
Figura 88 - Adição de memória de sensor geral.



Fonte: Autoria própria.

Todos os tipos de sensores digitais utilizados neste projeto foram representados apenas com o objeto de um círculo, porém os diversos tipos de objetos de sensores desenvolvidos na biblioteca se devem apenas ao tamanho do círculo ou as diferentes posições dos objetos da animação; como mostra a Figura 89.

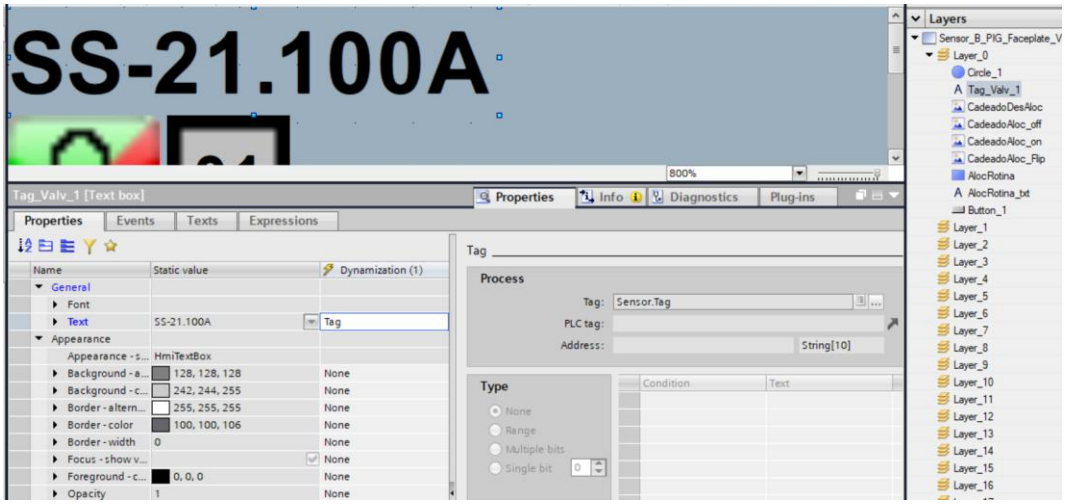
Figura 89 - Objetos utilizados na representação do sensor digital.



Fonte: Autoria própria.

Para representar o nome do equipamento foi utilizado a memória "Sensor.*Tag*", como apresenta a Figura 90.

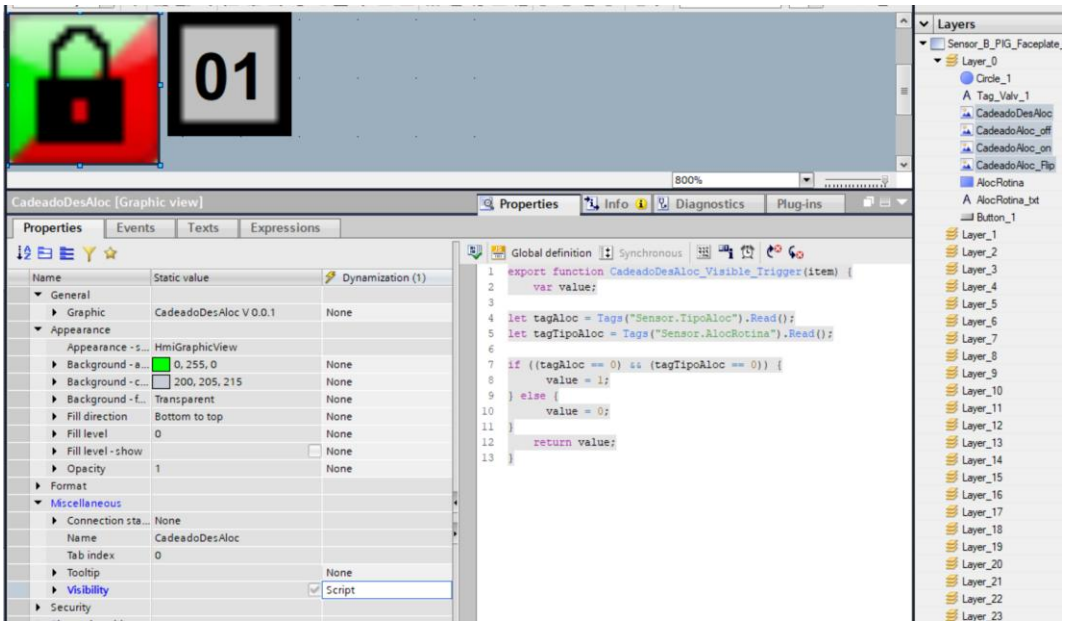
Figura 90 - Animação para o texto do nome do sensor.



Fonte: Autoria própria.

Igual aos objetos das válvulas, o sensor também pode apresentar um estado de operação vinda de uma *Slot* de operação – ferramenta que opera um conjunto de comandos de diversos equipamentos em sequência para dar origem a um produto ou função da máquina – que tem sua animação vinda de um símbolo de cadeado. Utilizando duas memórias, "Válvula.TipoAloc" e "Válvula.AlocRotina" na Figura 91.

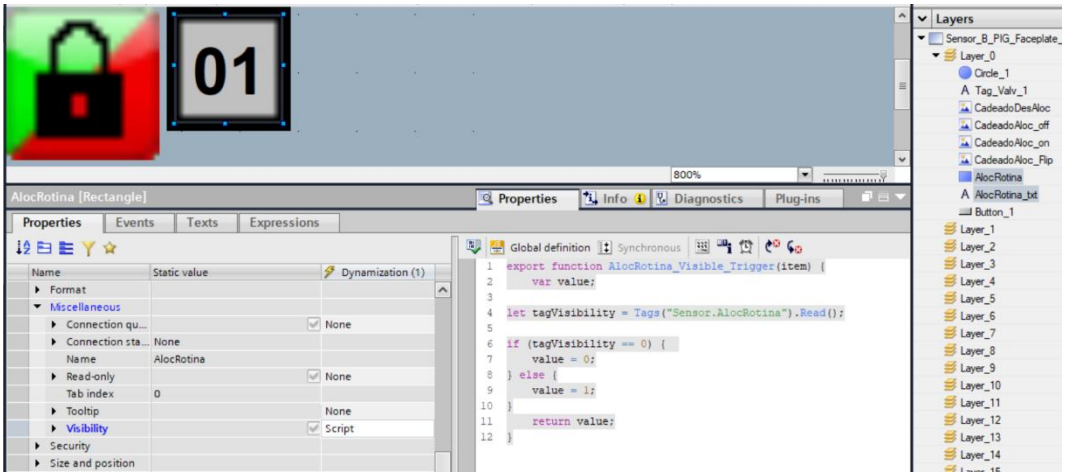
Figura 91 - Animação do sinal de comando vinda de *Slot* de operação para os sensores.



Fonte: Autoria própria.

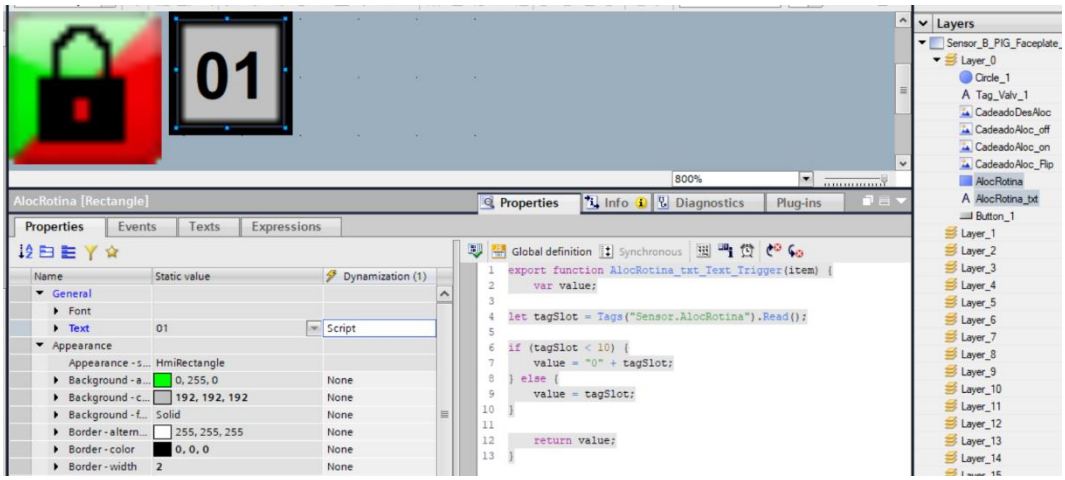
Como foi descrito antes, o sensor pode receber comando de um *Slot* de operação; porém além de animar o comando recebido, também se representará o número do *Slot* de que controlará o sensor. Através da memória "Sensor.*AlocRotina*" se animará a visibilidade e texto dos objetos da Figura 92 e Figura 93.

Figura 92 - Código de visibilidade do *Slot* de operação do sensor.



Fonte: Autoria própria.

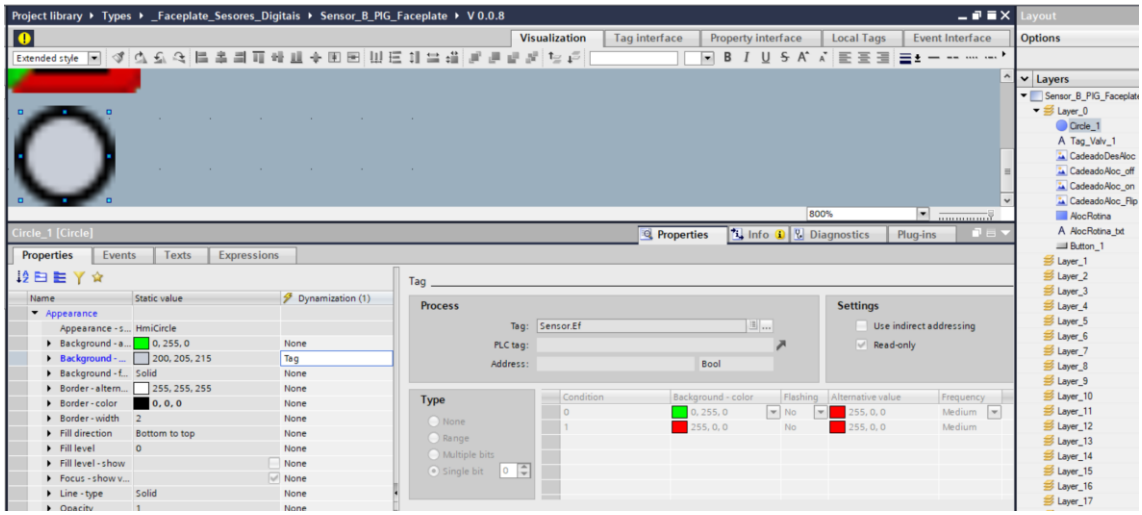
Figura 93 - Código de texto do número do *Slot* de operação do sensor.



Fonte: Autoria própria.

Utilizando a memória "Sensor.*Ef*" foi configurada a animação da coloração do círculo para representar o sinal atual do sensor digital, como apresentada na Figura 94

Figura 94 - Configuração da animação do estado do sensor.

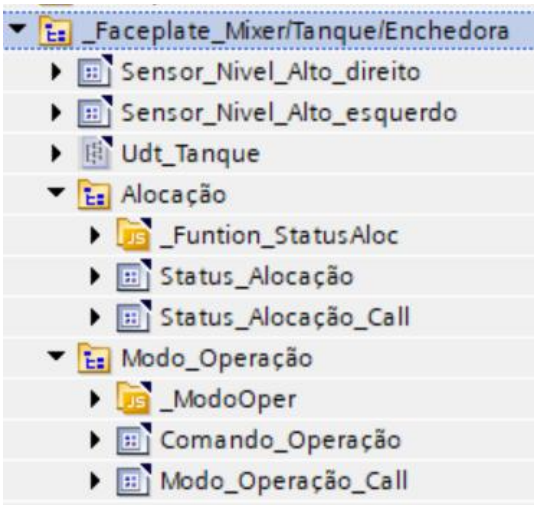


Fonte: Autoria própria.

3.8.6. Desenvolvimento dos objetos de animação do estado dos tanques

Da mesma forma que o objeto de válvula e sensores, neste projeto foi desenvolvido os objetos de estado do modo de alocação e modo de operação dos tanques; e compartimentalizados em pasta dentro da biblioteca, como mostrada na Figura 95.

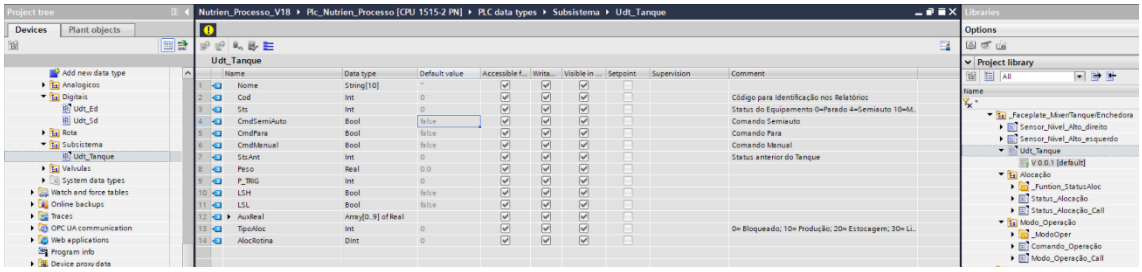
Figura 95 - Pasta da biblioteca para os estados dos tanques.



Fonte: Autoria própria.

Para adicionar o tipo de dado dos tanques, em vez de criar cada dado por si, é mais rápido arrastar o tipo de dado do tanque da pasta do PLC para dentro da biblioteca da HMI, desta forma se consegue sincronizar os dados; como mostra a Figura 96.

Figura 96 - Tipo de dados utilizados nas animações dos tanques.

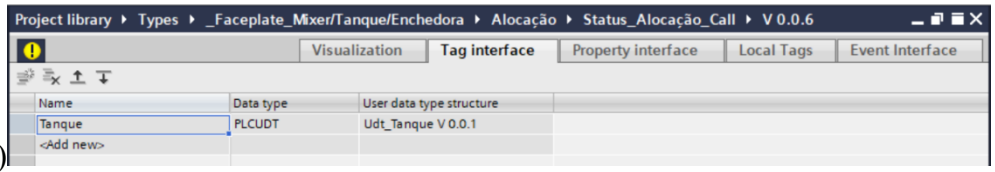


Fonte: Autoria própria.

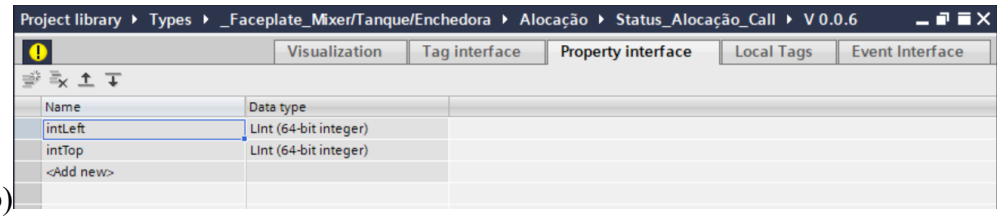
Após criar um objeto novo para representar qualquer estado dos tanques, toda animação desenvolvida deve utilizar um padrão de memória para poder trabalhar depois com o vetor de equipamento. Assim como os objetos das válvulas e sensores, foi inserida uma memória padrão para os tanques utilizando o tipo de dado de entradas digitais; como mostra a Figura 97.

Figura 97 - Adição de a) Memória de tanque geral e b) Memória local para posição do *faceplate*.

a)



b)



Fonte: Autoria própria.

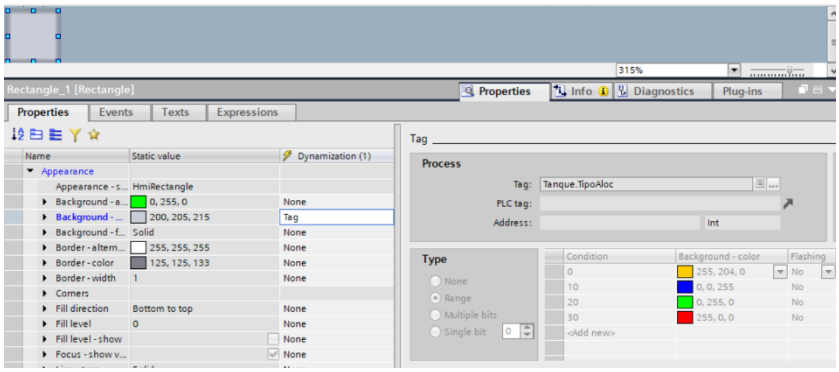
3.8.7. Desenvolvimento do objeto resumido do estado de alocação dos tanques

Foi desenvolvido dois tipos de representação dos estados dos tanques, um conjunto de objeto resumido e *faceplate* de comando para o estado de alocação e para o estado do modo de operação. Para o estado de alocação, se refere para qual função o tanque selecionado será utilizado; podendo escolher entre;

- "BLOQUEADO" – quando qualquer tipo de produção se finaliza ou está bloqueado para receber qualquer tipo de produção; cor laranja.
- "PRODUÇÃO" – quando o tanque está pronto para receber qualquer tipo de produção de produto, lembrando que o tanque só pode produzir um produto por produção; cor azul escuro.
- "ESTOCAGEM" – quando o tanque está na fase de transferência do produto para os tanques de estocagem; cor verde claro.
- "LIMPEZA" – quando o tanque está pronto para receber uma ação de limpeza dos seus equipamentos, tanques e tubulação de transferência; cor vermelho.

Todos os tipos de alocação do tanque possuem uma coloração que representa sua ação de forma resumida, essa representação foi utilizada na animação do objeto da Figura 98; pelo uso da memória "Tanque.TipoAloc".

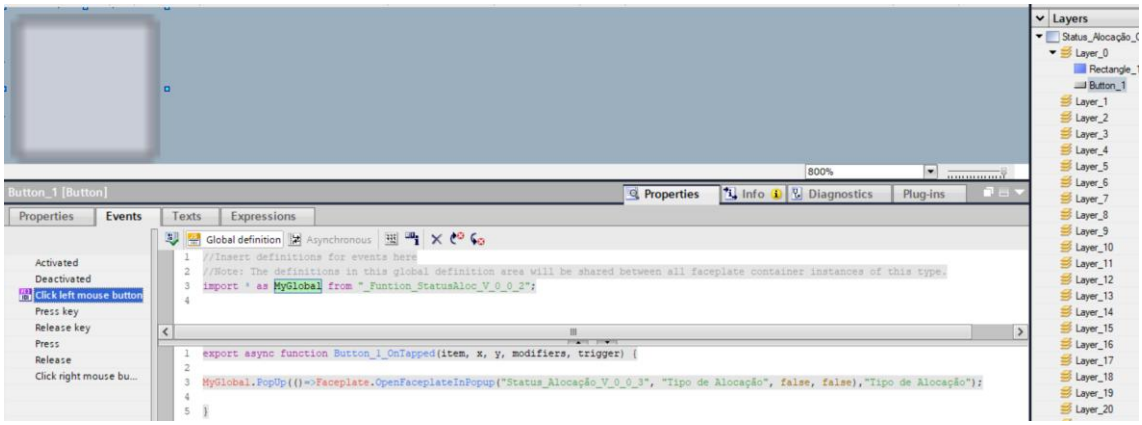
Figura 98 - Objeto utilizado na representação resumida do estado de alocação do tanque.



Fonte: Autoria própria.

Este objeto também possui um botão invisível para chamar o *faceplate* (subtela) de comando de seleção do tipo de alocação do tanque. Como mostrada na Figura 99, se utiliza um *script* externo na área de código global do objeto para a chamada do *faceplate*; seu código utilizado se encontra no Apêndice V com parâmetro de posicionamento.

Figura 99 - Animação da chamada do *faceplate* de comando do tipo de alocação do tanque.



Fonte: Autoria própria.

3.8.8. Desenvolvimento do *faceplate* de comando do estado de alocação dos tanques

Para o *faceplate* de comando da seleção do tipo de alocação do tanque, foi utilizado uma imagem base, desenvolvida do *software* de imagem, para representar as possíveis transições de estados e os nomes destes estados dos tanques, como mostrada na Figura 100.

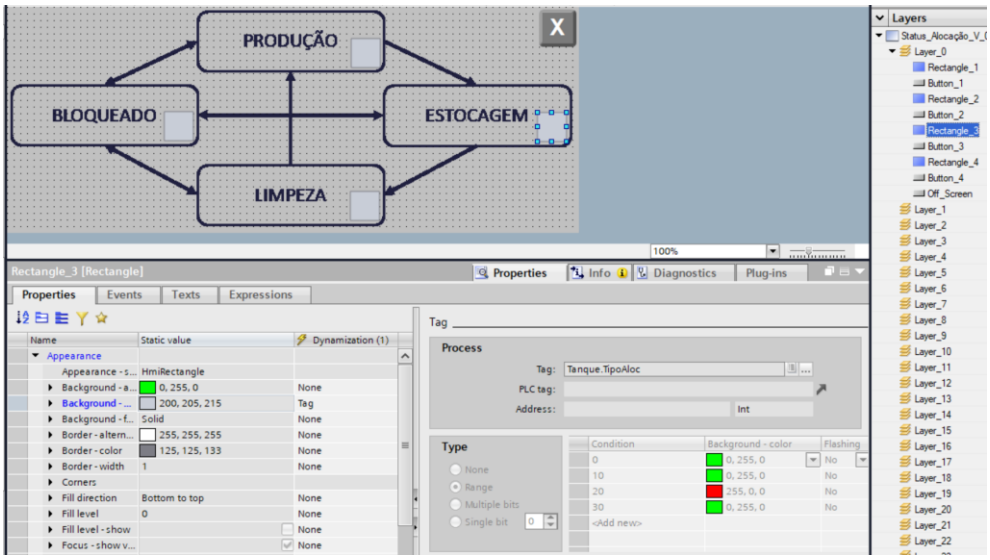
Figura 100 - *Faceplate* de comando da seleção dos tipos de alocação dos tanques.

Fonte: Autoria própria.

Todas as animações deste *faceplate* utilizou a mesma memória padrão do objeto resumido, sua criação pode ser vista pela Figura 97.

Este *faceplate* possui os mesmos objetos de quadrados que foi utilizado na Figura 98, porém com a diferença que quando o estado estiver selecionado, apresenta uma coloração avermelhado e o outros estado se mantém em verde claro; como se apresenta na Figura 101.

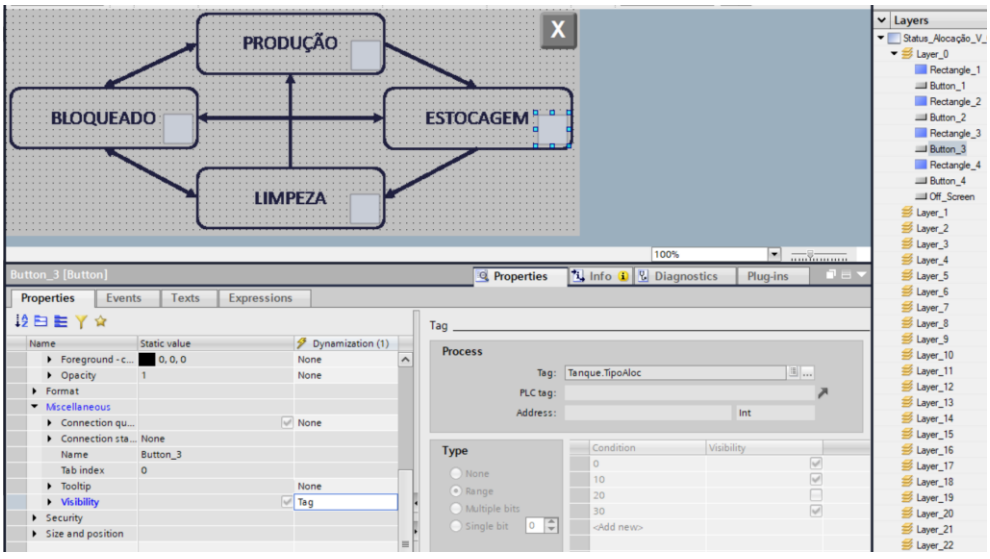
Figura 101 - Animação do estado de alocação selecionado.



Fonte: Autoria própria.

E em cada objeto quadrado, também possui um botão invisível que possui dois tipos de animação, uma delas apresenta o botão do estado selecionado uma vez, após pressionado o botão fica escondido para não ser pressionado de novo; como mostrada na Figura 102.

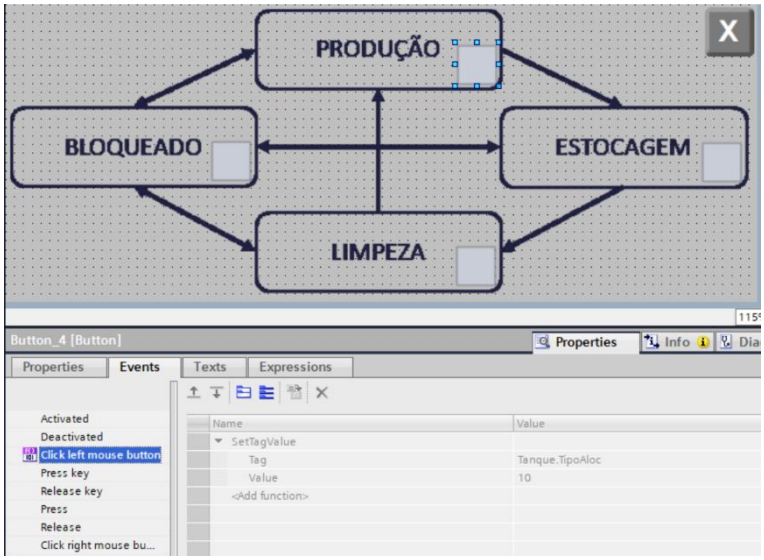
Figura 102 - Animação de invisibilidade dos botões de seleção do estado de alocação dos tanques.



Fonte: Autoria própria.

A segunda animação do botão se apresenta no comando de seleção do estado do estado de alocação do tanque. Cada botão dos estados envia um comando com um número específico para a memória do tanque “Tanque.TipoAloc”; como se mostra na Figura 103.

Figura 103 - Animação do comando dos botões de seleção do estado de alocação dos tanques.



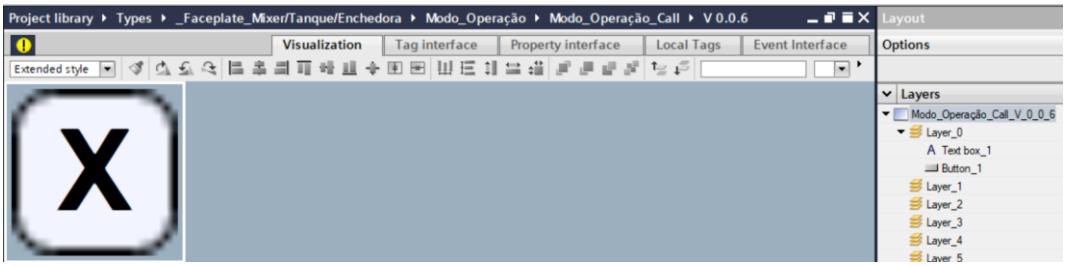
Fonte: Autoria própria.

E para fechar o *faceplate* de comando do tipo de alocação do tanque, existe um botão de ação identificada por “X”; cujo código da animação segue o mesmo formato do *faceplate* de comando, reveja a Figura 73.

3.8.9. Desenvolvimento do objeto resumido do estado do modo de operação dos tanques

Para o modo de operação do tanque, podendo ser manual, automático ou espera, também possui um objeto resumido para sua representação e um *faceplate* de comando para sua alteração. A Figura 104, apresenta o objeto resumido das informações do modo de operação.

Figura 104 - Objeto utilizados na representação resumida do modo de operação do tanque.

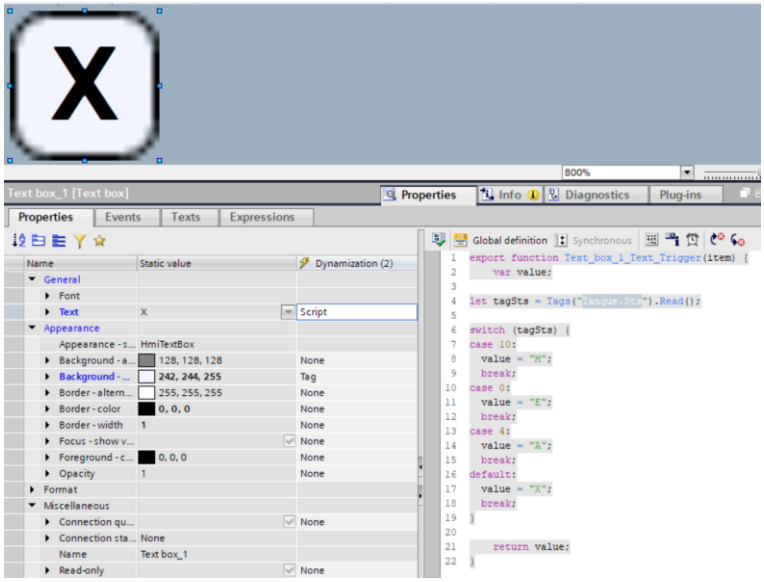


Fonte: Autoria própria.

Todas as animações deste objeto resumido utilizaram a mesma memória padrão do *faceplate* do comando do tipo de alocação, sua criação pode ser vista pela Figura 97.

O texto apresenta os modos de operação do tanque com letras únicas; “M” para manual, “E” para espera, “A” para automático e “X” para alguma falha no código do modo de operação selecionada. Como se apresenta na Figura 105, utilizando a memória “Tanque.*Sts*”.

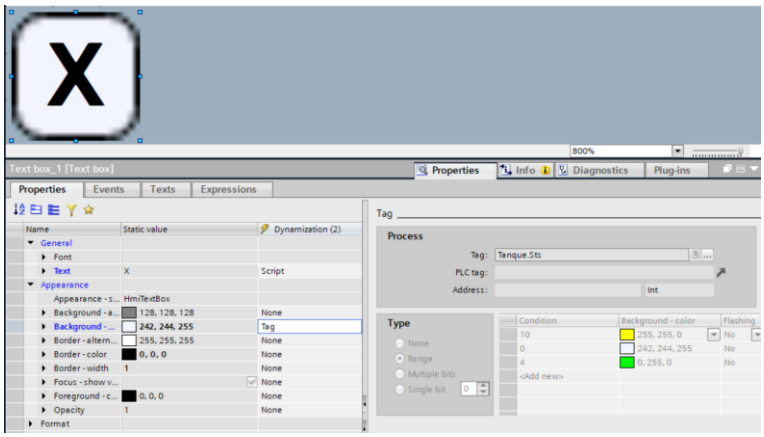
Figura 105 - Código para animação do texto resumido do modo de operação do tanque.



Fonte: Autoria própria.

Para facilitar a visualização rápido do modo de operação do tanque, foi utilizado coloração para representar os estados; amarelo para o modo manual, cinza claro para o modo de espera e verde claro para o modo automático. Como mostra a Figura 106.

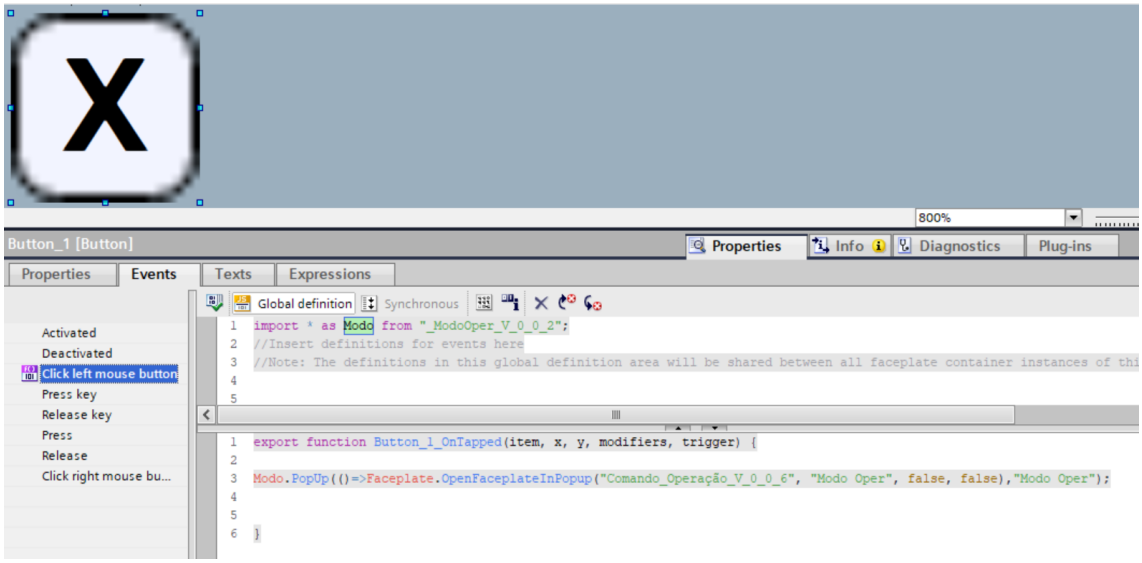
Figura 106 - Código para animação da coloração do texto resumido do modo de operação.



Fonte: Autoria própria.

Da mesma forma que o objeto do estado do tipo de alocação do tanque, este objeto também possui um botão invisível para chamar o *faceplate* (subtela) de comando de seleção do modo de operação do tanque. Como mostrada na Figura 107, se utiliza um *script* externo na área de código global do objeto para a chamada do *faceplate*; seu código utilizado se encontra no Apêndice V com parâmetro de posicionamento.

Figura 107 - Animação da chamada do *faceplate* de comando do modo de operação do tanque.



Fonte: Autoria própria.

3.8.10. Desenvolvimento do *faceplate* do modo de operação dos tanques

Para o *faceplate* de comando da seleção do modo de operação do tanque, foi utilizado um conjunto de botões de ação para cada modo, como mostrada na Figura 108.

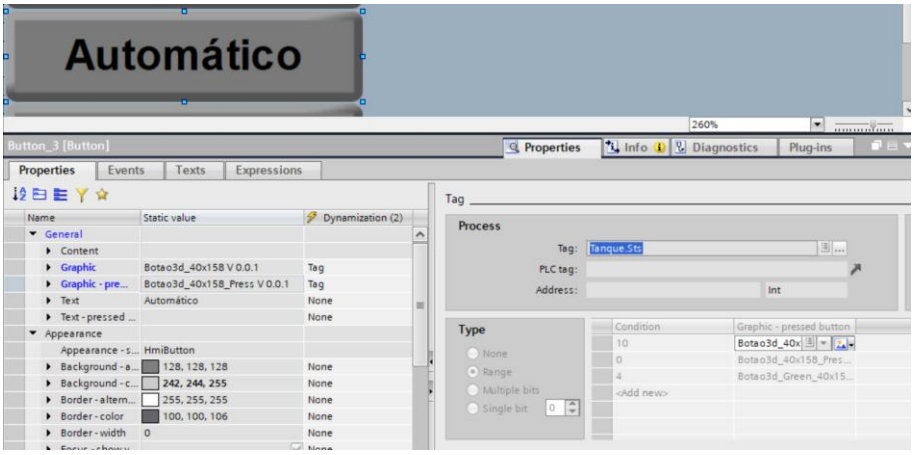
Figura 108 - *Faceplate* de comando da seleção do modo de operação dos tanques.

Fonte: Autoria própria.

Todas as animações deste *faceplate* utilizou a mesma memória padrão do objeto resumido, sua criação pode ser vista pela Figura 97.

Cada botão tem uma animação diferente, mas todas utilizam a mesma memória "Tanque.*Sts*"; e em cada botão se utilizou uma animação para coloração dos botões do estado atual do tanque. Ou seja, assim que abrir o *faceplate* de comando da operação o botão animara a coloração do seu estado atual. Seguindo o mesmo padrão já criado no objeto resumido; amarelo para o modo manual, cinza claro para o modo de espera e verde claro para o modo automático. Como se mostra na Figura 109.

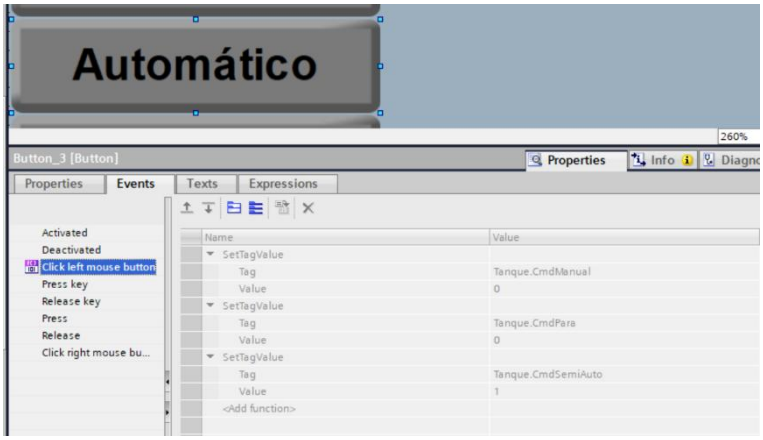
Figura 109 - Animação da coloração dos botões de seleção do modo de operação do tanque.



Fonte: Autoria própria.

Para o comando dos botões, foi utilizado as memórias "Tanque.CmdManual", "Tanque.CmdPara" e "Tanque.CmdSemiAuto" para trocar os modos de operação do tanque; como se apresenta na Figura 110.

Figura 110 - Animação do comando dos botões de seleção do modo de operação do tanque.



Fonte: Autoria própria.

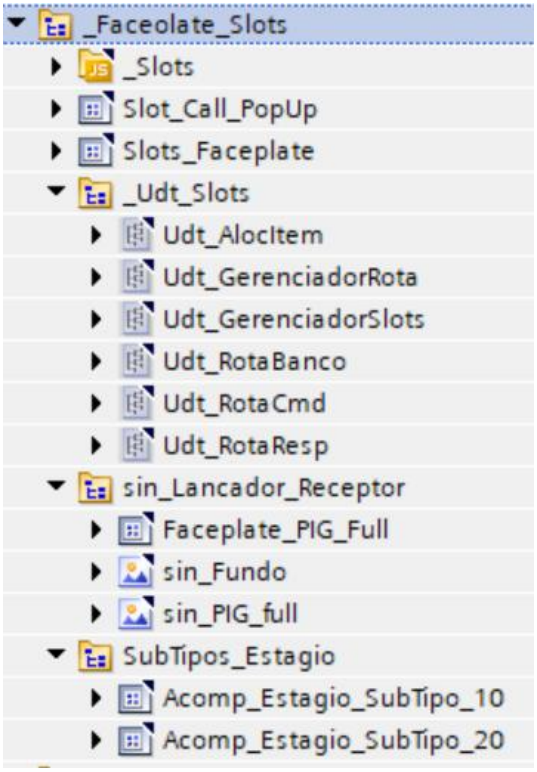
E para fechar o *faceplate* de comando do modo de operação do tanque, existe um botão de ação identificada por “FECHAR”; cujo código da animação segue o mesmo formato do *faceplate* de comando, reveja a Figura 73.

3.8.11. Desenvolvimento dos objetos de animação dos *SLOT* de operação

Neste projeto, além do desenvolvimento dos equipamentos e tanques, foi desenvolvido os *Slots* de operação. Elas servem para operar e monitorar uma receita da produção; neste projeto um conjunto de válvulas e sensores para realizar a limpeza da tubulação.

Foi criado 10 *Slots* de operação e um número considerável de receita de operação; lembrando que cada *slots* pode operar apenas uma receita por vez, de acordo com a necessidade do operador. Da mesma forma que o objeto de válvula e sensores, neste projeto foi desenvolvido os objetos dos *Slots*; e compartimentalizados em pasta dentro da biblioteca, como mostrada na Figura 111.

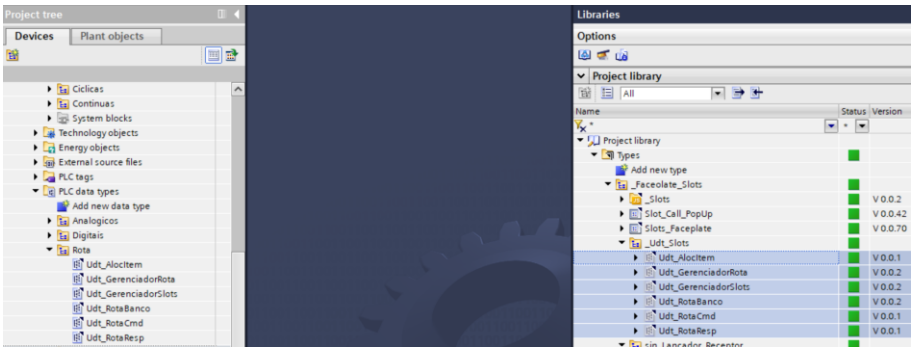
Figura 111 - Pasta da biblioteca para os estados dos tanques.



Fonte: Autoria própria.

Para adicionar o tipo de dado dos tanques, em vez de criar cada dado por si, é mais rápido arrastar o tipo de dado do tanque da pasta do PLC para dentro da biblioteca da HMI, desta forma se consegue sincronizar os dados; como mostra a Figura 96.

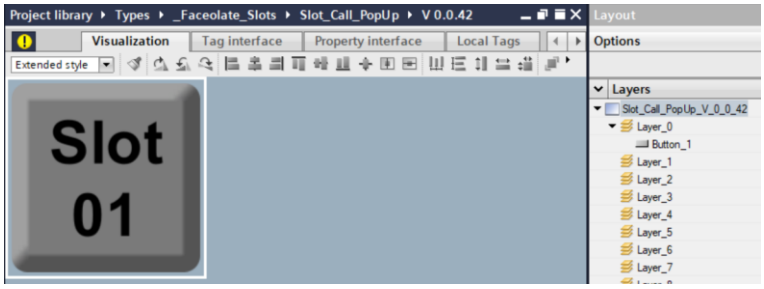
Figura 112 - Tipo de dados utilizados nas animações dos *Slots* de operação.



Fonte: Autoria própria.

O primeiro objeto desenvolvido foi um objeto resumido com a chamada do *slot* de operação específico; como mostra a Figura 113.

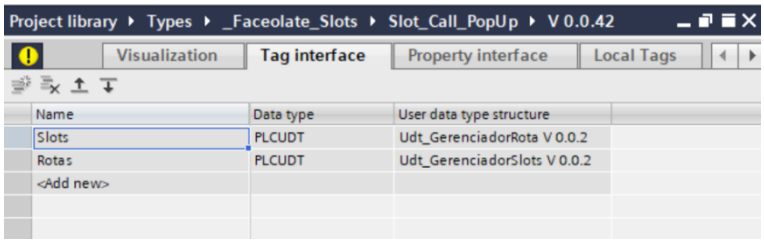
Figura 113 - Objeto utilizado na animação do *Slot* de operação.



Fonte: Autoria própria.

Após criar um objeto novo para representar o objeto resumido, toda animação desenvolvida utilizou um padrão de memória para poder trabalhar depois com o vetor de equipamento. Assim como os objetos das válvulas e sensores, foi inserida uma memória padrão para os *Slots* utilizando o tipo de dado dos *Slots* e Rotas; como mostra a Figura 114.

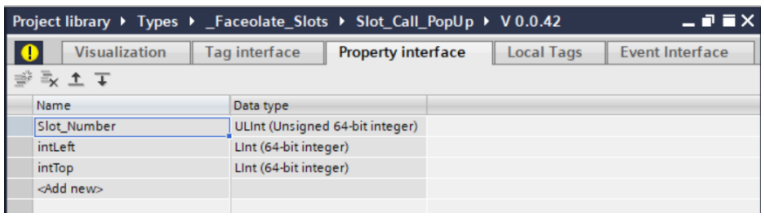
Figura 114 - Adição de memória do *Slot* e Rota geral.



Fonte: Autoria própria.

E para a animação de posição de do número *Slot* a ser chamado, foi utilizado algumas memorias locais do objeto; como apresentada na Figura 115.

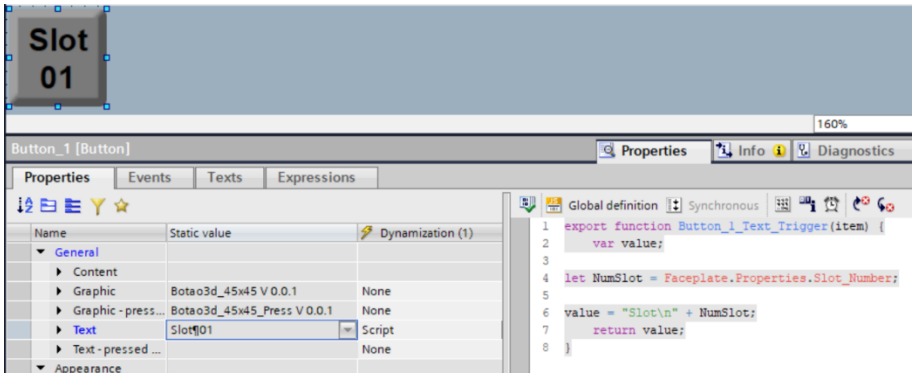
Figura 115 - Adição de memória local do *Slot* geral.



Fonte: Autoria própria.

Dentro do botão de chamada do *Slot* de operação, foi inserido uma animação de texto utilizando a memória "*Faceplate.Properties.Slot_Number*" da aba de *"Property Interface"* como mostrada na Figura 116.

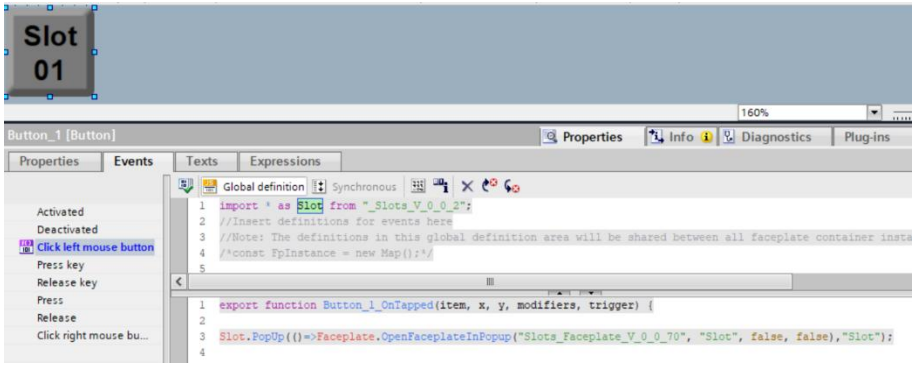
Figura 116 - Animação do texto do botão de chamada do *Slot* de operação.



Fonte: Autoria própria.

Para a animação de chamada do *faceplate* de comando, foi utilizado o código de identificação da Figura 117 e o Apêndice V para o código de chamada com parâmetros de posicionamento.

Figura 117 - Código de chamada do *Slot* de operação.



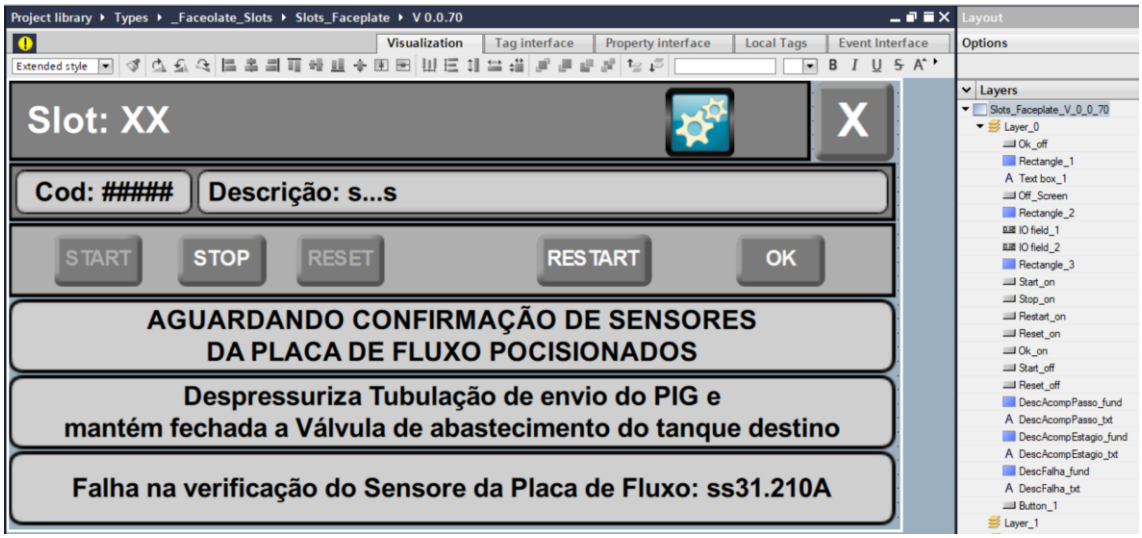
Fonte: Autoria própria.

3.8.12. Desenvolvimento do *faceplate* de comando do *Slot* de operação

Para o desenvolvimento do *faceplate* de comando dos *slots* foi utilizado o mesmo modelo de memórias padrão e memórias locais do objeto resumido do *Slot*; reveja a Figura 114 e Figura 115.

Em seguida utilizou os objetos de animação apresentados na Figura 118 para o desenvolvimento do *faceplate* de comando dos *Slots* de operação.

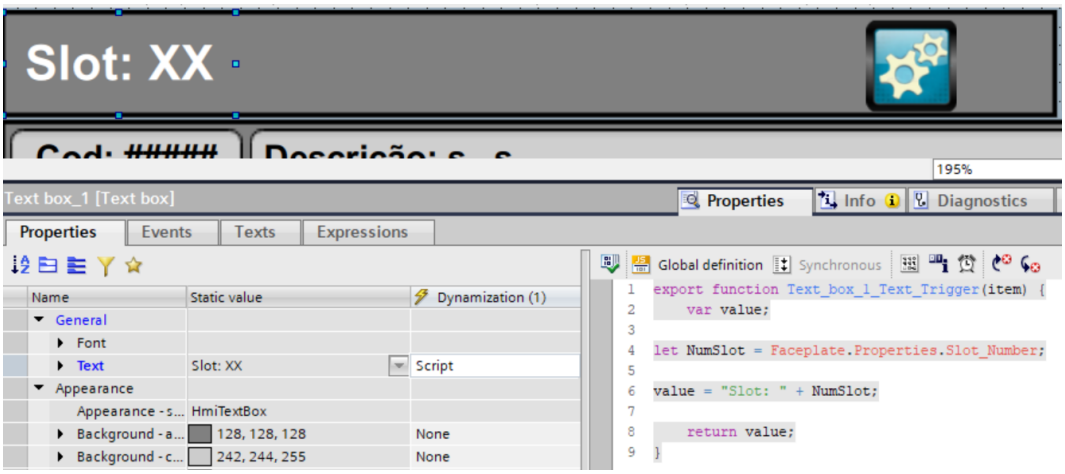
Figura 118 - Objeto utilizados no *faceplate* de comando dos *Slots* de operação.



Fonte: Autoria própria.

Para identificar o número do slot aberto no faceplate de comando, utilizou a animação do texto da Figura 119, usando a memória local "*Faceplate.Properties.Slot_Number*".

Figura 119 - Animação do texto de identificação do *Slot* de operação no *faceplate*.

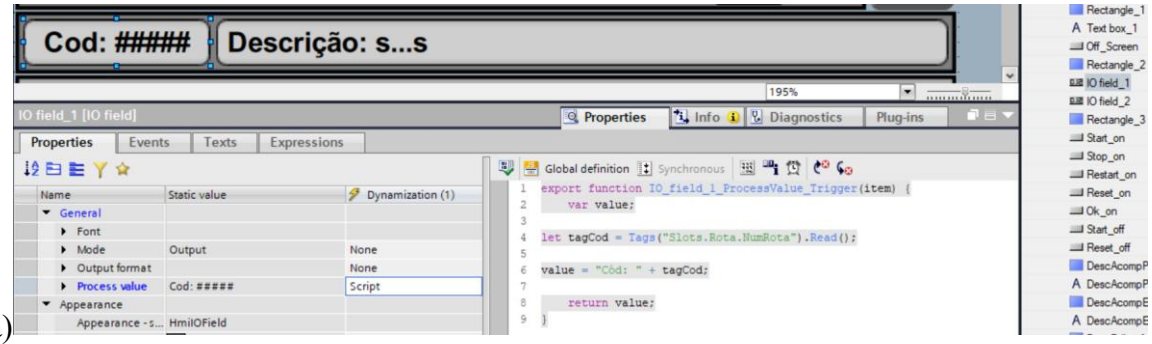


Fonte: Autoria própria.

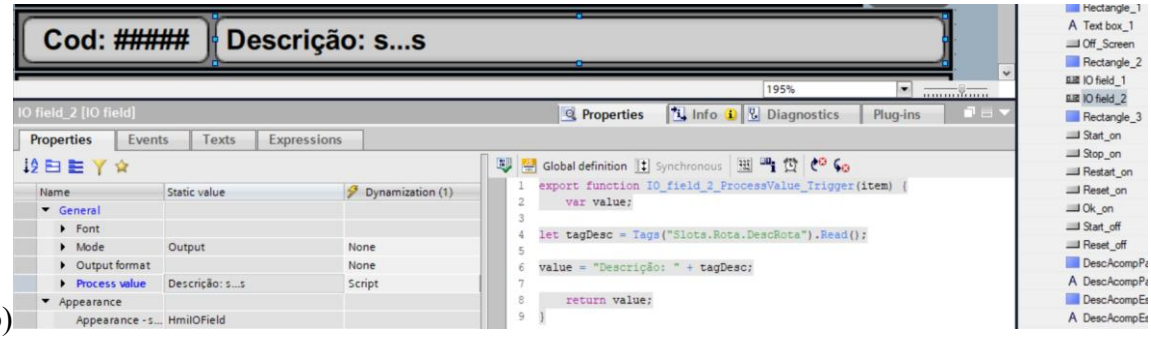
A fim de organizar e identificar as receitas de operando dentro dos *Slots* foi inserido objetos de leitura de memórias do “Slots.Rota.NumRota” e do “Slots.Rota.DescRota” para identificação visual do operador; ambas apresentadas na Figura 120.

Figura 120 - Animação do texto de identificação do a) Código e b) Descrição da receita operando no *Slot* de operação do *faceplate*.

a)



b)

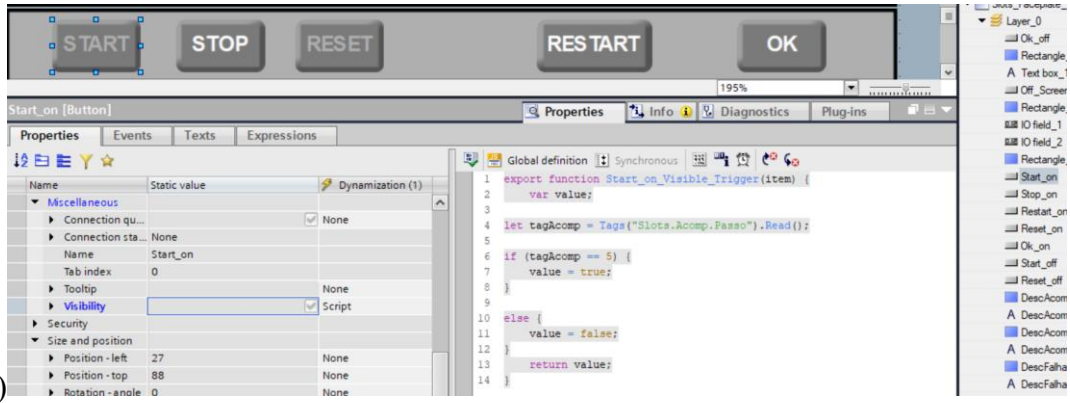


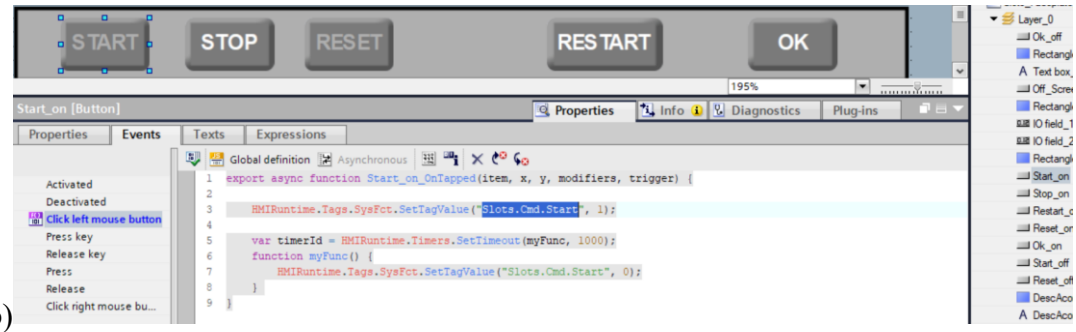
Fonte: Autoria própria.

Para iniciar a receita, é necessário o pressionar do botão “*START”*; esse botão só aparece para ser pressionado apenas no início da receita, após isso esse botão é desativado para evitar algum tipo de falha durante a operação da receita. Memórias utilizadas para as animações; “*Slots.Acomp.Passo*” e “*Slots.Cmd.Start*”. Como mostrado na Figura 121.

Figura 121 - Animação da a) Invisibilidade do botão *“START”* e do b) Comando

a)

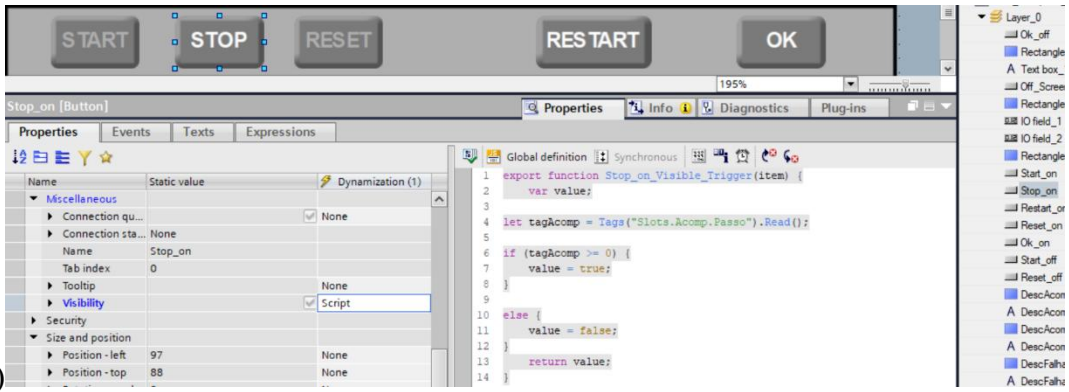


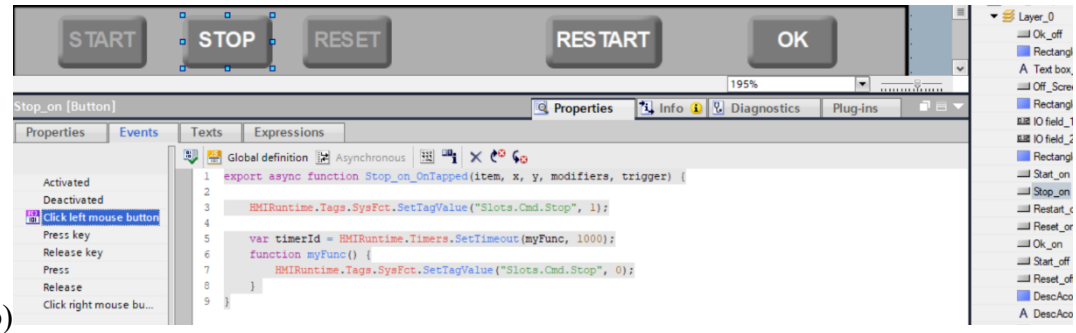
b) 

Fonte: Autoria própria.

E uma vez que a receita se inicia, é possível pausar a receita em todos os passos; basta o operador pressionar o botão "*STOP*". Esse botão tem uma animação de invisibilidade e comando que aparecem sempre que uma receita tiver sido iniciada. Memórias utilizadas para as animações; "*Slots.Acomp.Passo*" e "*Slots.Cmd.Stop*", como mostra a Figura 122.

Figura 122 - Animação da a) Invisibilidade do botão *"START"* e do b) Comando

a) 

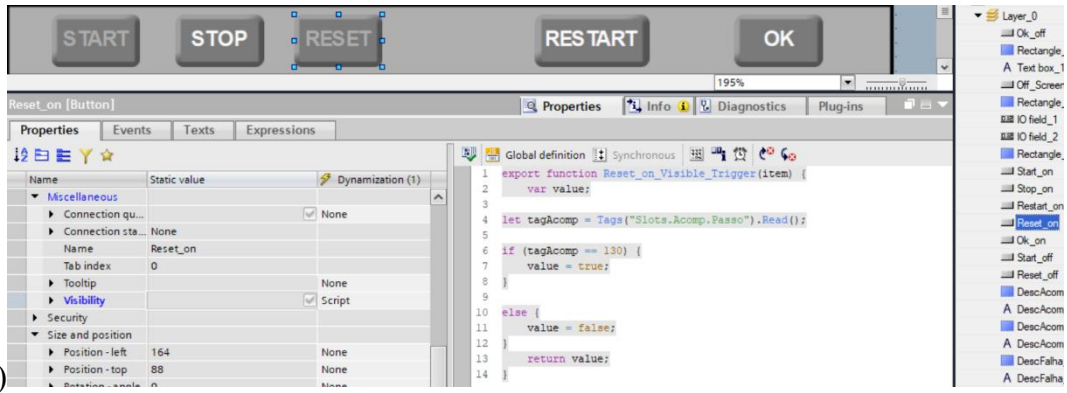
b) 

Fonte: Autoria própria.

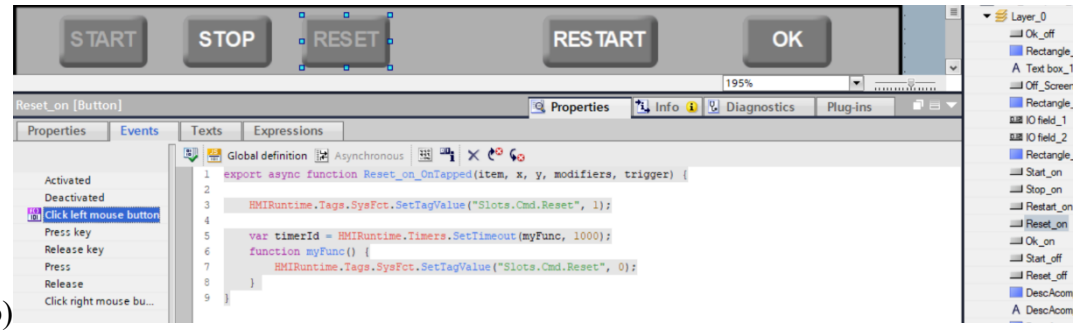
O botão de *"RESET"* só pode ser pressionado quando a operação da receita estiver em pausa ou finalizada, ela tem a função de limpar o *Slot* de operação para a próxima receita ser inserida. Este botão tem uma animação de invisibilidade e comando utilizando as memórias "*Slots.Acomp.Passo*" e "*Slots.Cmd.Reset*", como apresentada na Figura 123.

Figura 123 - Animação da a) Invisibilidade do botão *"RESET"* e do b) Comando

a)



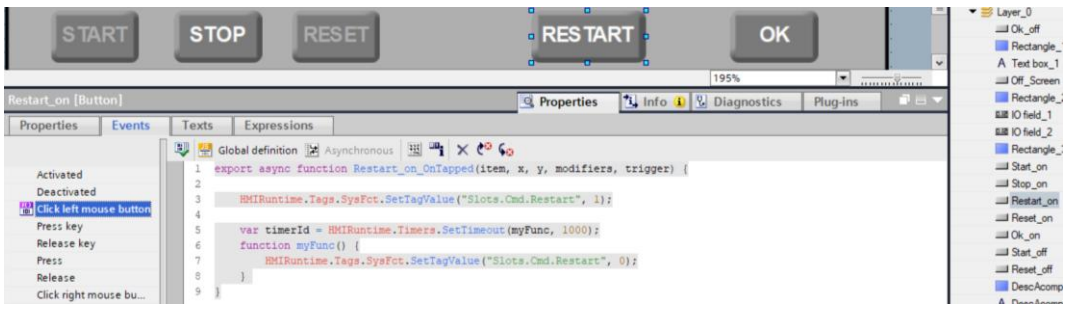
b)



Fonte: Autoria própria.

O botão de *"RESTART"* tem a função de restaurar o passo da receita desde o momento que ela foi pausada, o comando só é aceito quando a receita estiver em pausa ou finalizada. Este botão tem uma animação de comando utilizando a memória "*Slots.Cmd.Restart*", como apresentada na Figura 124.

Figura 124 - Animação do comando do botão *"RESTART"*.

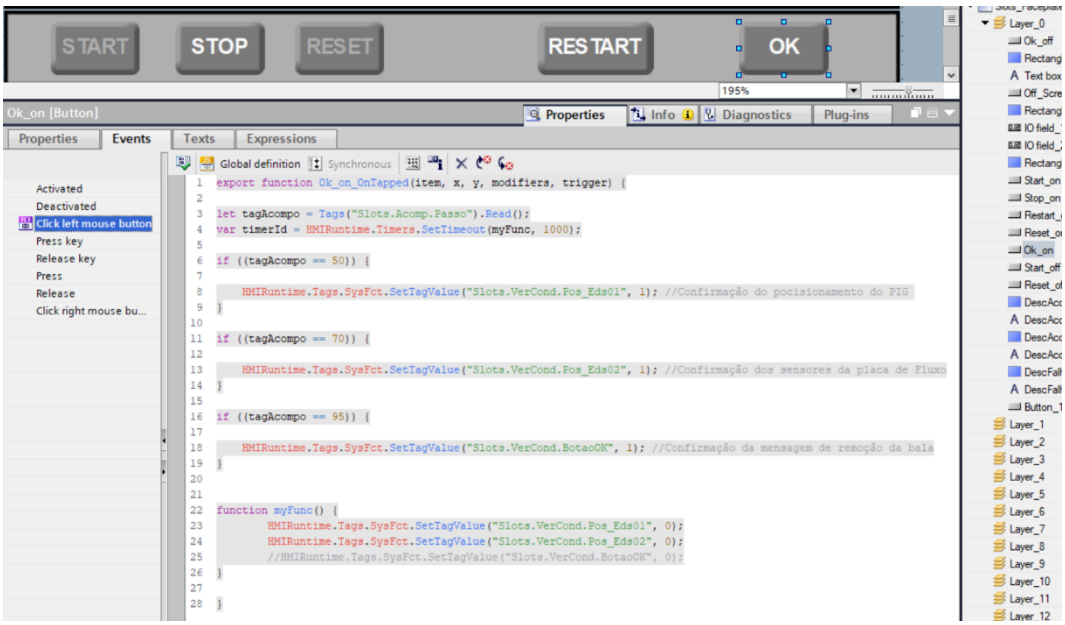


Fonte: Autoria própria.

O botão *"OK"* deve ser pressionado sempre que uma ação do operador for concluída nos passos da receita, por exemplo, a ação de preparar a pastilha *PIG* na posição de lançamento, ou a ação de posicionar as conexões das tubulações para a limpeza ou a remoção da pastilha do *PIG* no passo final da receita de limpeza. Este botão tem uma animação de comando utilizando

as memórias "*Slots.Acomp.Passo*", "*Slots.VerCond.Pos_Eds01*", "*Slots.VerCond.Pos_Eds02*" e "*Slots.VerCond.BotaoOK*", como apresentada na Figura 125.

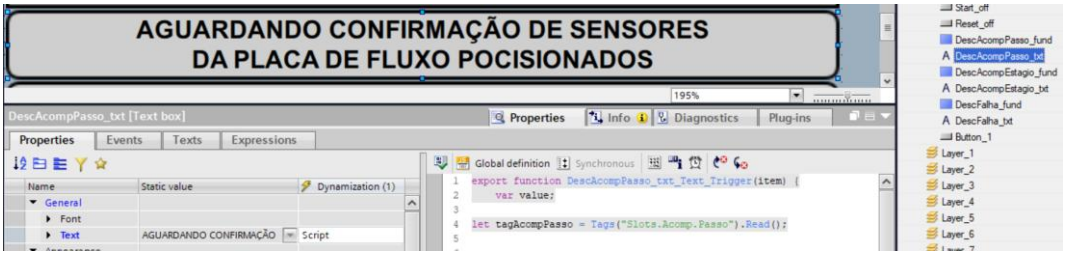
Figura 125 - Animação do comando do botão *"OK"*.



Fonte: Autoria própria.

Para acompanhar os passos da receita, foi criado uma animação de texto que se altera de acordo com os passos da receita, o código completa da animação se encontra no Apêndice VI. Utilizando a memória *"Slots.Acomp.Passo",* foi implementada a animação da Figura 126.

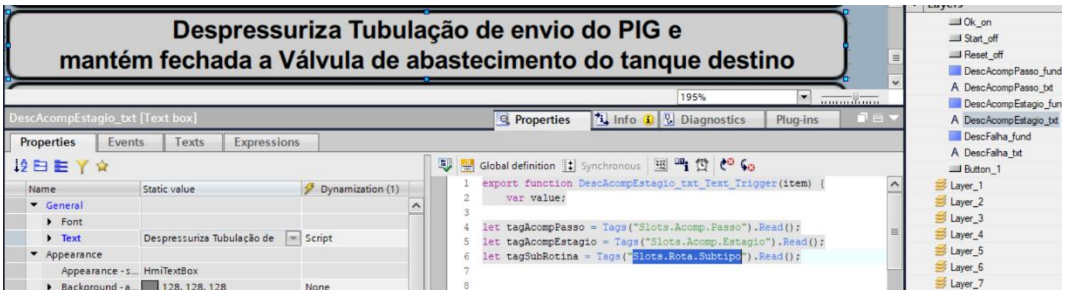
Figura 126 - Animação de texto para mensagem dos passos da receita.



Fonte: Autoria própria.

Para acompanhar os estágios de ação do *PIG*, foi criado outra animação de texto que se altera de acordo com as memórias *"Slots.Acomp.Passo"*, *"Slots.Acomp.Estagio"* e *"Slots.Rota.Subtipo"*; como apresentado na Figura 127. O texto de estágio do *PIG* muda de acordo com o tipo de receita, pode ser dois tipos; transferências de tubulações entre tanque ou transferência de tubulações entre tanque e envasadora. O código completo da animação se encontra no Apêndice VII.

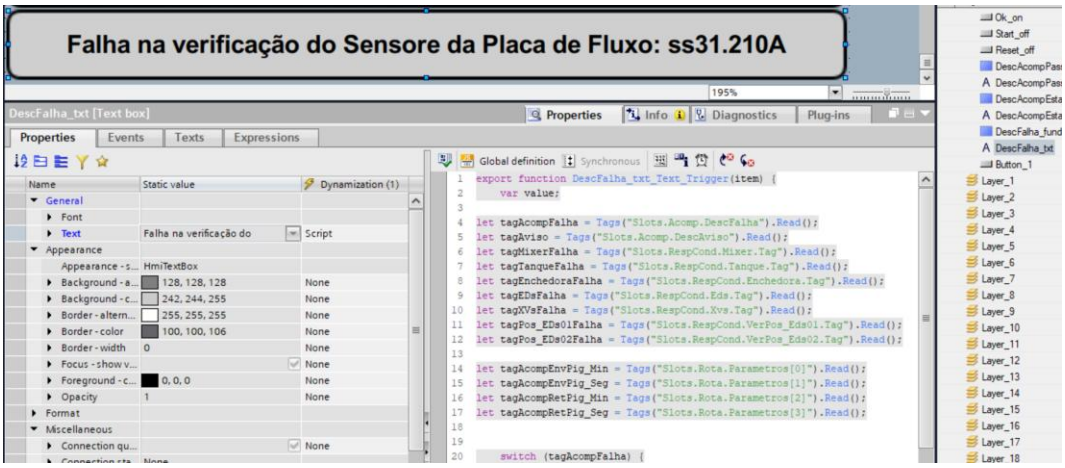
Figura 127 - Animação de texto para mensagem dos passos do estágio do *PIG*.



Fonte: Autoria própria.

Agora também há bloco de mensagens para as falhas que podem ocorrer durante a receita, o que leva muitas memórias do *Slot*, como mostra a Figura 128. A mensagem de falha procura identificar o tipo de falha e o nome o equipamento em ocorreu a falha. O código completo da lógica da animação se encontra no Apêndice VIII.

Figura 128 - Animação de texto para mensagem de falha do *faceplate* de operação.

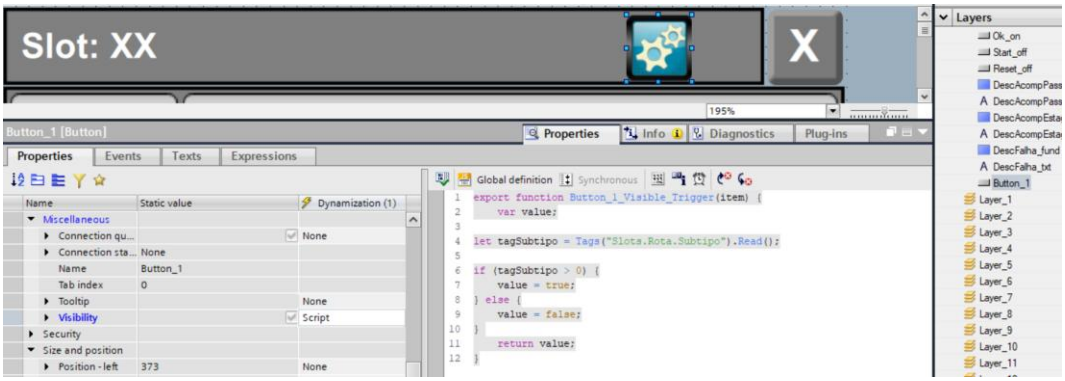


Fonte: Autoria própria.

Para melhor visualização dos passos da receita, foi desenvolvida um *faceplate* para visualizar todos os passos da receita no formato de fluxograma. Esse botão de chamada só aparece após o início da receita em operação, com a animação utilizando a memória "*Slots.Rota.Subtipo*", como mostra a Figura 129.

Foi desenvolvido dois faceplates de fluxograma de receita, pois existem dois padrões de receita de operação, receita de limpeza nas tubulações entre tanques e limpeza entre tanque e envasadora. Seu código completo se encontra no Apêndice IX.

Figura 129 - Animação de visibilidade do botão de chamada do fluxograma dos passos da receita do *faceplate*.



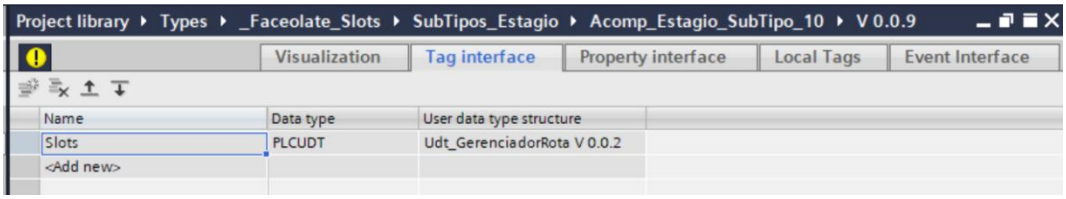
Fonte: Autoria própria.

E para fechar o *faceplate* de comando do *Slot*, existe um botão de ação identificada por "X"; cujo código da animação segue o mesmo formato do *faceplate* de comando, reveja a Figura 73.

3.8.13. Desenvolvimento do *faceplate* do fluxograma dos passos da receita em operação

Antes de começar a desenvolver as animações dos objetos do *faceplate*, é necessário inserir a memória padrão do *Slot*, como demonstrada na Figura 130.

Figura 130 - Objeto utilizado na animação do fluxograma dos passos da receita.

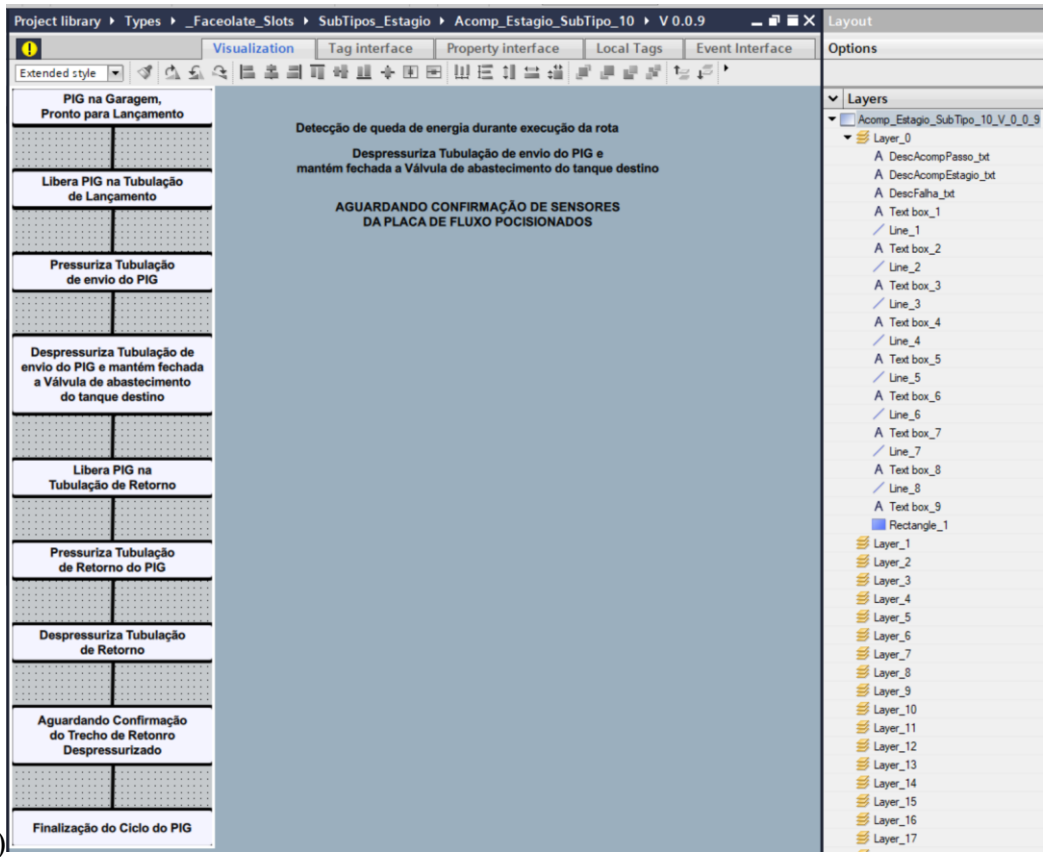


Fonte: Autoria própria.

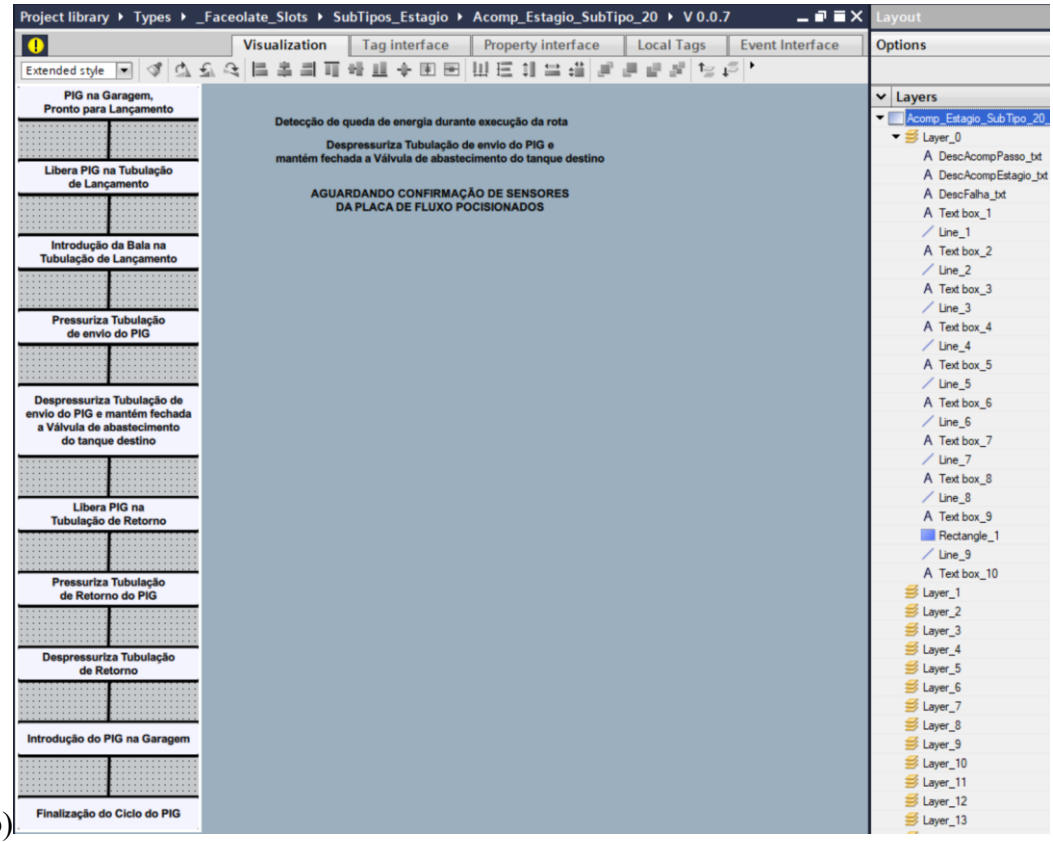
Como foi dito antes, existem dois tipos de padrão de receia no projeto, uma delas tem o padrão de sequência de equipamentos entre tanques, e a outra é o padrão de sequência entre os tanques e as envasadoras. Todas as receitas podem ser identificadas o tipo de padrão através da memória *"Slots.Rota.Subtipo"*. Os tipos de fluxograma disponível no projeto podem ser visualizados pela Figura 131.

Figura 131 - Objetos do Fluxograma do a) Subtipo 10 e do b) Subtipo 20.

a)



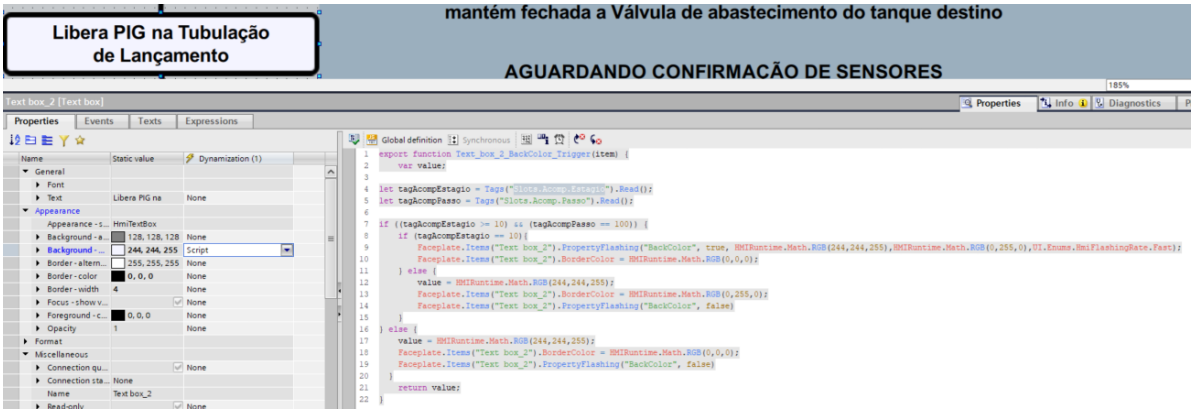
b)



Fonte: Autoria própria.

Para a animação dos blocos do fluxograma se utilizou o código mostrada na Figura 132; onde procura apresentar duas situações de animação, quando o passo está em na posição do bloco do fluxograma, suas bordas ficam piscando entre verde claro e branco; e quando o passo da receita está para a frente do bloco do fluxograma, suas bordas se mantem verde claro, demonstrando que este passo da receita já foi realizado. O texto de cada bloco do fluxograma foi inserido manualmente, pois ela não se alterará de acordo com o passo da receita, apenas a sua animação das bordas.

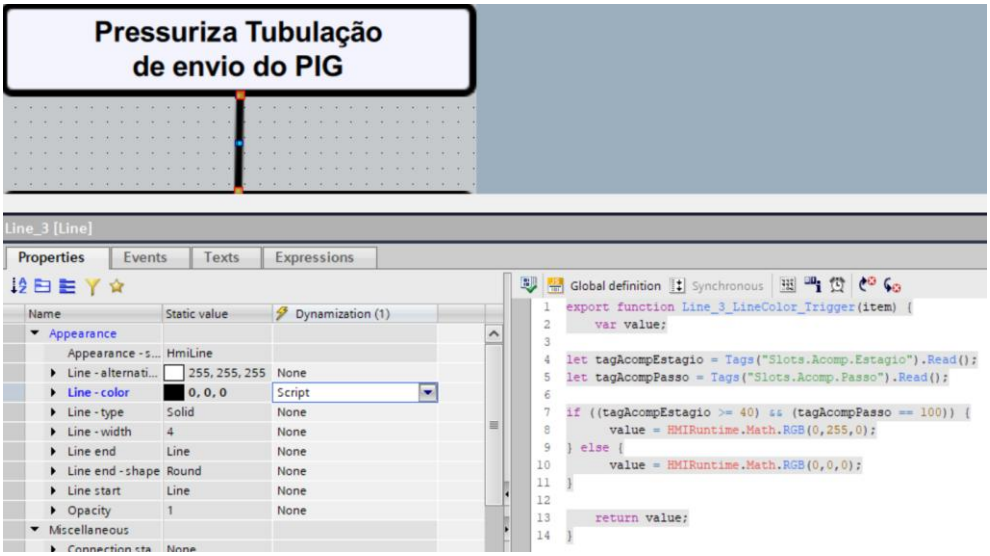
Figura 132 - Código da animação dos blocos do *faceplate* do fluxograma.



Fonte: Autoria própria.

E as linhas de conexão entre os blocos do fluxograma, também tem animação de coloração. Como este é apenas uma linha, não tem a necessidade da animação piscando, apenas da coloração verde quando o passo da receita já passou. Como mostra a Figura 133.

Figura 133 - Código da animação das linhas de conexão do *faceplate* do fluxograma.



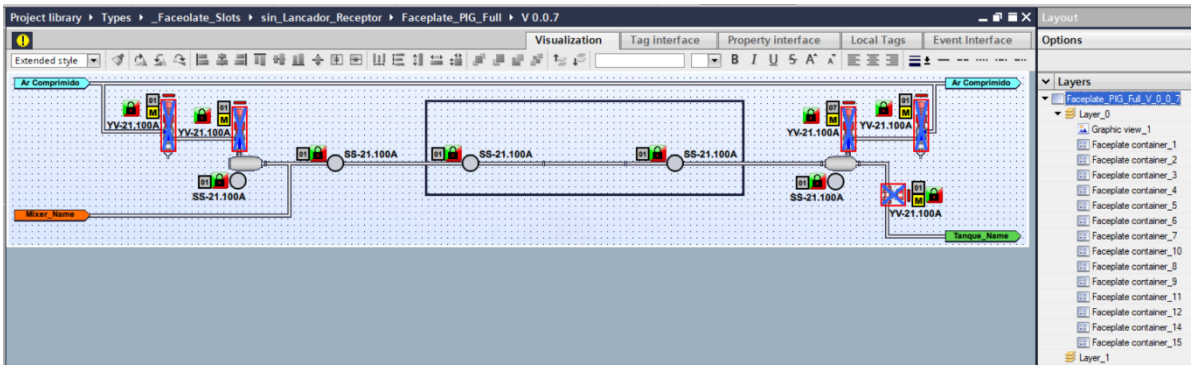
Fonte: Autoria própria.

3.8.14. Desenvolvimento do *faceplate* da linha de operação do *PIG* resumida

Como a receita utiliza vários equipamentos entre válvulas, tanques e sensores; por conta do tamanho da tela da HMI, não se pode colocar todos os equipamentos de ação da receita em uma tela. Então para acompanhar visualmente todos os passos da receita da limpeza, era necessário ficar alternando as telas de comando.

Para evitar isso, foi desenvolvido um faceplate de visualização de todos os equipamentos da receita numa tela pequena. Apenas para fins de visualização; como mostra a Figura 134

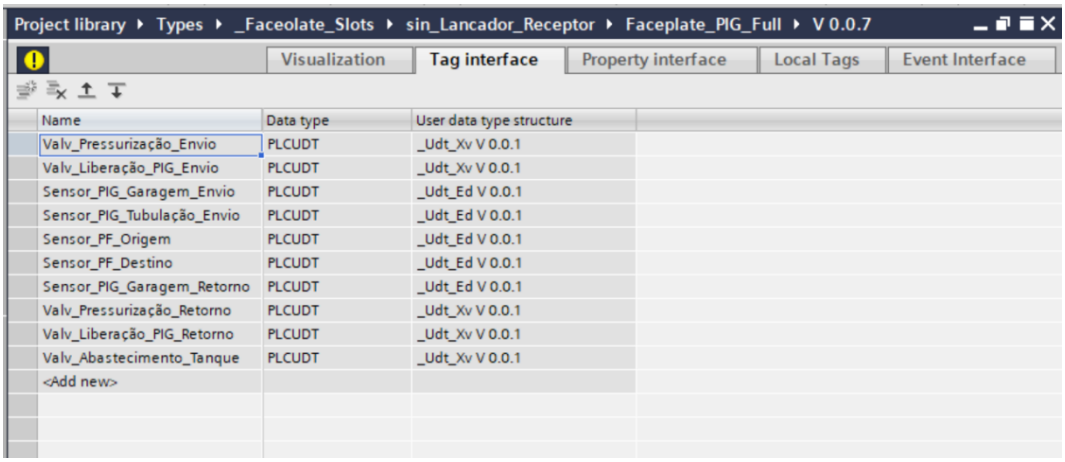
Figura 134 - Objeto utilizado na animação do *faceplate* da operação total.



Fonte: Autoria própria.

Como ela possui todos os equipamentos já desenvolvidos até então, na aba de *"Tag Interface"* devem ser incluídas todas as memórias padrão dos equipamentos; como apresenta a Figura 135.

Figura 135 - Adição das memórias dos equipamentos utilizados no *faceplate*.

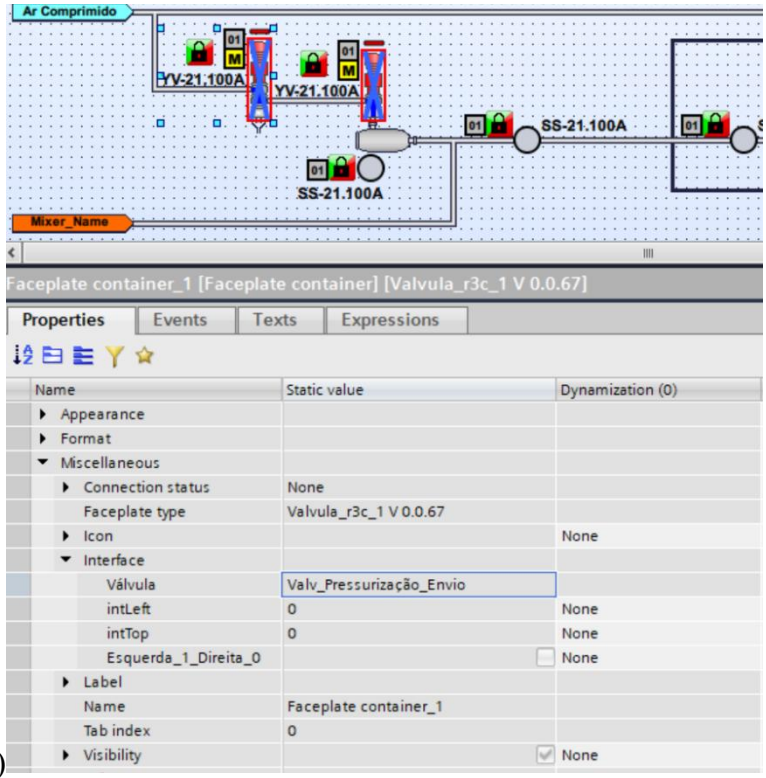


Fonte: Autoria própria.

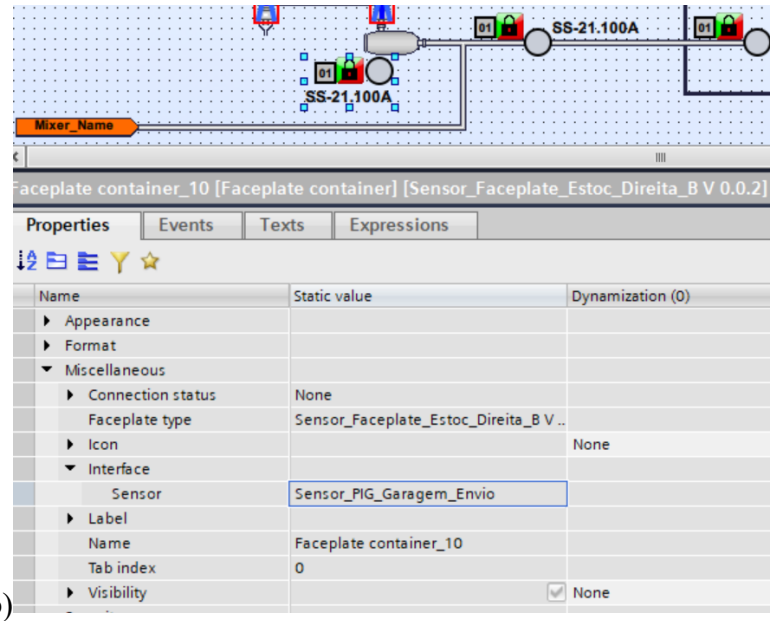
Após inserido os objetos de válvulas e sensores, é necessário inserir os dados de interface de onde o objeto irá recolher todas as informações para a animações, será explicado com mais detalhe mais afrente, porém neste caso, será inserido as memórias padrão criadas na Figura 136 para criar as conexões com os equipamentos reais.

Figura 136 - Adição dos parâmetros da a) Válvula e do b) Sensor.

a)



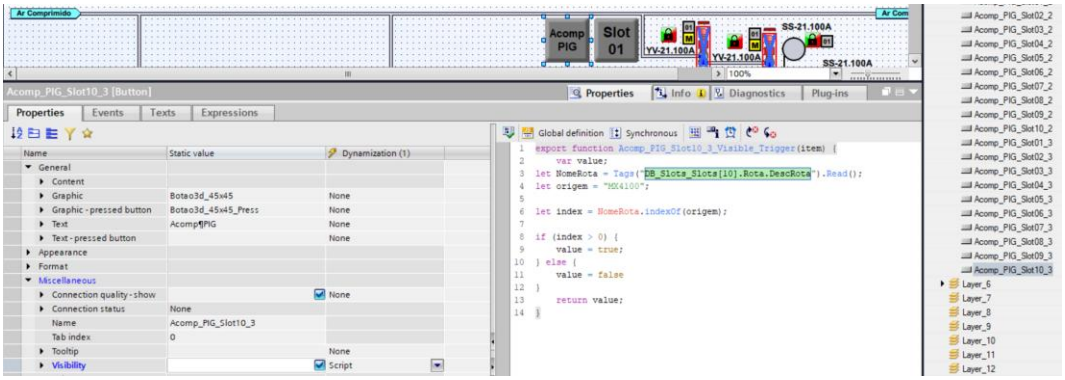
b)



Fonte: Autoria própria.

O botão de chamada deste *faceplate*, se desenvolveu direto na tela principal de comando da HMI. Utilizando a descrição da receita dos *slots* para a animação de visibilidade do botão de chamada, vinculando com a busca do nome do tanque; como mostra a Figura 137, foi utilizado a busca do nome *"*MX4100*"* no *slot* número 10. Nas telas principais de comando, foi desenvolvido botões de chamada de busca de nome para todos os tanques, para todos os 10 tipos de *slots* de operação.

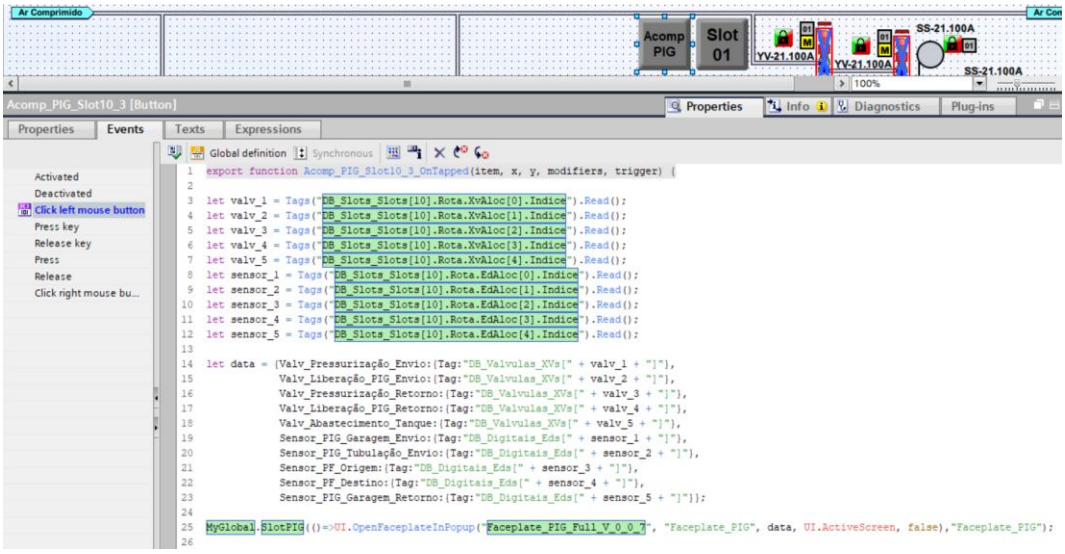
Figura 137 – Animação de visibilidade dos botões de chamada do *faceplate* de operação total da receita.



Fonte: Autoria própria.

Uma vez acionado a animação de visibilidade do botão de chamada, sua ação de chamada se dá pelo código da Figura 138; onde ocorre a vinculação das válvulas e sensores da receita com as memorias padrão do *faceplate* da operação total da receita.

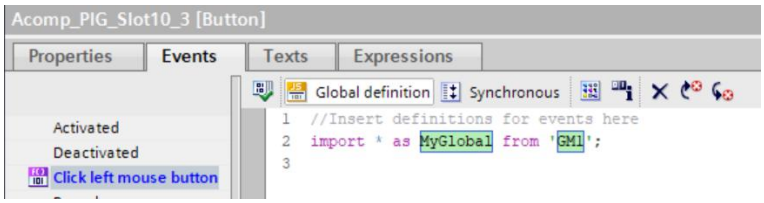
Figura 138 - Animação de vinculação das memórias padrão da chamada do *faceplate* de operação total da receita.



Fonte: Autoria própria.

O código de chamada do *faceplate* ocorre em duas etapas; desta vez utilizando os recursos dos *Scripts* da pasta *GM1*. Primeiro se deve inserir na aba de *"Global definition"* o código de uso dos *scripts* da pasta *GM1*; como mostra a Figura 139.

Figura 139 - Código para importar os *scripts* da pasta do *GM1*.



Fonte: Autoria própria.

E o código desenvolvido para criar o faceplate simples da operação total da receita se dá pela Figura 140.

Figura 140 - Código de criação do *faceplate* da operação completa da receita.

Nutrien_Processo_V18 ▸ HMI_Nutrien_Processo [MTP1000 Unified Comfort] ▸ Scripts ▸ GM1 ▸ SlotPIG

Synchronous

```
1  export function SlotPIG(OpenPopUpCall, Objectname) {
2
3  let ui = UI;
4  let count = ui.PopupScreenWindows.Count;
5  let fsExisits = false;
6
7      for (let i=0; i<count; i++){
8          let name = ui.PopupScreenWindows(i).Name;
9
10         if (PopUpSlot.get (name) == Objectname){
11               PopUpSlot.delete(name);
12               ui.PopupScreenWindows(i).Close();
13               fsExisits = true;
14               break;
15         }
16     }
17
18     if (!fsExisits){
19
20         let po = OpenPopUpCall();
21         po.Left = 0;
22         po.Top = 475;
23         //po.Left = Faceplate.Parent.Left + (Faceplate.Properties.intTop);
24         //po.Top = Faceplate.Parent.Top + (Faceplate.Properties.intTop);
25         //po.Visible = true;
26         PopUpSlot.set(po.Name, Objectname);
27     }
28
29
30 }
```

Fonte: Autoria própria.

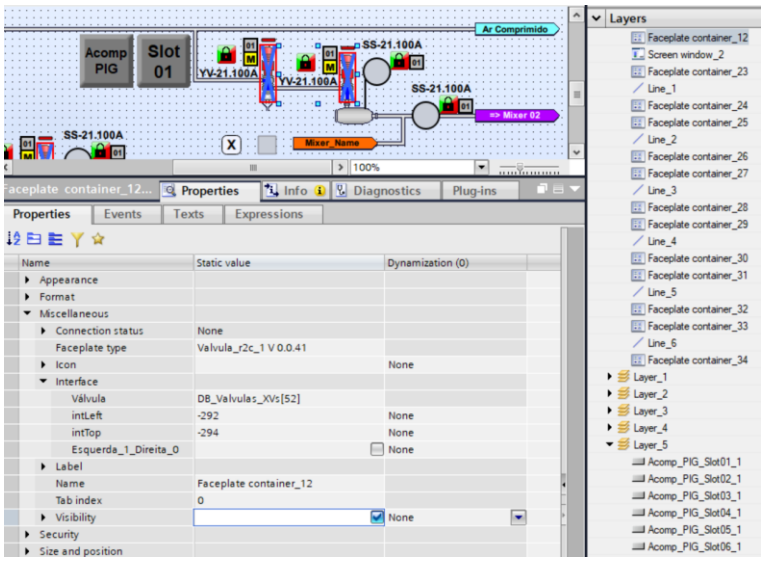
3.9. Configuração de objetos de animação nas telas principais de comando da HMI

Para inserir um objeto de válvula, sensor ou outro nas telas principais de comando da HMI, basta arrastar da biblioteca de objetos do projeto para dentro da tela. Após isso é necessário vincular esses objetos com as memórias do PLC.

3.9.1. Configuração dos parâmetros das válvulas

Dentro das opções de propriedade do objeto da válvula, há uma aba dentro *"Miscellaneous / Interface"* em que se encontra o endereço de vinculação e parâmetros de posição de *faceplate* da válvula; como mostra a Figura 141.

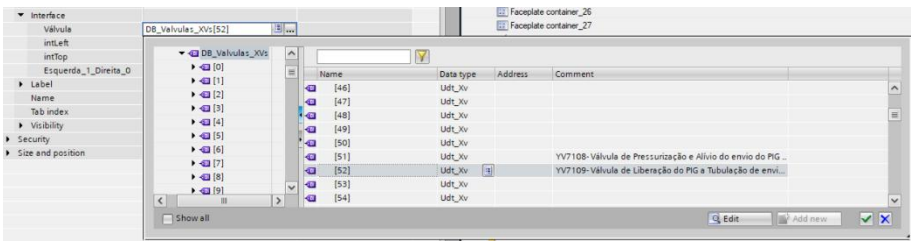
Figura 141 - Localização da aba *"Interface"* do objeto da válvula.



Fonte: Autoria própria.

Após isso, na opção de "Válvula" (nome da memória padrão do objeto), encontra se uma seleção das memórias, que se encontram dentro da pasta *"*HMI *tags"*, para selecionar a válvula do programa do PLC; como demonstrada na Figura 142.

Figura 142 - Vinculação da válvula com a memória padrão do objeto.



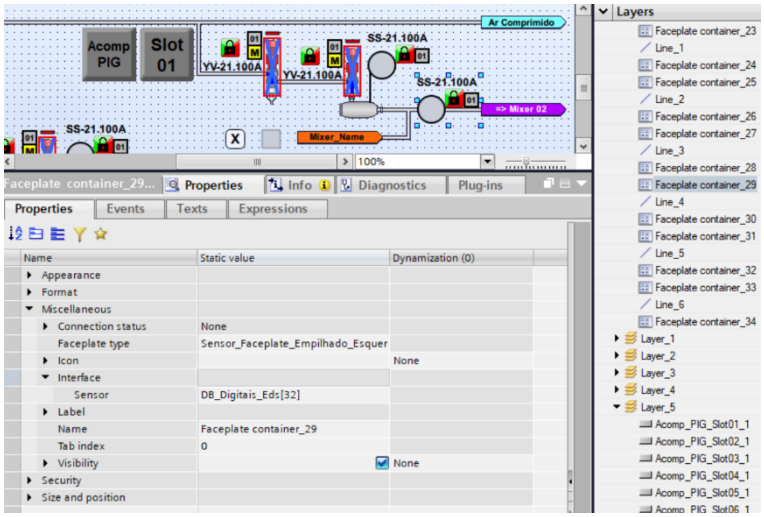
Fonte: Autoria própria.

Sendo quantificado em pixel a distância entre os objetos de animação da tela da HMI. Foi utilizado nessa medida de contagem para inserir o valor dos parâmetros *"intLeft"* e *"intTop"* para posicionar a abertura do *faceplate* do equipamento mais ao centro possível da tela da HMI.

3.9.2. Configuração dos parâmetros dos sensores digitais

Dentro das opções de propriedade do objeto do sensor, há uma aba dentro *"Miscellaneous / Interface"* se encontra o endereço de vinculação do *faceplate* do sensor; como mostra a Figura 143.

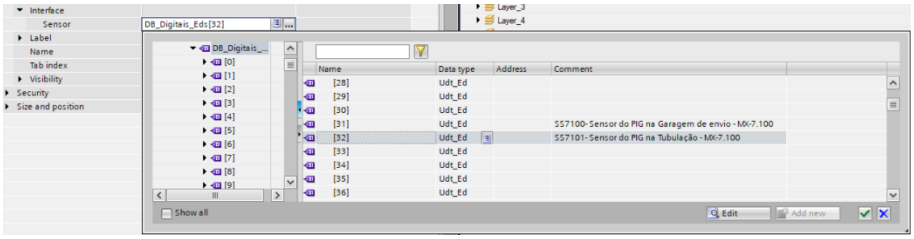
Figura 143 - Localização da aba *"Interface"* do objeto do sensor digital.



Fonte: Autoria própria.

Após isso, na opção de "Sensor" (nome da memória padrão do objeto), encontra se uma seleção das memórias, que se encontram dentro da pasta *"HMI tags"*, para selecionar o sensor digital do programa do PLC; como demonstrada na Figura 144.

Figura 144 - Vinculação do sensor com a memória padrão do objeto.

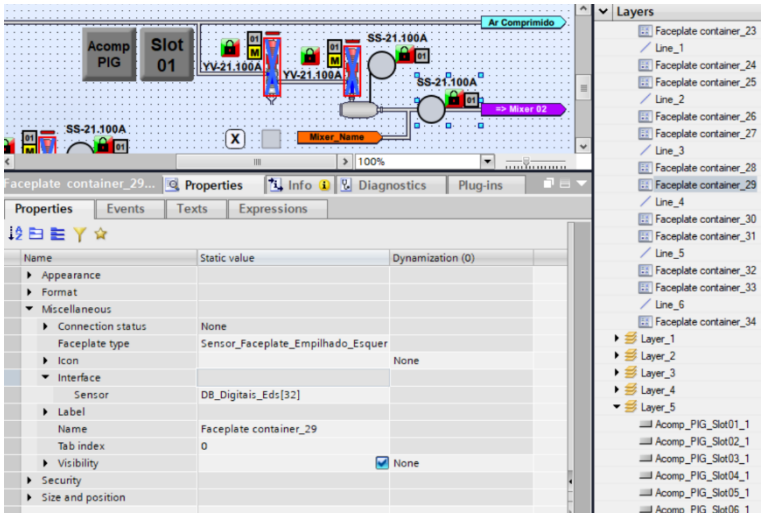


Fonte: Autoria própria.

3.9.3. Configuração dos parâmetros dos objetos dos tanques

Dentro das opções de propriedade do objeto do tanque, há uma aba dentro *"Miscellaneous / Interface"* em que se encontra o endereço de vinculação e parâmetros de posição de *faceplate* do objeto; como mostra a Figura 145.

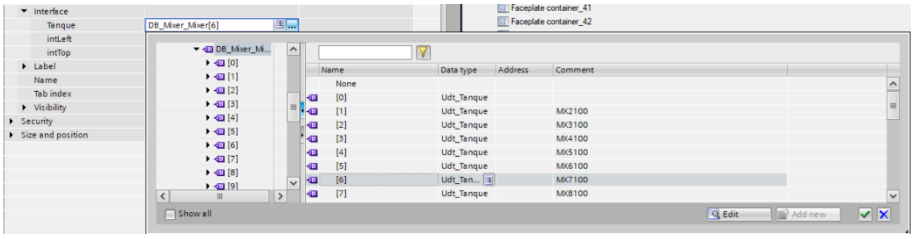
Figura 145 - Localização da aba *"Interface"* dos objetos do tanque.



Fonte: Autoria própria.

Após isso, na opção de "Tanque" (nome da memória padrão do objeto), encontra se uma seleção das memórias, que se encontram dentro da pasta *"*HMI *tags"*, para selecionar o tanque do programa do PLC; como demonstrada na Figura 146.

Figura 146 - Vinculação do tanque com a memória padrão do objeto.



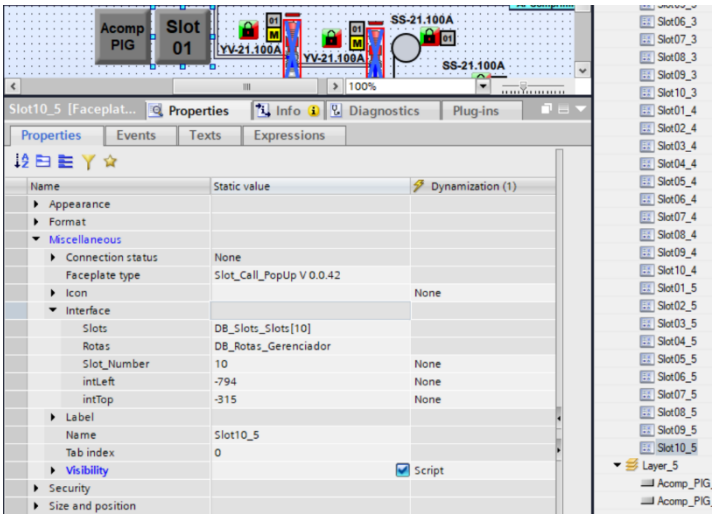
Fonte: Autoria própria.

Foi inserido o valor dos parâmetros *"intLeft"* e *"intTop"* para posicionar a abertura do *faceplate* do equipamento mais ao centro possível da tela da HMI.

3.9.4. Configuração dos parâmetros dos objetos dos *Slots* de operação

Dentro das opções de propriedade do objeto do *Slot* de operação, há uma aba dentro *"Miscellaneous / Interface"* em que se encontra o endereço de vinculação e parâmetros de posição de *faceplate* do objeto; como mostra a Figura 147.

Figura 147 - Localização da aba *"Interface"* do objeto do *Slot* de operação.

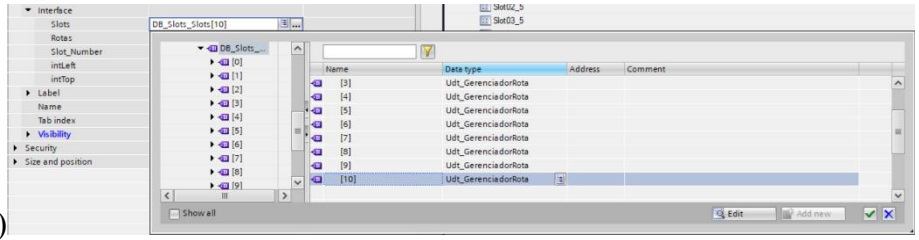


Fonte: Autoria própria.

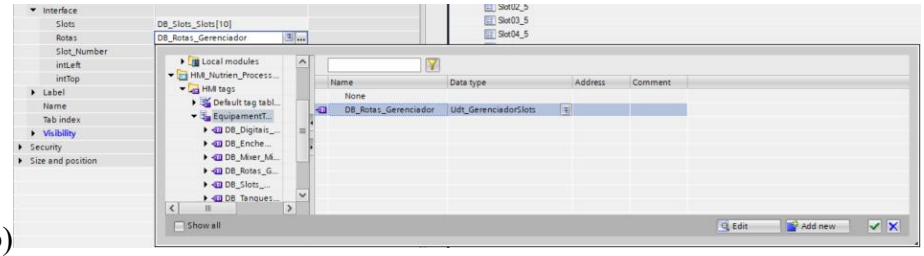
Após isso, na opção de *"Slots"* e "Rota" (nomes da memória padrão do objeto), encontra se uma seleção das memórias, que se encontram dentro da pasta *"*HMI *tags"*, para selecionar o *Slot* e gerenciador de *slot* do programa do PLC; como demonstrada na Figura 148.

Figura 148 - Vinculação do a) Slot e da b) Rota com a memória padrão do objeto.

a)



b)



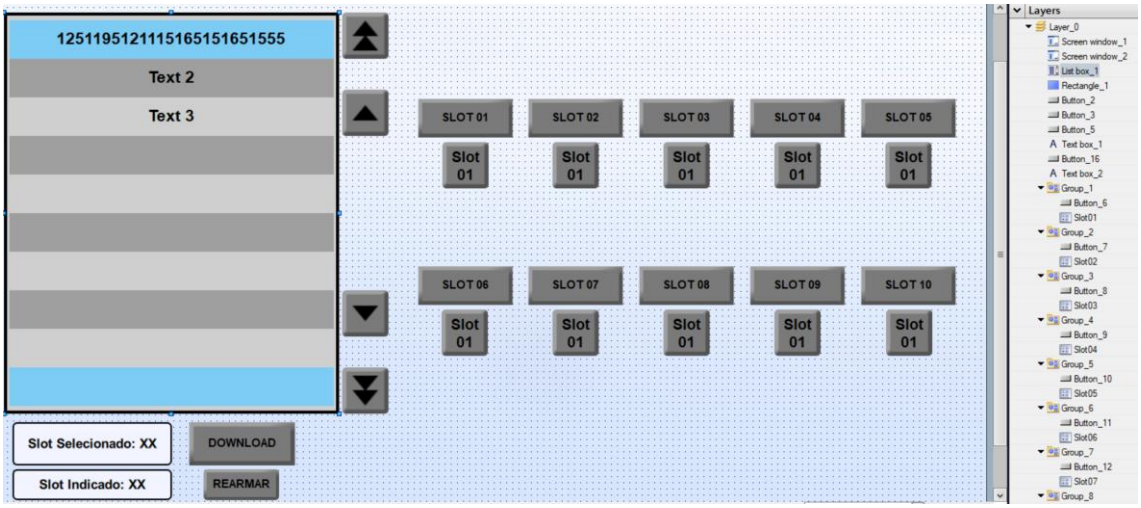
Fonte: Autoria própria.

Foi inserido o valor dos parâmetros *"intLeft"* e *"intTop"* para posicionar a abertura do *faceplate* do equipamento mais ao centro possível da tela da HMI. E no parâmetro *"Slot_Number"* deve ser inserido o número do *slot* que está sendo configurado o objeto, apenas para animação de texto dentro do objeto e *faceplates*.

3.9.5. Configuração da tela de gerenciamento de *Slots*

Esta tela foi desenvolvida para visualizar todas as receitas disponíveis no projeto e realizar o descarregamento da receita no *slot* de operação selecionado. É uma tela de gerenciamento para o operador. Como mostra a Figura 149.

Figura 149 - Objetos de animação para a tela de gerenciamento de *slots*.

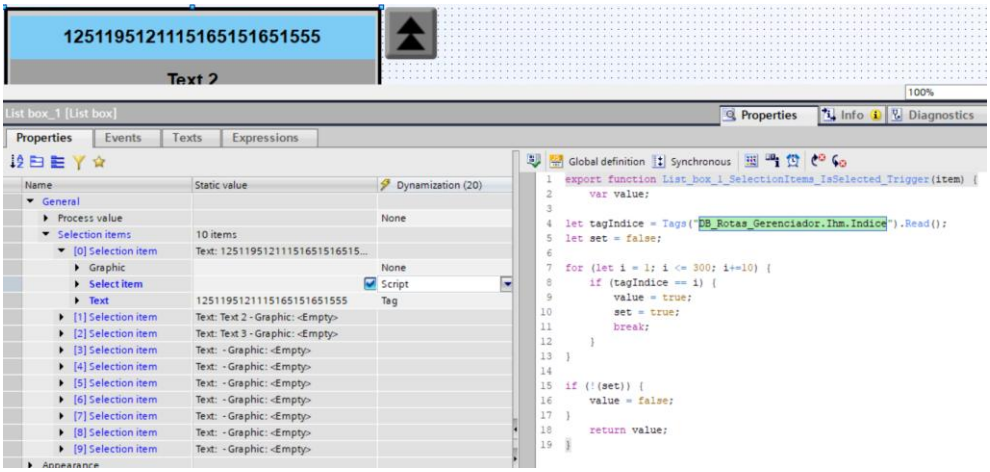


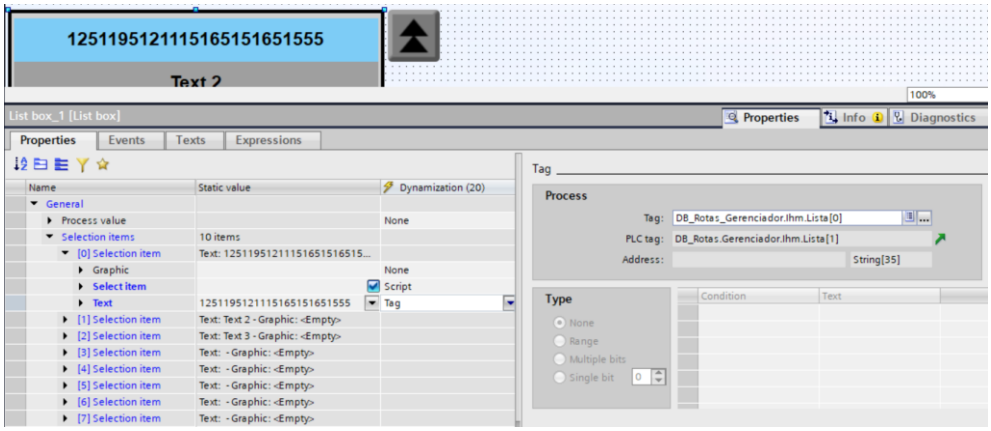
Fonte: Autoria própria.

Para inserir a lista de receita disponíveis, foi utilizado um objeto do tipo *"List_box"*, onde foi configurado para apresentar a lista de receitas em vetores de dez espaços por vez. Nesse objeto foi desenvolvido uma animação navegar a seleção da receita e uma animação de vinculação com a descrição da receita; apresentada na Figura 150.

Figura 150 - Animação da a) Seleção da posição da lista da receita e da b) Descrição da posição da receita.

a)

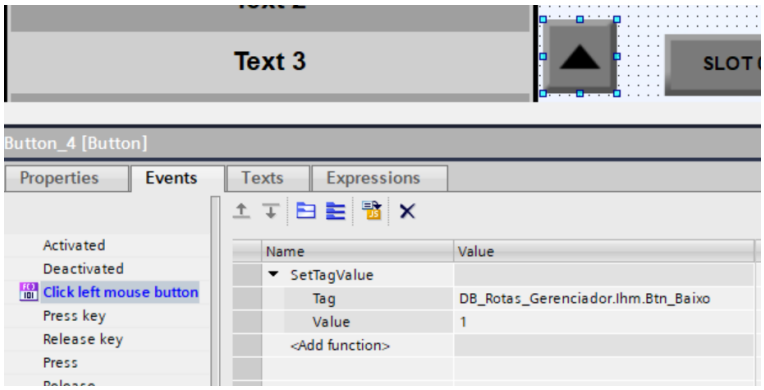


b) 

Fonte: Autoria própria.

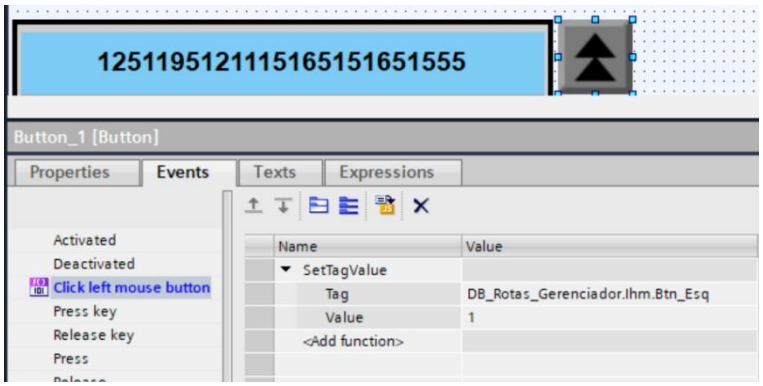
Para navegar entre as posições da lista de receita, utilizou se os botões laterais do objeto de lista. Com esses botões é possível navegar entre a lista de receita, utilizando comando de navegação de uma posição por vez ou dez posições por vez; como mostra a Figura 151 e a Figura 152.

Figura 151 - Botão de navegação de uma posição da lista de receita.



Fonte: Autoria própria.

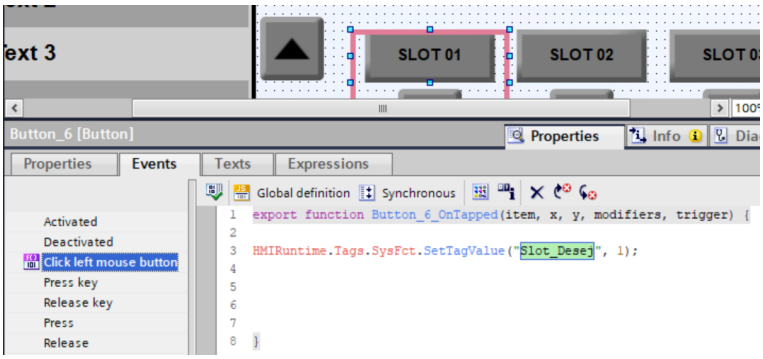
Figura 152 - Botão de navegação de dez posição da lista de receita.



Fonte: Autoria própria.

Uma vez selecionado a receita que será descarregada no *Slot* de operação, é necessário selecionar o *slot* que esteja vazio que possa receber a receita nova. Utilizando a memória *"Slot_Desej"* no código de ação da Figura 153, será selecionado o *slot*. Nessa janela de gerenciamento também está presente todas as chamadas dos *slots* disponíveis, então é possível verificar *slot* por *slot*, qual está vazio para poder receber a nova receita de operação.

Figura 153 - Botão de seleção de *slot* para o descarregamento da receita.

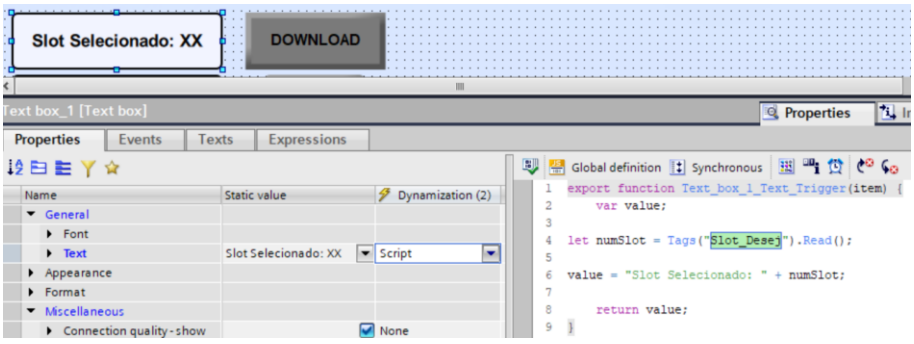


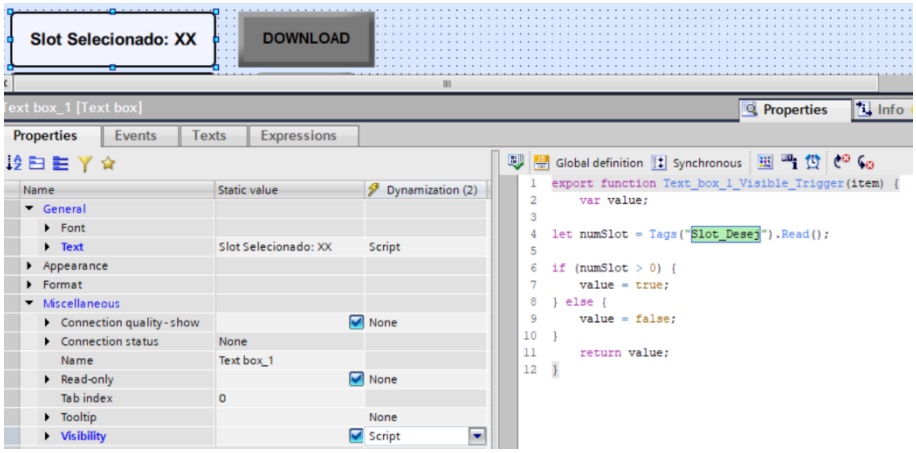
Fonte: Autoria própria.

Para verificar o número do *slot* selecionado, basta verificar a caixa de texto que se encontra embaixo da lista de receita, essa caixa de texto possui animação de visibilidade, quando não há nenhum *slot* selecionado, e de texto apresentando o *slot* selecionado para o descarregamento; como mostrada na figura 154

Figura 154 - Animação de a) Texto indicando o slot selecionado para o descarregamento e da b) Visibilidade do texto na tela da *HMI*.

a)

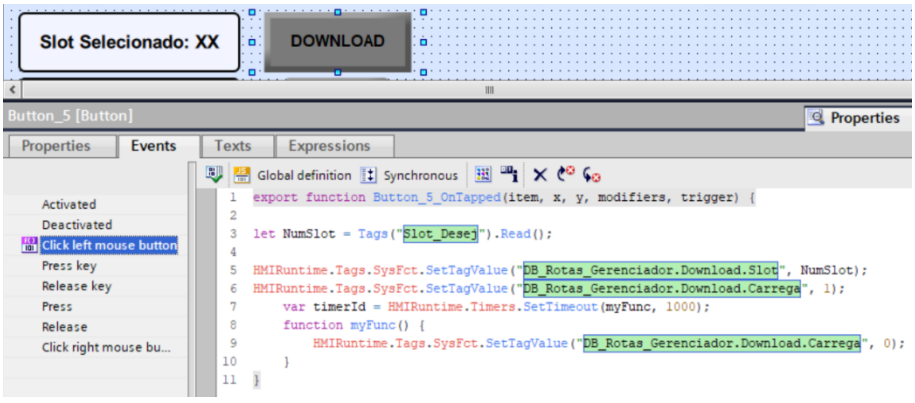


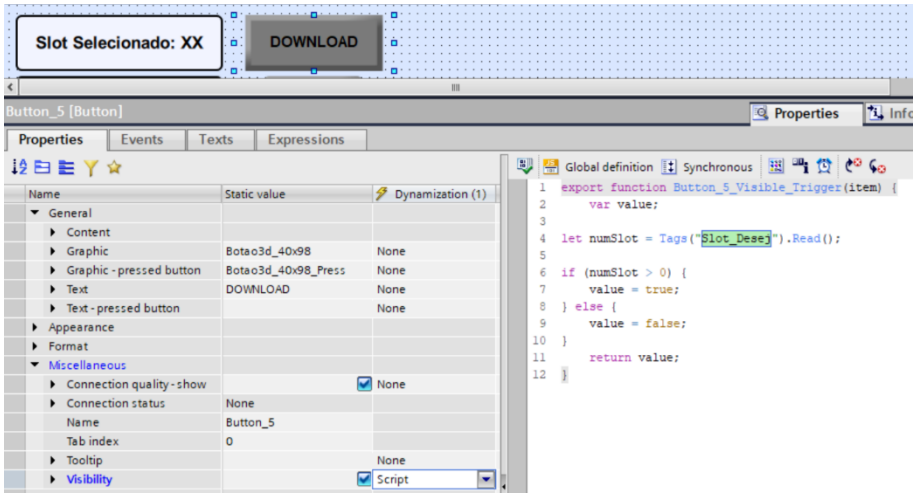
b) 

Fonte: Autoria própria.

Uma vez selecionado a receita e o *slot*, o botão *"DONWLOAD"* se encontrará habilitado para pressionar. E uma vez pressionado, o botão enviará o número do *slot* para o programa do *PLC* e iniciará o descarregamento da receita no *slot* selecionado; como mostra a Figura 155.

Figura 155 - Animação da a) Ação de comando para o descarregamento da receita no *slot* selecionado e da b) Visibilidade do botão na tela da *HMI*.

a) 

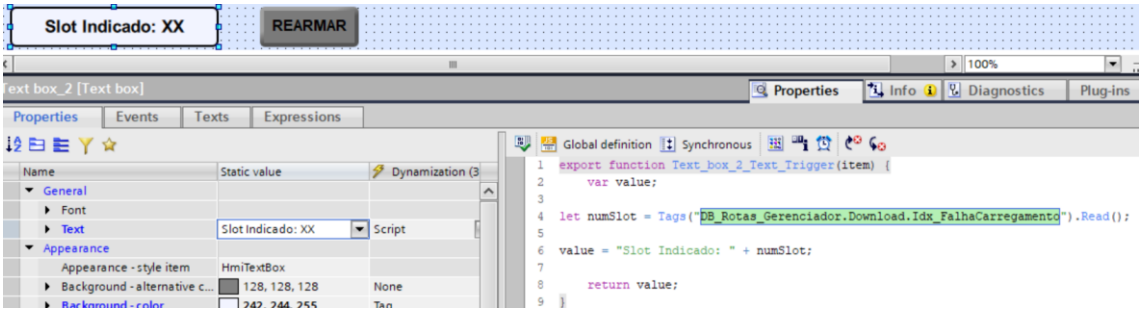
b) 

Fonte: Autoria própria.

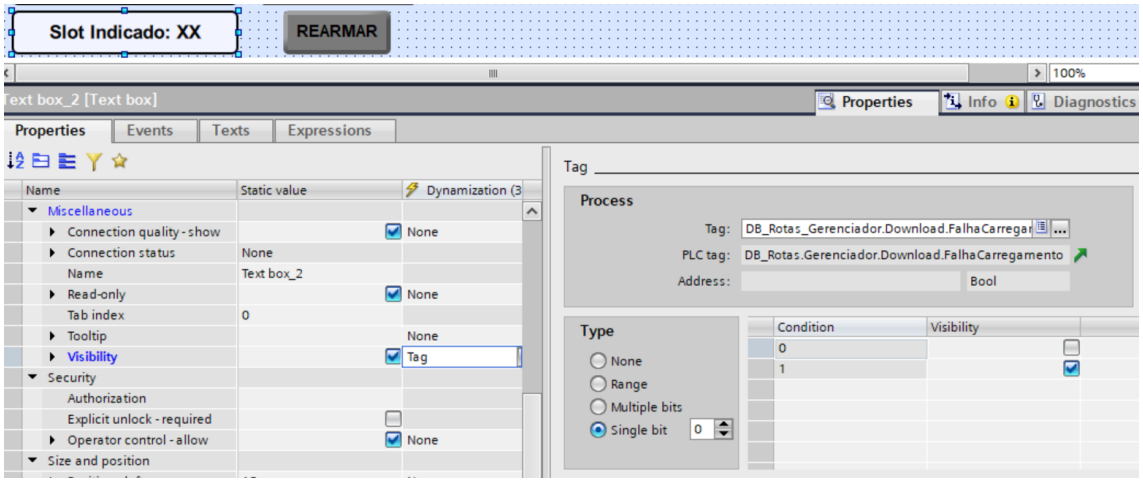
Caso venha a acontecer algum erro no descarregamento da receita, por conta da receita incompleta ou por causa do *slot* selecionado já possuir uma receita em operação. O número do *slot* em questão aparecerá na última caixa de texto com uma animação piscando avermelhada, indicando uma falha no descarregamento; essas animações utilizaram a memória *"DB_Rotas_Gerenciador.Download.FalhaCarregamento"* como mostrada na Figura 156.

Figura 156 - Animação de a) Texto indicando o slot em que descarregamento falhou, da b) Visibilidade do texto na tela da *HMI* e da c) Coloração piscante avermelhada indicando uma falha.

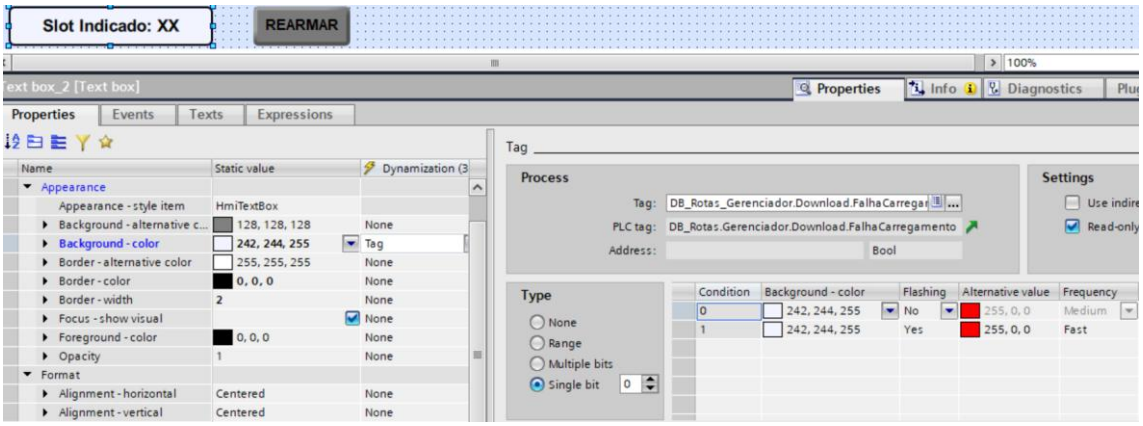
a)



b)



c)

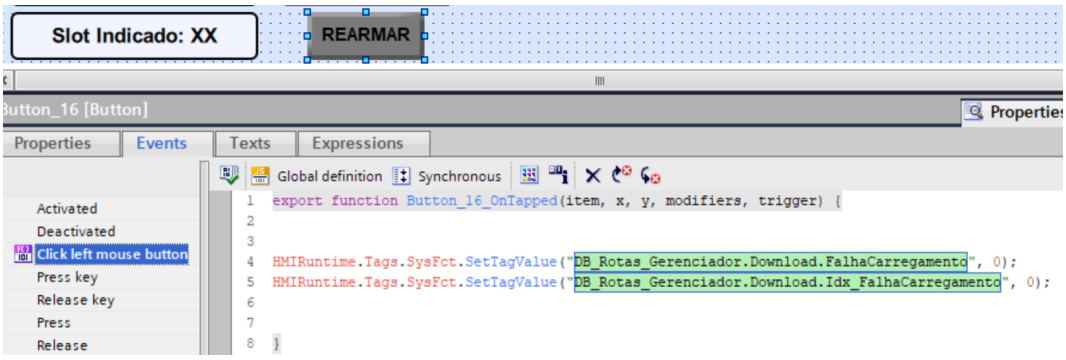


Fonte: Autoria própria.

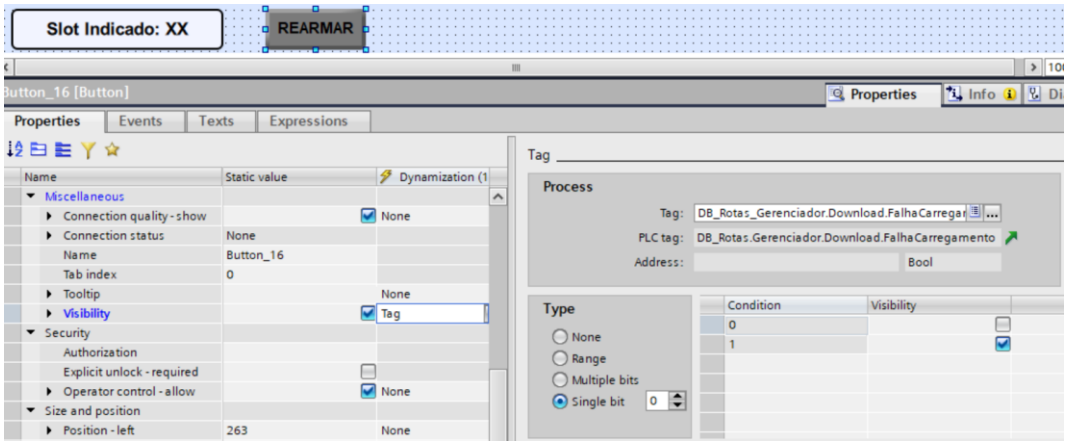
E assim como as demais falhas do projeto, mesmo que o slot da falha esteja limpa e pronta para a próxima receita, é necessário rearmar o alarme para liberar as funções de descarregamento voltem a normalidade. Utilizando as memórias *"DB_Rotas_Gerenciador.Download.FalhaCarregamento"* e *"DB_Rotas_Gerenciador.Download.Idx_FalhaCarregamento"*, como mostra a Figura 157.

Figura 157 - Animação da a) Ação de comando para o limpar e rearmar o alarme e da b) Visibilidade do botão na tela da *HMI*.

a)



b)



Fonte: Autoria própria.

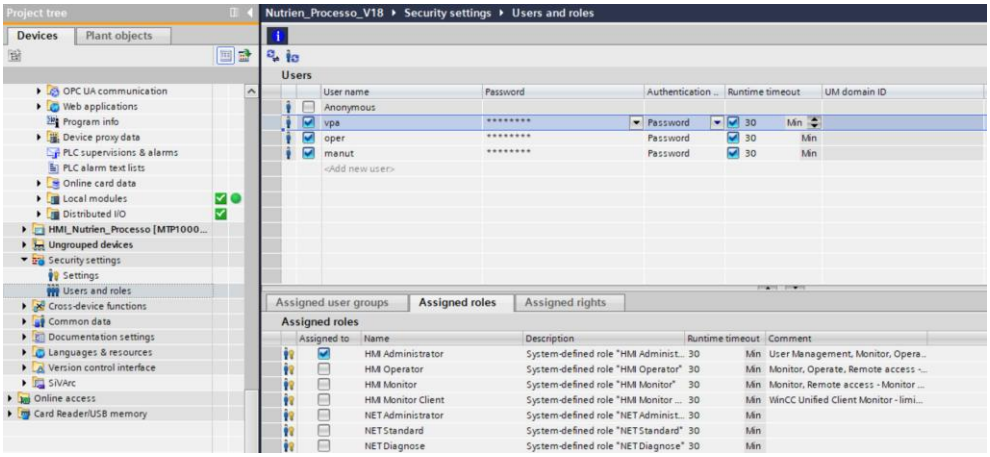
4. RESULTADOS

Após desenvolvido todas as imagens, objetos de animações e telas principais. Utilizando os softwares disponíveis da *Siemens*, pode se simular o código do PLC e as telas da HMI.

4.1. Configuração da janela de login e senha do projeto

A única diferença com máquina HMI, é que quando ocorre qualquer tipo de *download* das telas da HMI, para poder acessá-las é necessário inserir um *login* e senha de usurário do sistema. Sua configuração de acesso e entrada de usuário novo se mostra pela Figura 158.

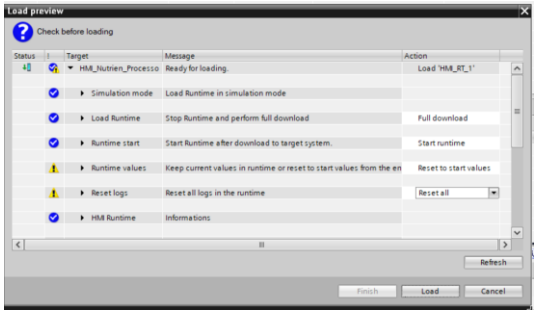
Figura 158 - Janela de configuração de login e senha de usuário do projeto.



Fonte: Autoria própria.

Após a configuração, ela pode ser usada para a simulação ou para máquina de HMI. Lembrando um detalhe da máquina HMI, para que o login e senha sejam validos, é necessário realizar um descarregamento do projeto completo, com as opções de descartar os dados atuais e limpar a memória de *logs*; como mostra a Figura 159.

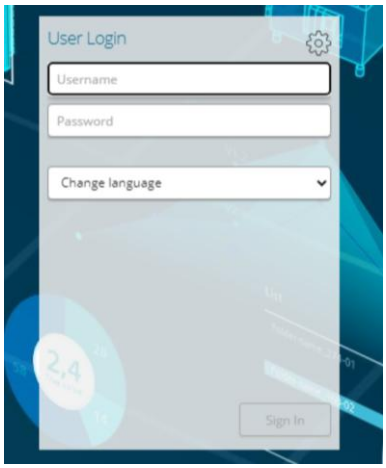
Figura 159 - Janela de configuração de login e senha de usuário do projeto.



Fonte: Autoria própria.

Em seguida para inserir os dados de *login* e senha, basta iniciar a simulação a primeira vez ou pressionar o botão com a função de *"LogOff"*, vista pela Figura 27. Assim pode se apresentar uma janela *login* igual a Figura 160.

Figura 160 - Janela de *Login* do usuário.

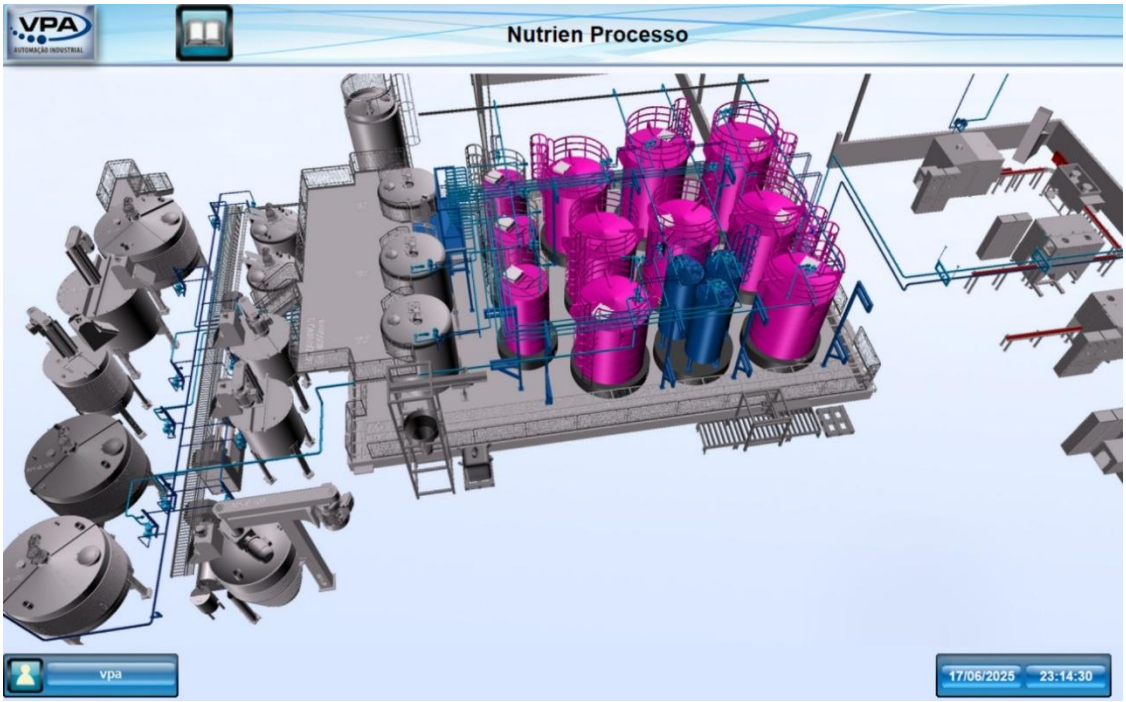


Fonte: Autoria própria.

4.2. Telas desenvolvidas no projeto

4.2.1. Tela inicial do projeto.

Figura 161 - Tela inicial do projeto.



Fonte: Autoria própria.

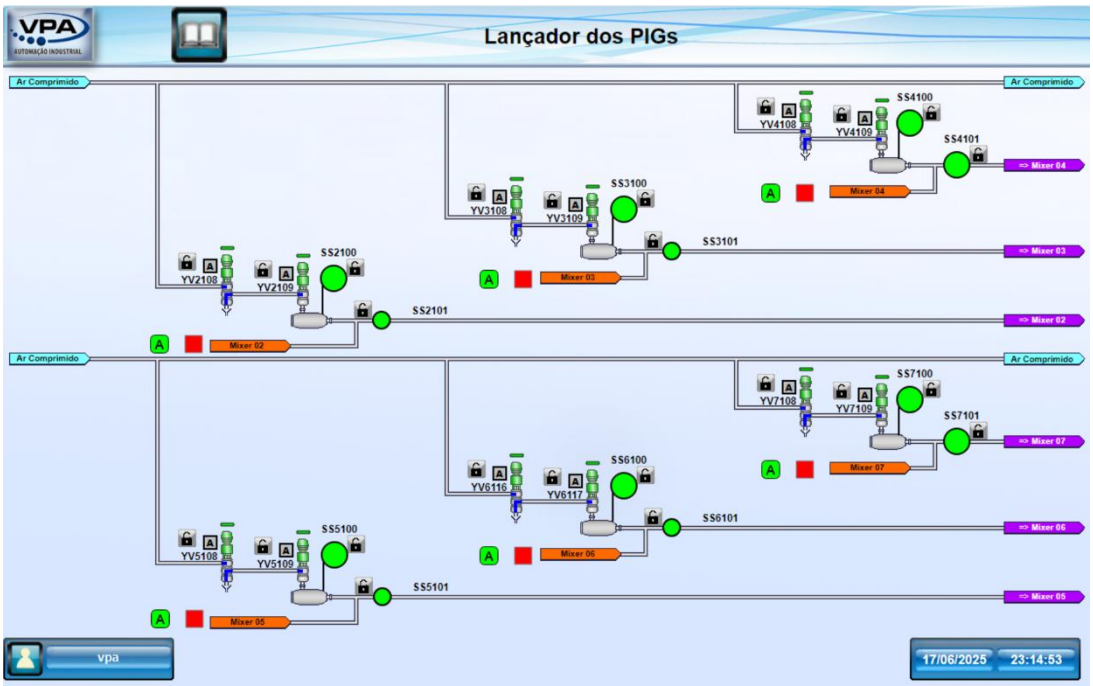
4.2.2. *Pop Up* do menu de telas do projeto

Figura 162 - *Pop Up* do menu de telas do projeto.

Fonte: Autoria própria.

4.2.3. Tela do Lançador de *PIG* (Parte 01)

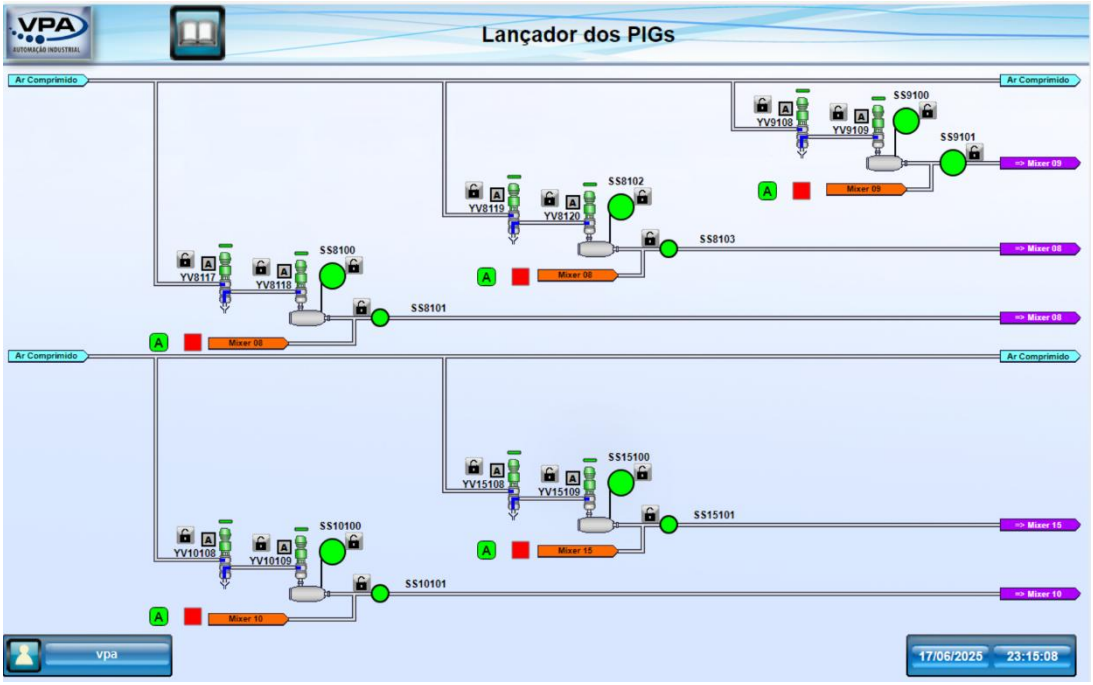
Figura 163 - Tela do Lançador de *PIG* (Parte 01).



Fonte: Autoria própria.

4.2.4. Tela do Lançador de *PIG* (Parte 02)

Figura 164 - Tela do Lançador de *PIG* (Parte 02).



Fonte: Autoria própria.

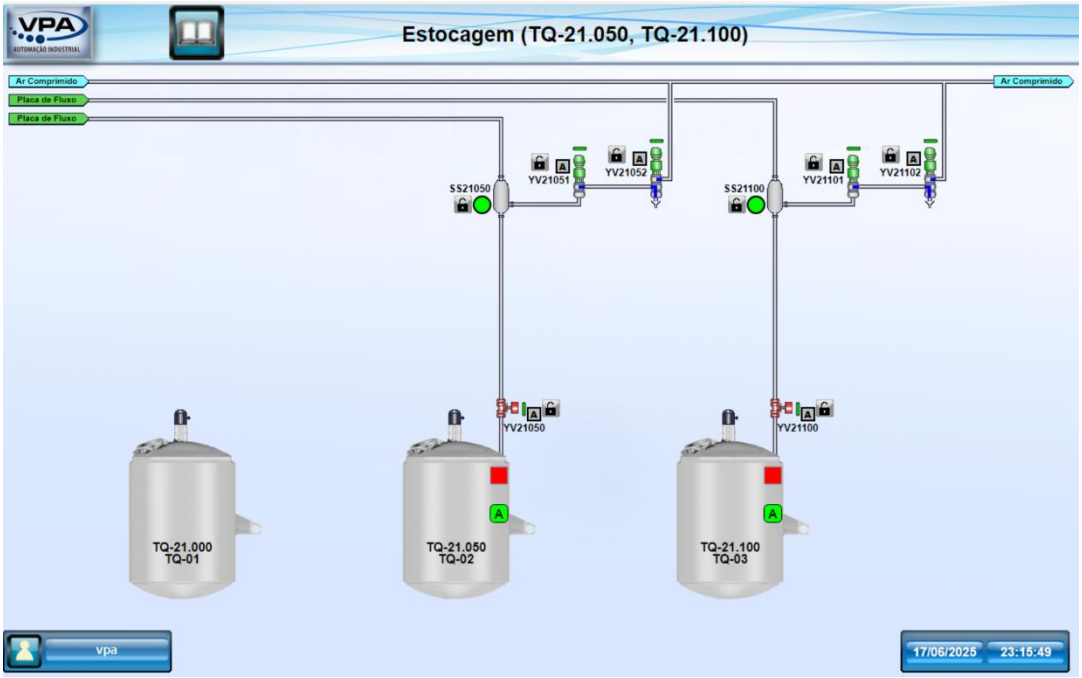
4.2.5. Tela da Placa de Fluxo

Figura 165 - Tela da Placa de Fluxo.

Fonte: Autoria própria.

4.2.6. Tela da Estocagem (TQ-21.050, TQ-21.100)

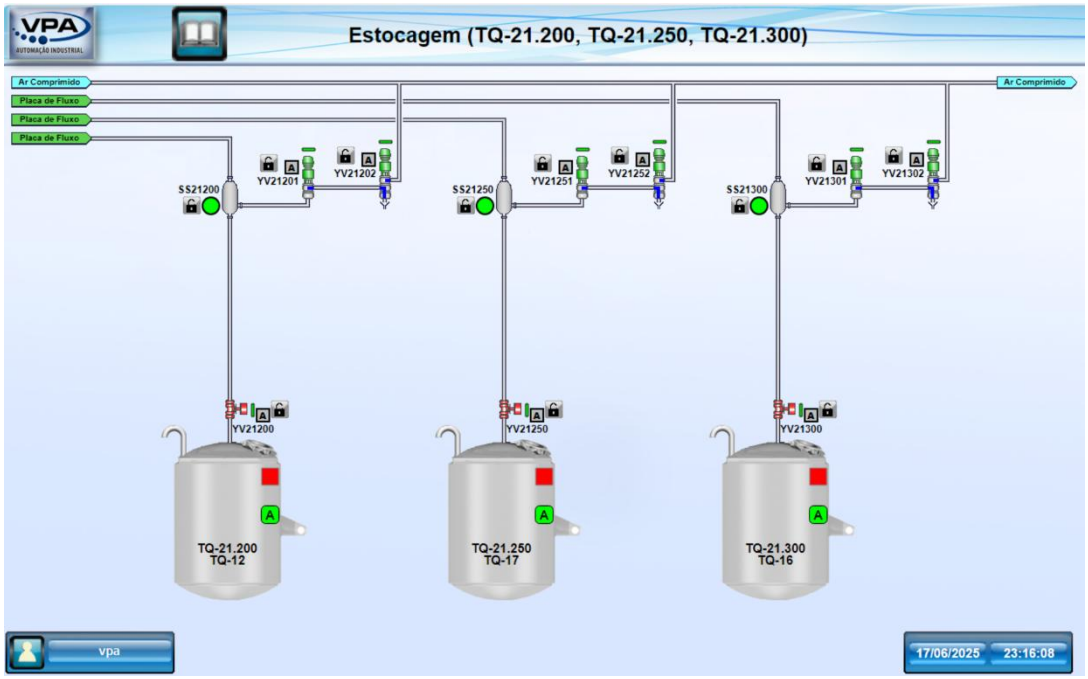
Figura 166 - Tela da Estocagem (TQ-21.050, TQ-21.100).



Fonte: Autoria própria.

4.2.7. Tela da Estocagem (TQ-21.200, TQ-21.250, TQ-21.300)

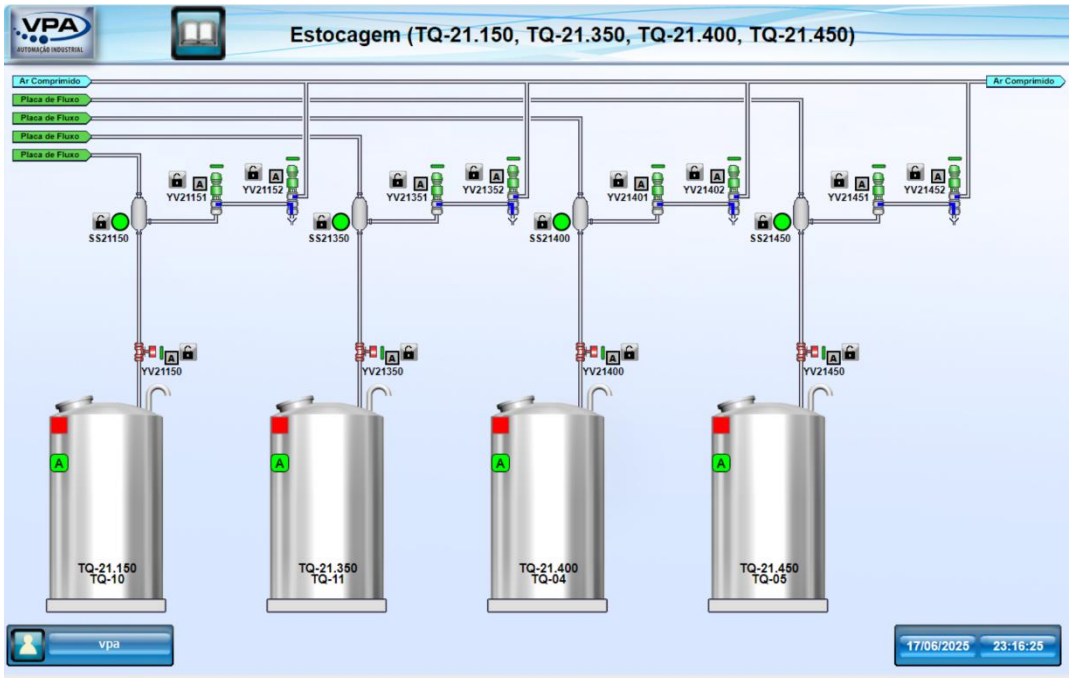
Figura 167 - Tela da Estocagem (TQ-21.200, TQ-21.250, TQ-21.300).



Fonte: Autoria própria.

4.2.8. Tela da Estocagem (TQ-21.150, TQ-21.350, TQ-21.450)

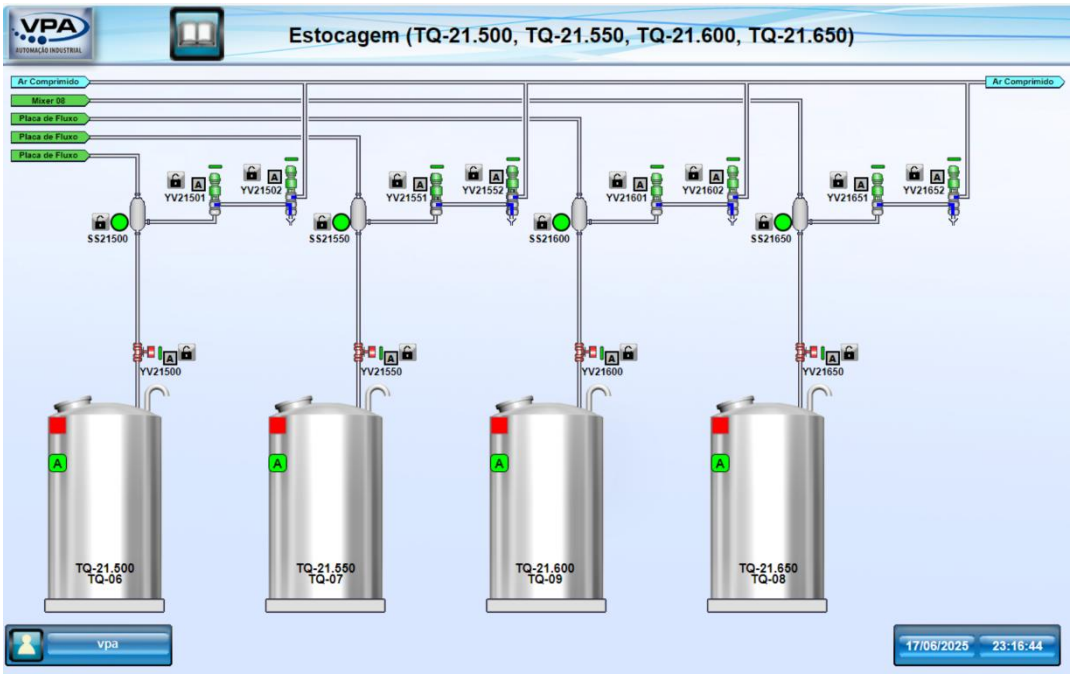
Figura 168 - Tela da Estocagem (TQ-21.150, TQ-21.350, TQ-21.450).



Fonte: Autoria própria.

4.2.9. Tela da Estocagem (TQ-21.500, TQ-21.600, TQ-21.650)

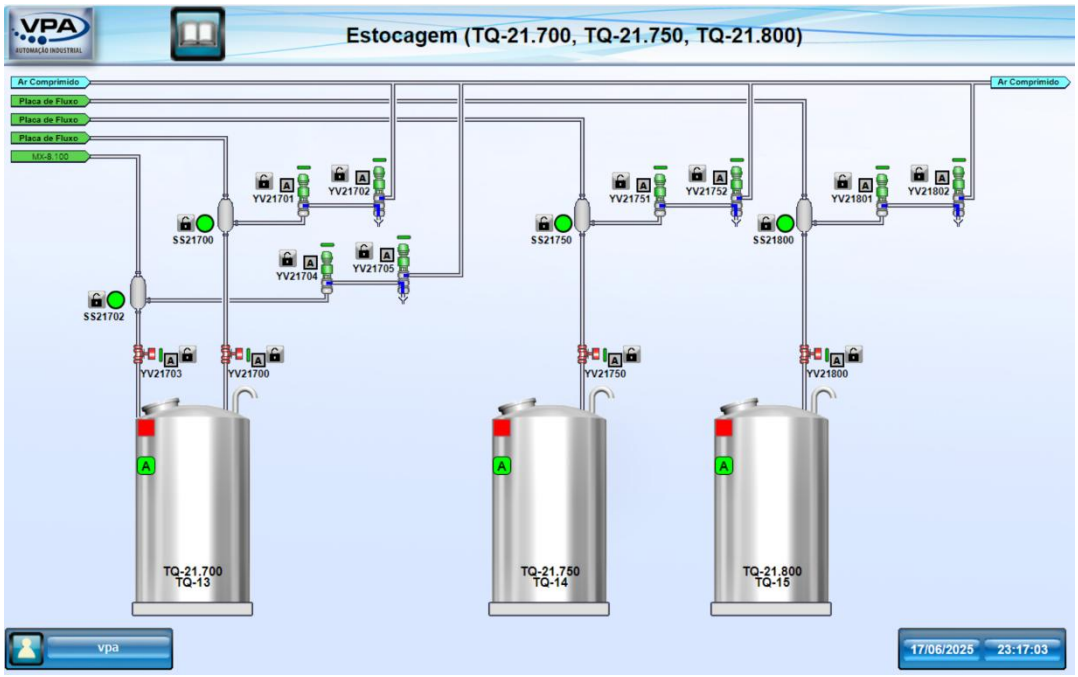
Figura 169 - Tela da Estocagem (TQ-21.500, TQ-21.600, TQ-21.650).



Fonte: Autoria própria.

4.2.10. Tela da Estocagem (TQ-21.700, TQ-21.750, TQ-21.800)

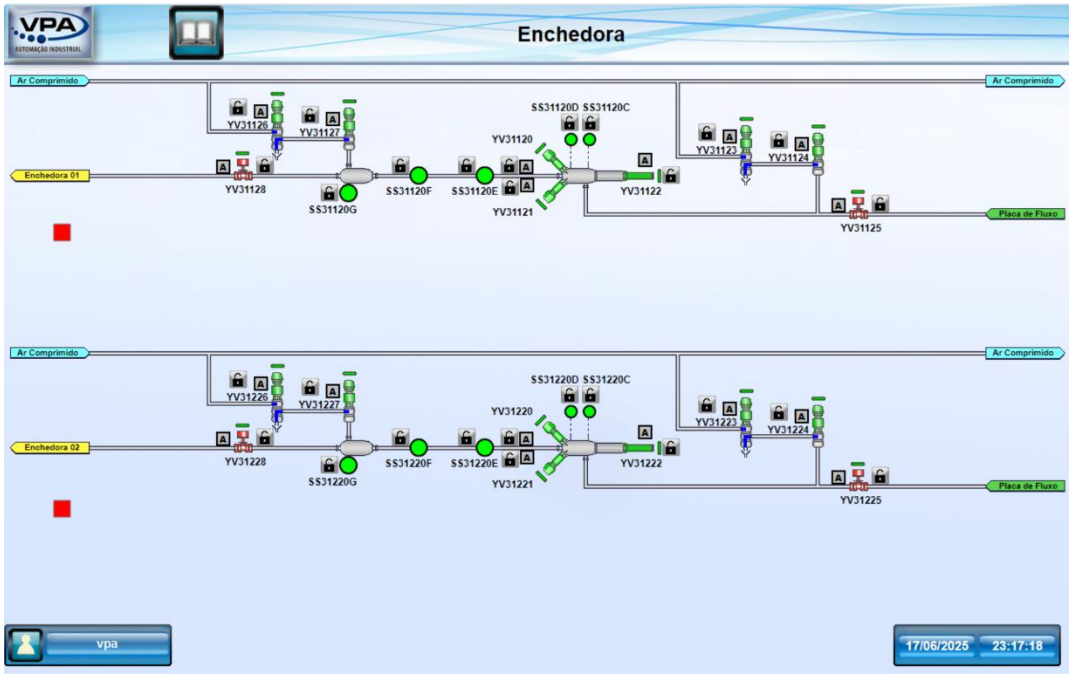
Figura 170 - Tela da Estocagem (TQ-21.700, TQ-21.750, TQ-21.800).



Fonte: Autoria própria.

4.2.11. Tela da Enchedora

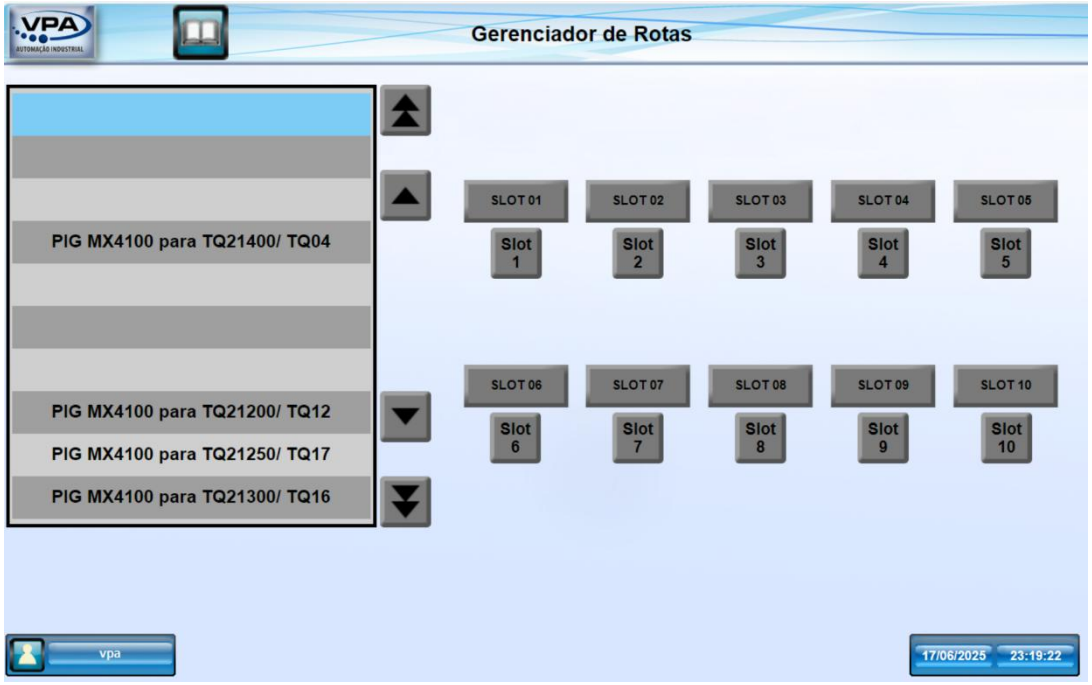
Figura 171 - Tela da Enchedora.



Fonte: Autoria própria.

4.2.12. Tela do Gerenciador de Rotas

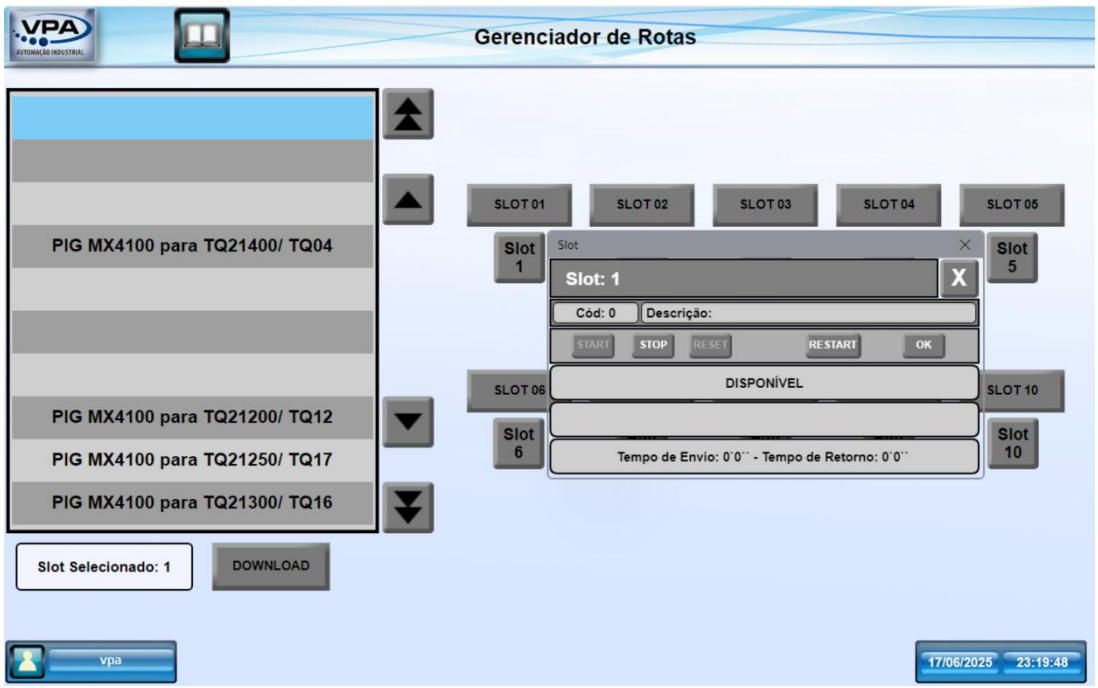
Figura 172 - Tela do Gerenciador de Rotas.



Fonte: Autoria própria.

4.2.13. Seleção de *Slot* e descarregamento de receita

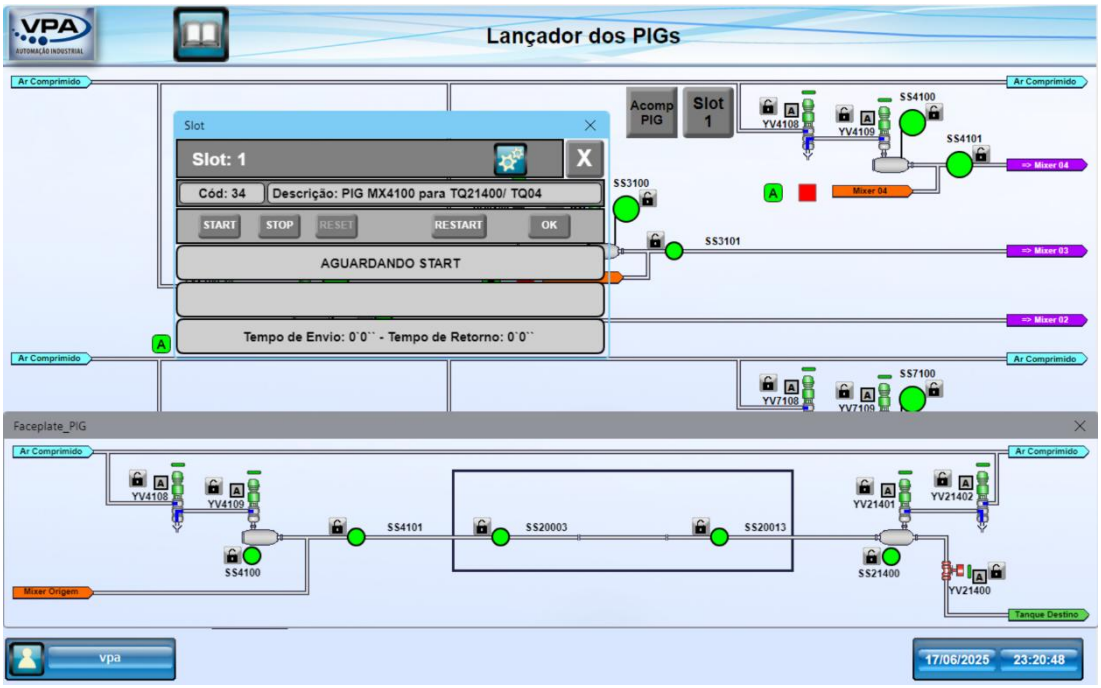
Figura 173 - Seleção de *Slot* e descarregamento de receita.



Fonte: Autoria própria.

4.2.14. Visualização do *faceplate* de acompanhamento da receita

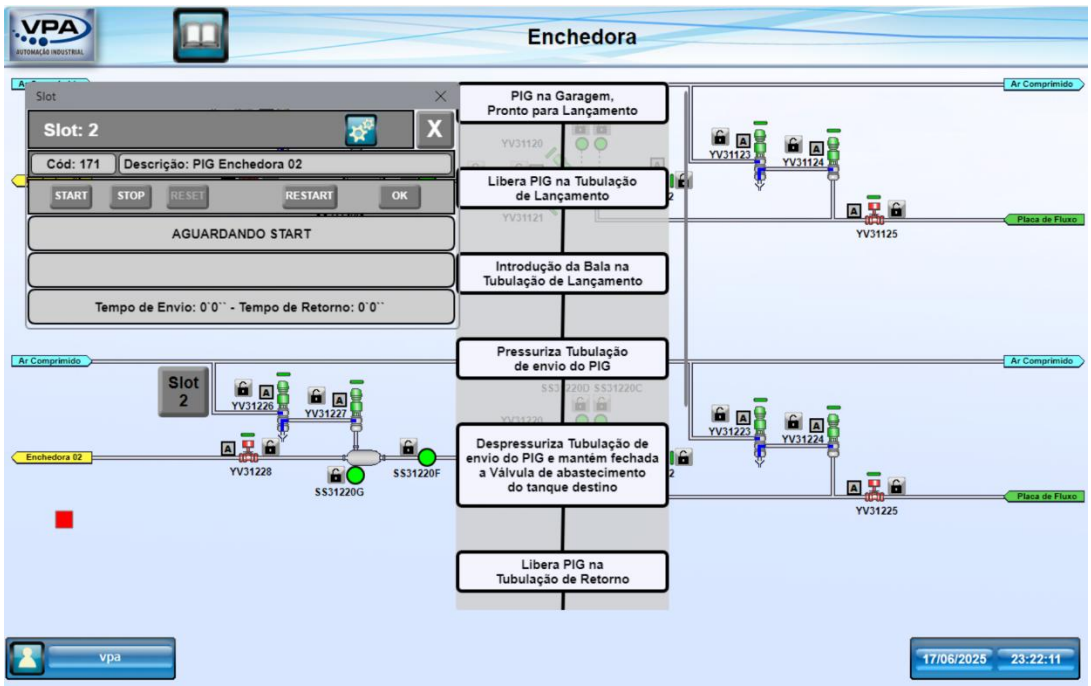
Figura 174 - Visualização do *faceplate* de acompanhamento da receita.



Fonte: Autoria própria.

4.2.15. Visualização do *faceplate* de fluxograma da receita

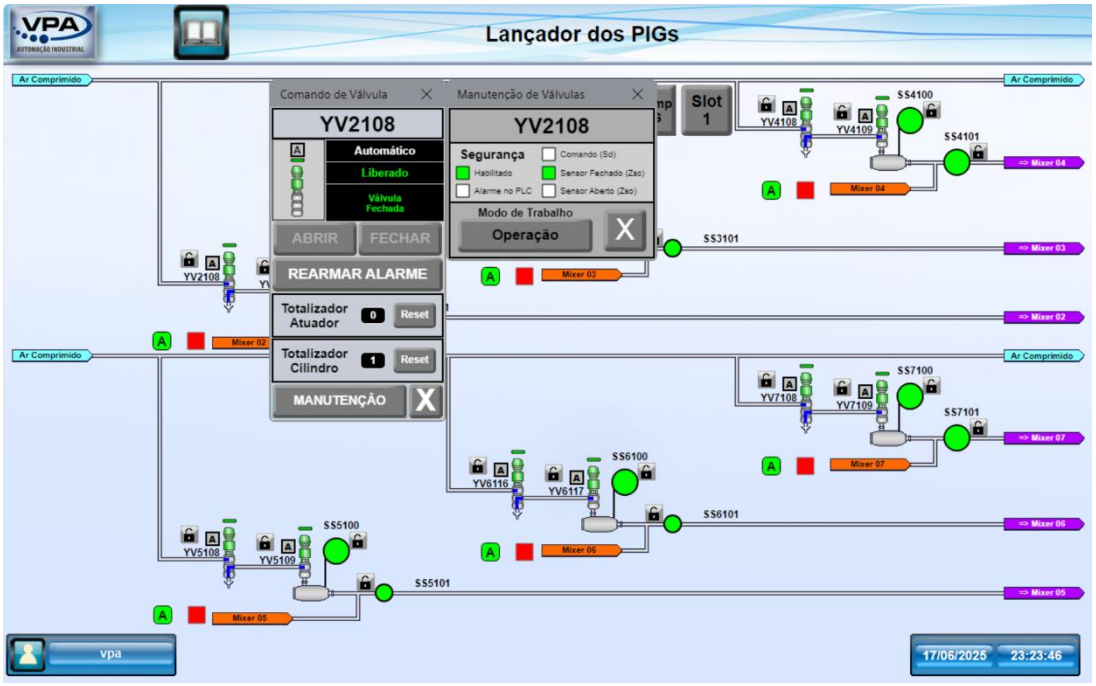
Figura 175 - Visualização do *faceplate* de fluxograma da receita.



Fonte: Autoria própria.

4.2.16. Visualização do *faceplate* das válvulas

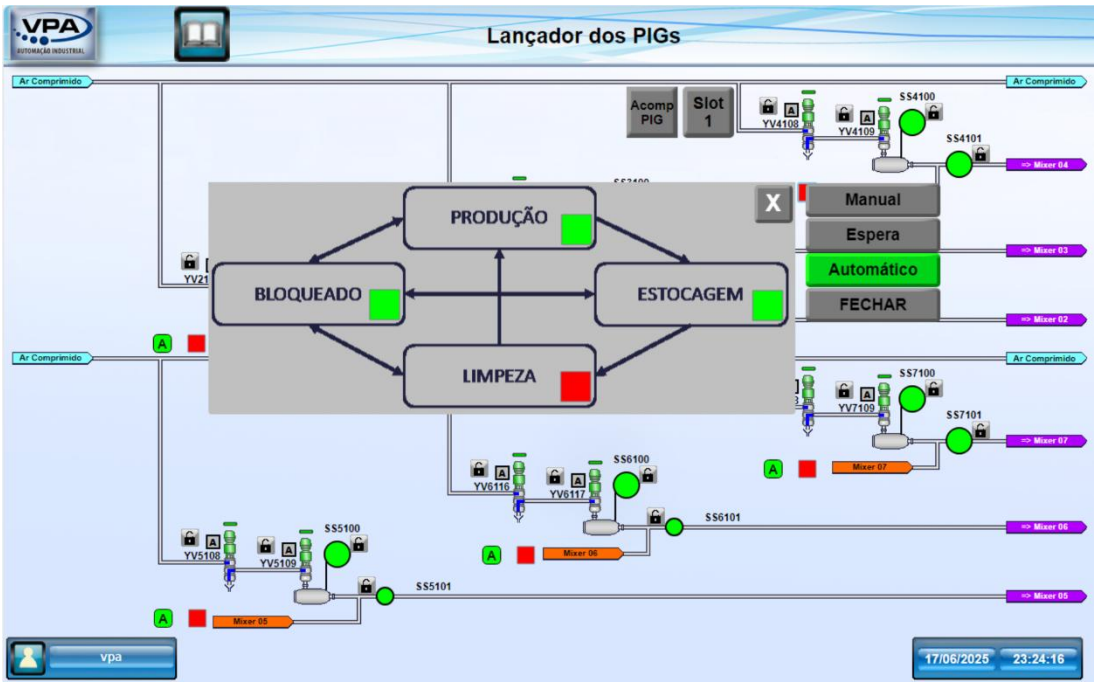
Figura 176 - Visualização do *faceplate* das válvulas.



Fonte: Autoria própria.

4.2.17. Visualização do *faceplate* dos tanques

Figura 177 - Visualização do *faceplate* dos tanques.



Fonte: Autoria própria.

4.3. Discussão

Como foi dito no começo do documento, a empresa cliente estava com um problema muito grande em relação ao tempo de ciclo de limpeza dos seus equipamentos, necessário para iniciar a produção do próximo produto. Por se tratar de uma ação totalmente manual, necessitava de muitas mãos e olhos no processo, além de despender quase um dia inteiro para finalizar o processo de limpeza.

Porém, agora, com a implementação do sistema automatizado de limpeza do tipo CIP, o tempo do procedimento de limpeza foi reduzido para menos de 30 min - o que representa um ganho significativo de eficiência para empresa, que pode iniciar a produção do próximo produto no mesmo dia da finalização do anterior.

Além de necessitar de menos pessoas para acompanhar o processo de limpeza, o controlador PLC o tempo todo monitorando todos os equipamentos envolvidos nos passos de um ciclo de limpeza. Desta forma, mesmo que ocorra um mal funcionamento de uma válvula ou sensor, o bloco de monitoramento dos passos do ciclo irá intertravar o procedimento em que este equipamento se encontra e não deixará dar início ou retomar os passos, até que o problema seja sanado.

Assim, devido a este monitoramento refinado do controlador PLC, o operador tem uma visão mais operacional dos passos do ciclo da limpeza. Todas essas informações e comandos de execução se encontram nas telas de comando da HMI.

5. CONCLUSÕES

Observando os resultados obtidos na simulação do projeto e no acompanhamento da instalação do projeto na empresa cliente, pode se dizer que o resultado é muito positivo com relação ao ganho de eficiência no tempo do procedimento de limpeza.

Antes o processo de limpeza era realizado manualmente, o que requisitava de mais pessoas para: acompanhar o progresso da limpeza; garantir que o procedimento fosse finalizado com segurança; validar a conclusão da limpeza para dar início na produção do próximo produto.

A implementação da automação do sistema de limpeza CIP permite realizar todos esses passos com menos recursos e menos pessoas para acompanhamento, uma vez que todo o processo de limpeza pode ser visualizado nas telas da HMI. O sistema mantém o mesmo nível de segurança ou até maior, já que o controlador PLC monitora todos os equipamentos envolvidos no ciclo de limpeza ativa. Além disso, oferece uma limpeza quase perfeita das tubulações e válvulas, pois utiliza uma pressão de ar muito forte para remover todo o produto químico remanescente (sistema CIP). Como consequência, o ciclo de limpeza dos equipamentos é finalizado em um tempo significativamente menor - no caso estudado, levou menos de 30 minutos.

O método de aplicação do projeto não precisa se limitar apenas a esta situação de limpeza. A organização dos equipamentos utilizados pode ser implementada em outros contextos, como motores de partida direta ou por inversor, sensores analógicos ou válvulas analógicas. Dessa forma, podem ser desenvolvidos novos objetos de animação parametrizados.

Portanto, dependendo das necessidades do ambiente dos equipamentos para a solução, podem ser inseridos diversos objetos nas telas de comando da HMI, como em um 'quebra-cabeça'. É importante ressaltar que a configuração correta dos parâmetros é essencial para evitar a animação dos objetos em posições incorretas.

REFERÊNCIAS

GEA - GEA VALVE AUTOMATION. 2024. Disponível em https://www.gea.com/en/assets/262615/. Acessado 18/10/2024

HIGTOP - Conheça a limpeza CIP e sua importância para a indústria. Disponivel em https://higtop.com.br/2017/04/18/conheca-a-limpeza-cip/. Acessado em 02/08/2024.

JavaScript. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2024. Disponível em: https://pt.wikipedia.org/wiki/JavaScript. Acesso em: 18/10/2024.

Kalatec Automação - Linguagem de programação Ladder: aplicações, exemplos e dicas! 2025. Disponível em https://blog.kalatec.com.br/linguagem-programacao-ladder/. Acessado em 18/068/2025.

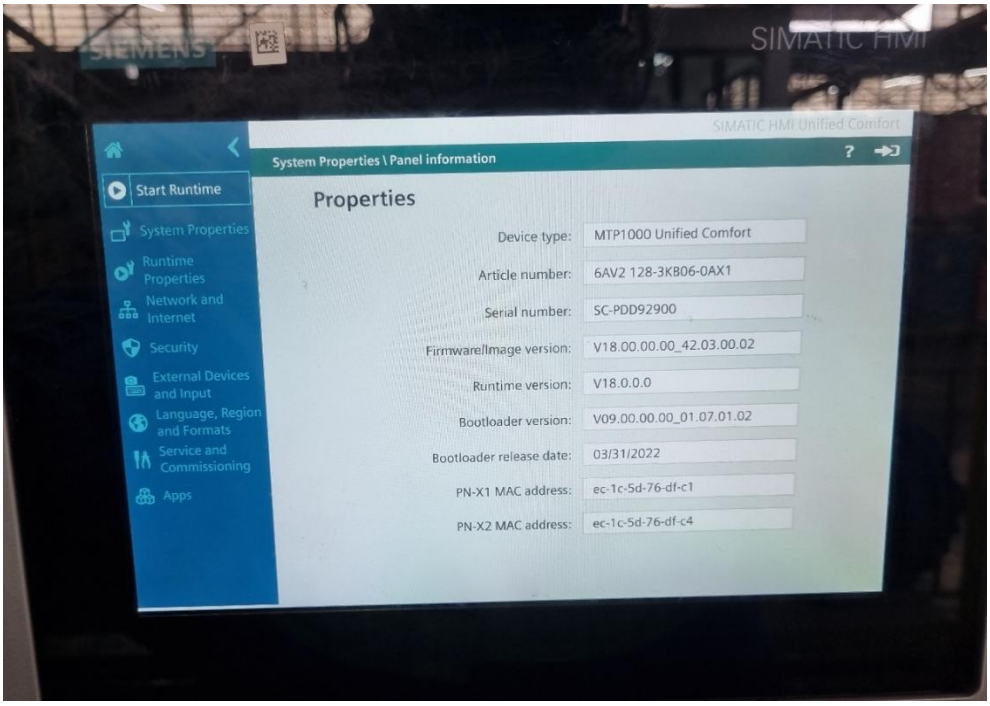
Siemens - SIMATIC HMI WinCC Unified Engineering V18. 2022 Disponível em https://support.industry.siemens.com/cs/mdm/109813308?c=162513941515&lc=en-BR. Acessado em 18/10/2024

Siemens - Portal de Automação Totalmente Integrado – Sempre pronto para o amanhã. Disponível em https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html. Acessado em 18/06/2025

Siemens - SIMATIC WinCC Unified - Tips and Tricks for Scripting (JavaScript). 2020. Disponível em https://support.industry.siemens.com/cs/document/109758536/simatic-wincc-unified-tips-and-tricks-for-scripting-(javascript)?dti=0&lc=en-BR . Acessado em 18/10/2024

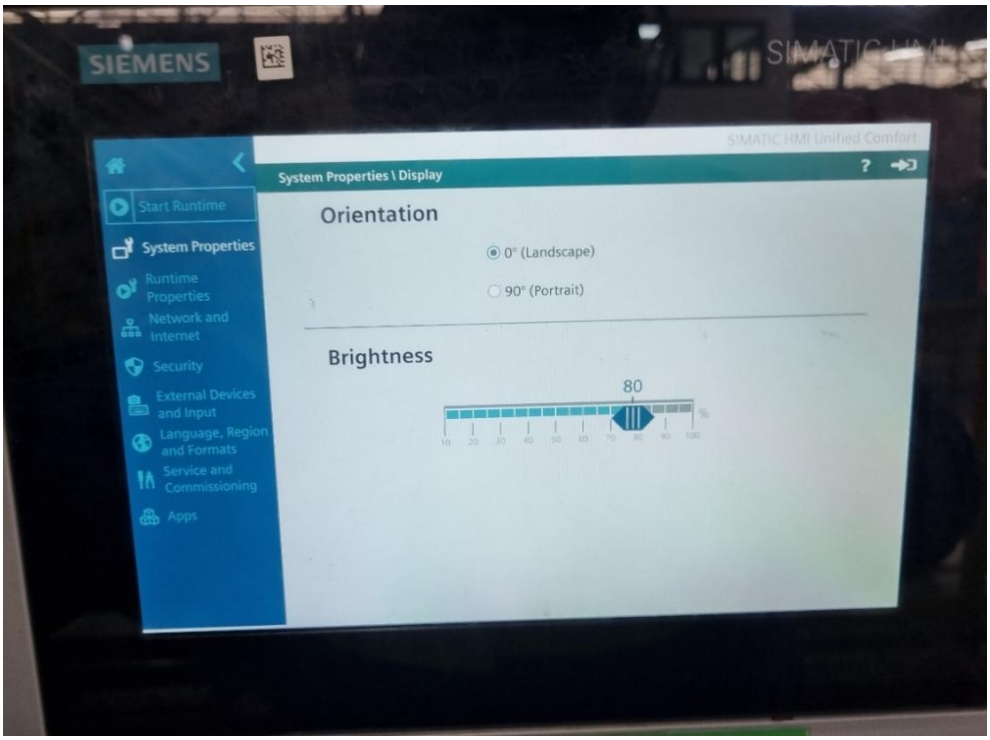
APÊNDICE A - Telas de visualização/configuração das propriedades da HMI

Figura 178 - Janela de informações da *HMI*.



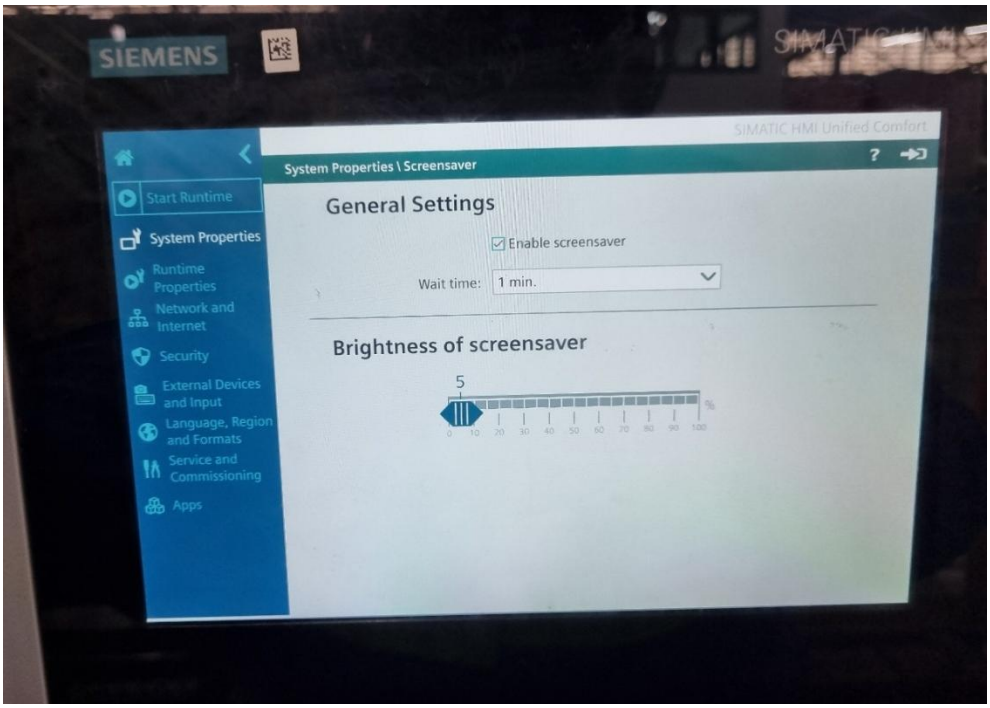
Fonte: Autoria própria.

Figura 179 - Janela de configuração de brilho da *HMI*.



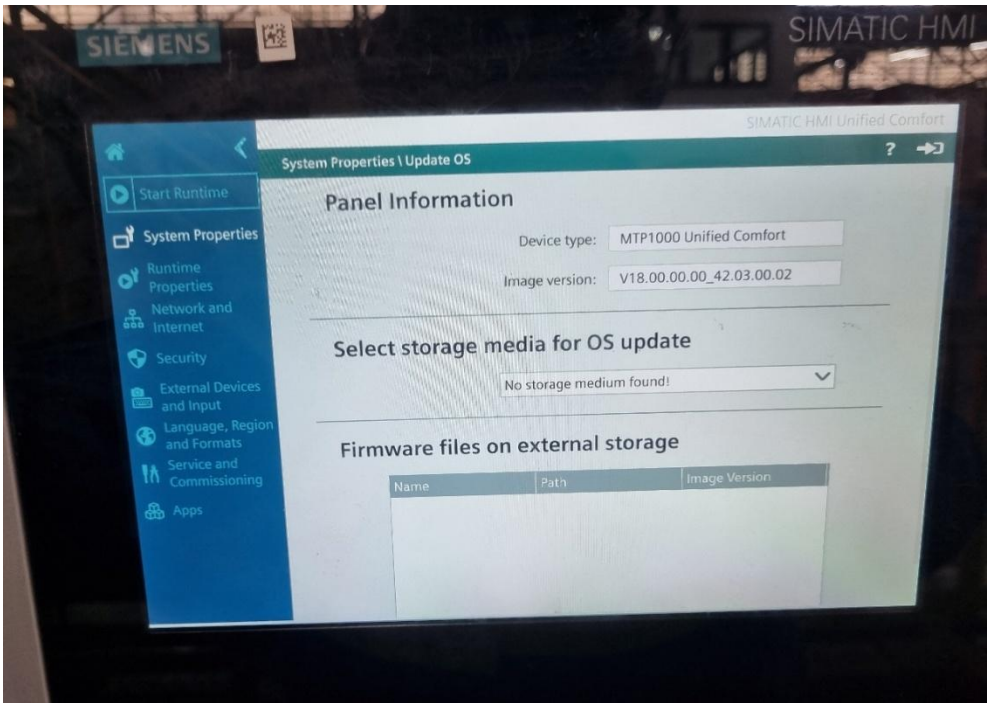
Fonte: Autoria própria.

Figura 180 - Janela de configuração de tempo ativo de tela da *HMI*.



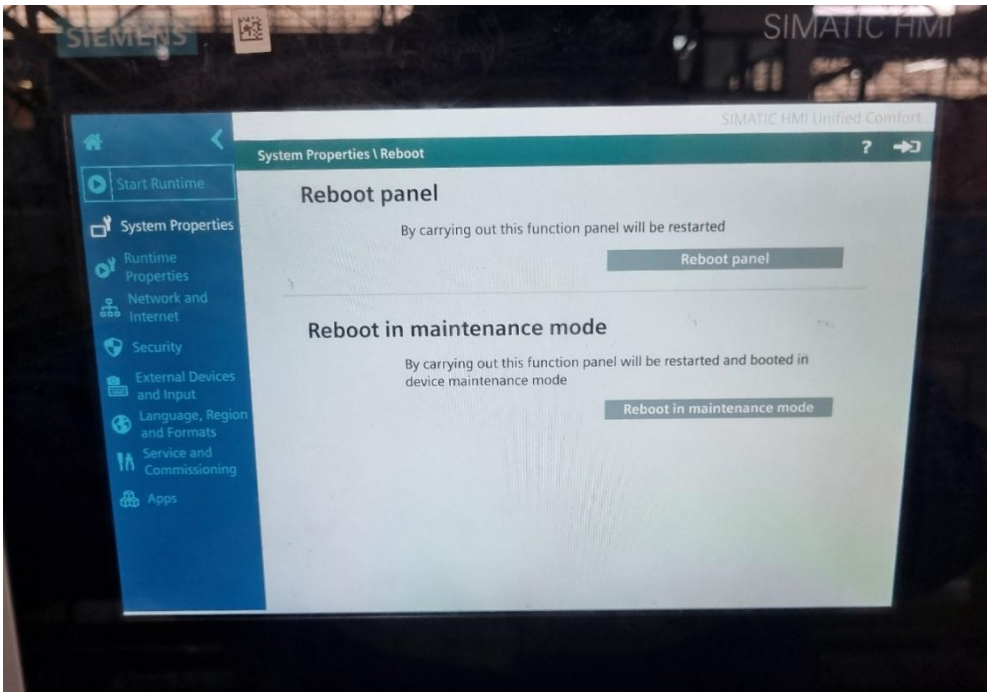
Fonte: Autoria própria.

Figura 181 - Janela de atualização de *firmware* da *HMI*.



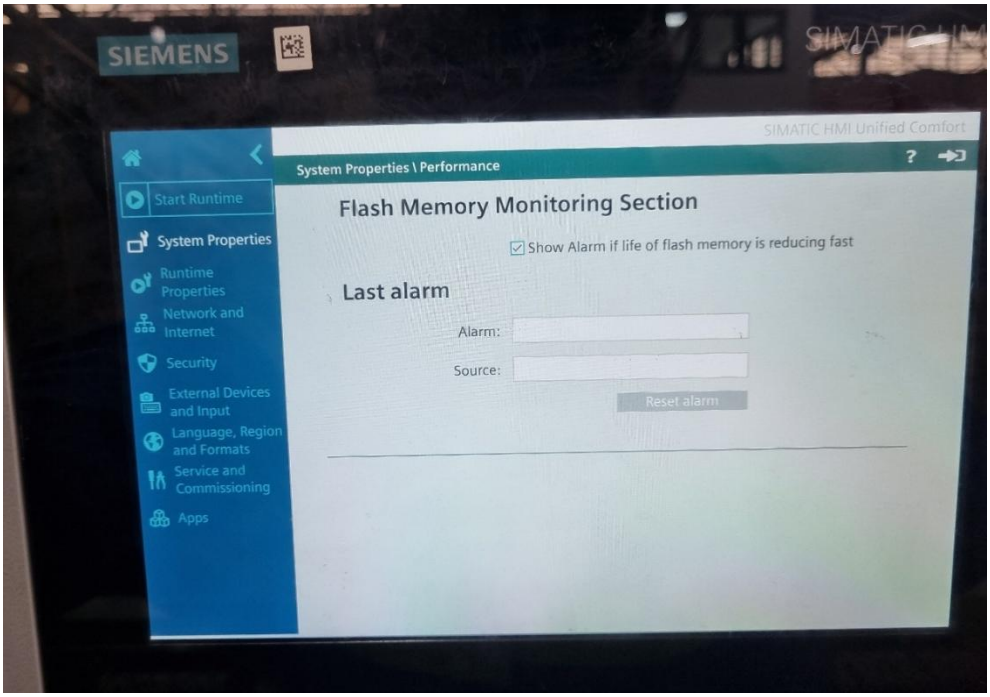
Fonte: Autoria própria.

Figura 182 - Janela de *reboot* da *HMI*.



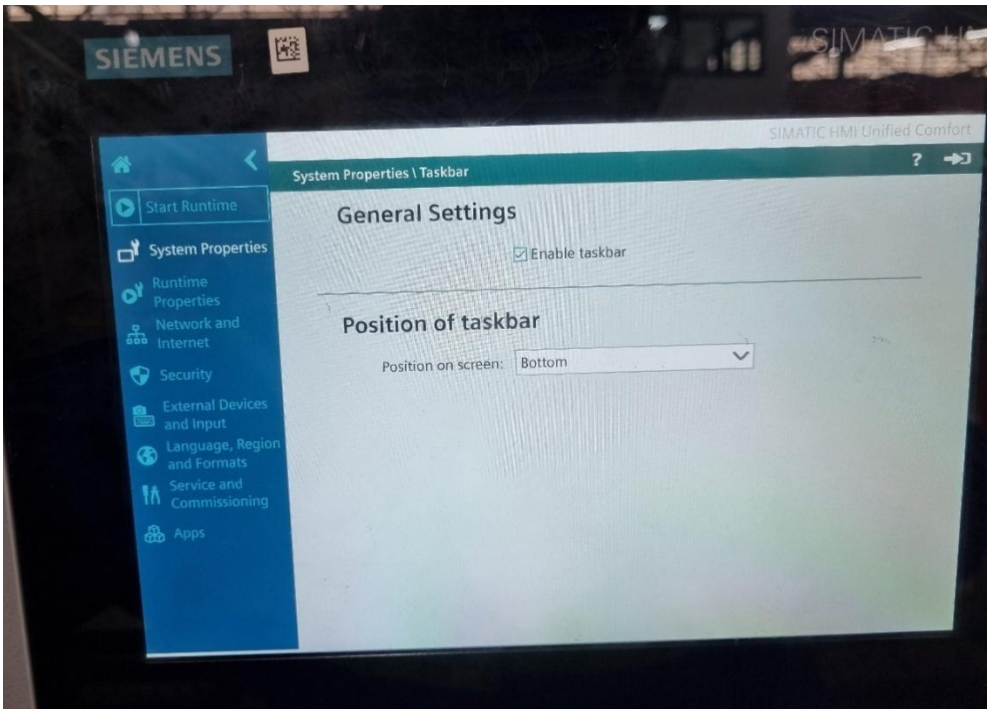
Fonte: Autoria própria.

Figura 183 - Janela de configuração de alarme de performance da *HMI*.



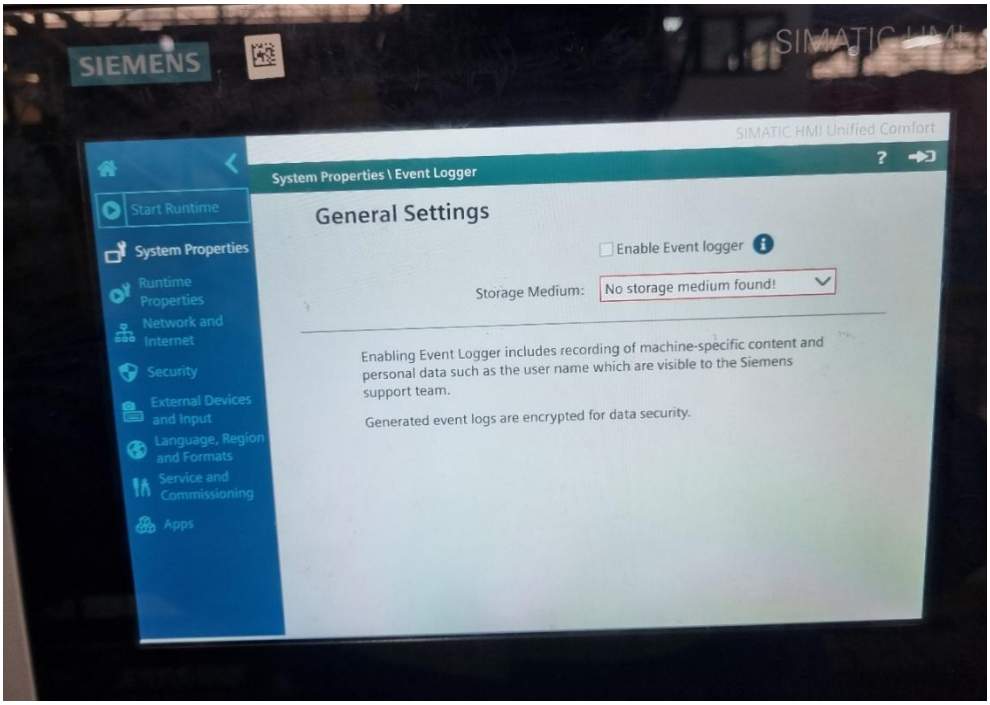
Fonte: Autoria própria.

Figura 184 - Janela de configuração dos botões flutuantes da *HMI*.



Fonte: Autoria própria.

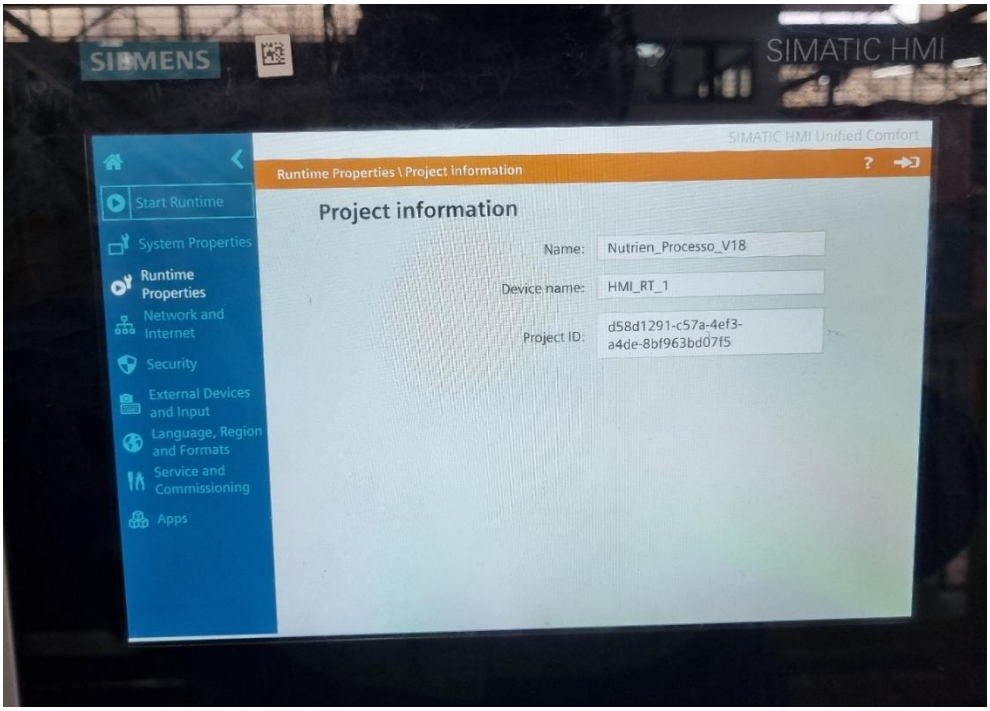
Figura 185 - Janela para exportar *logs* da *HMI*.



Fonte: Autoria própria.

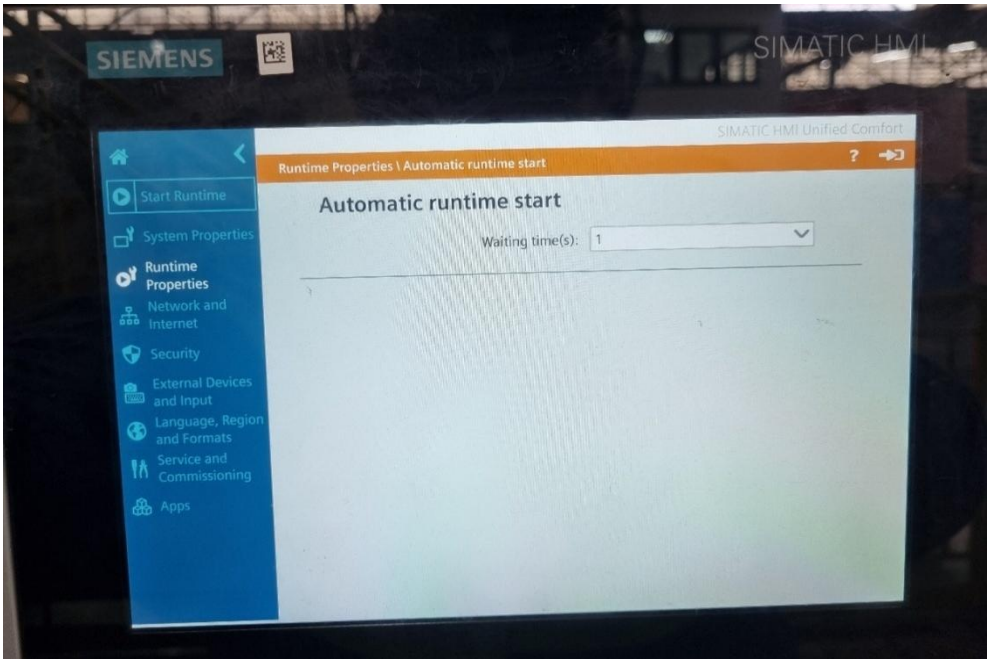
APÊNDICE B - Telas de visualização/configuração das propriedades da *Runtime*

Figura 186 - Janela de informações de projeto.



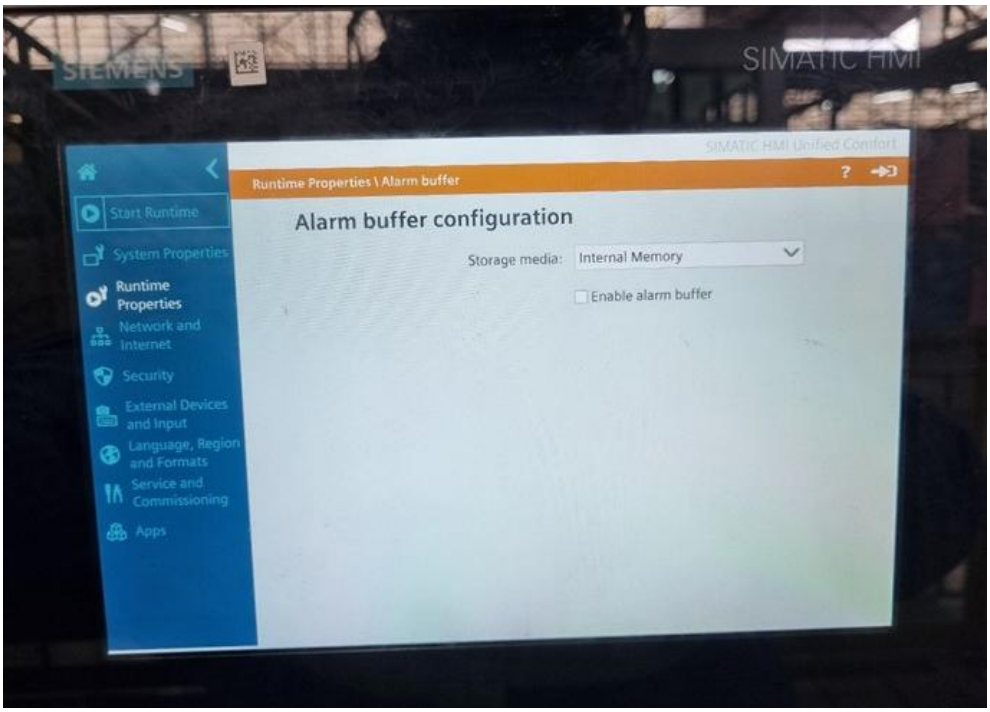
Fonte: Autoria própria.

Figura 187 - Janela de parâmetro do tempo de *AutoStart* do projeto.



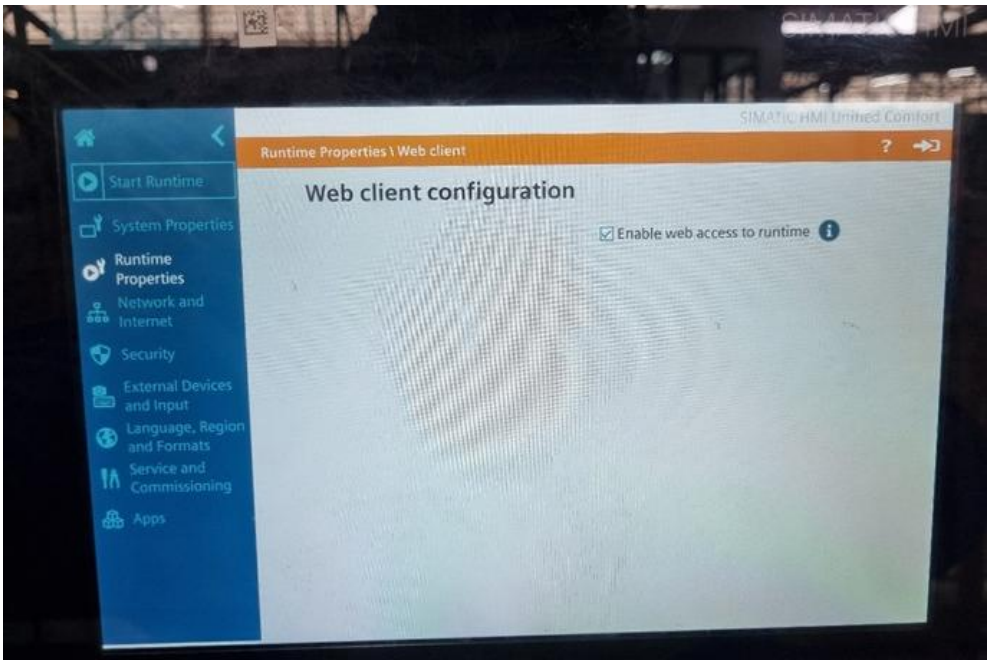
Fonte: Autoria própria.

Figura 188 - Janela de configuração do *buffer* de alarme.



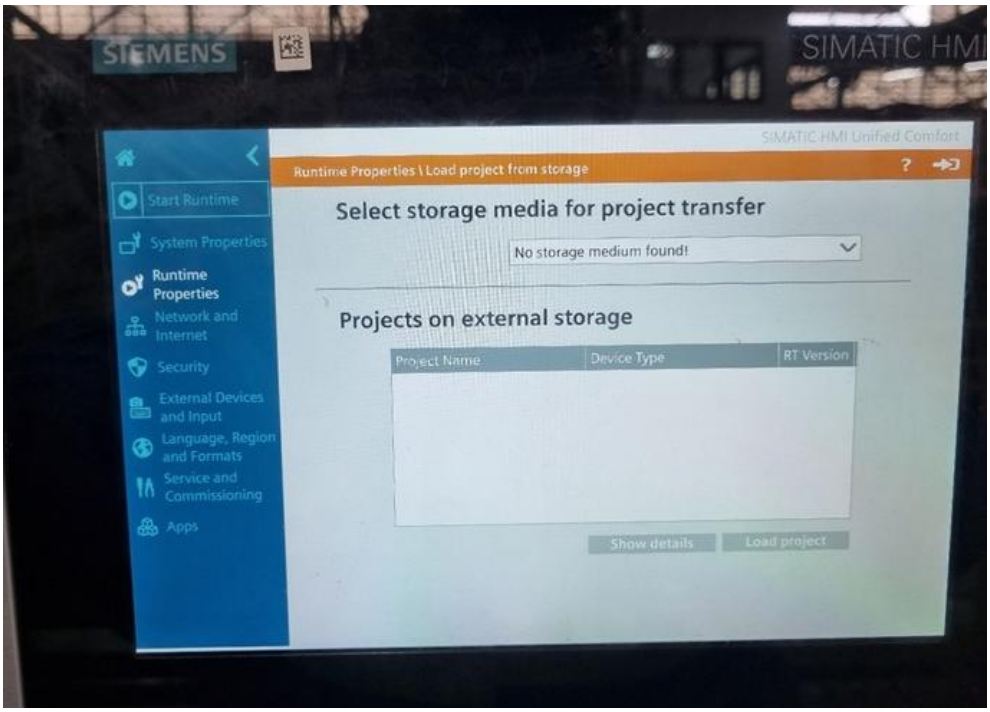
Fonte: Autoria própria.

Figura 189 - Janela do parâmetro de acesso *web*.



Fonte: Autoria própria.

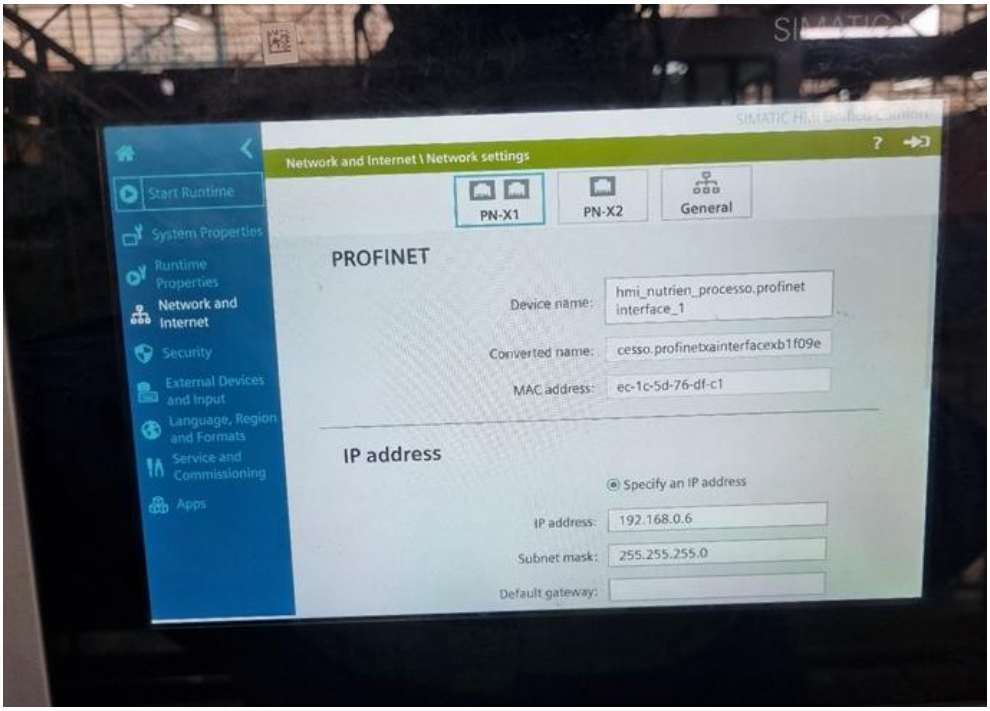
Figura 190 - Janela de carregamento de projetos a partir da memória externa.



Fonte: Autoria própria.

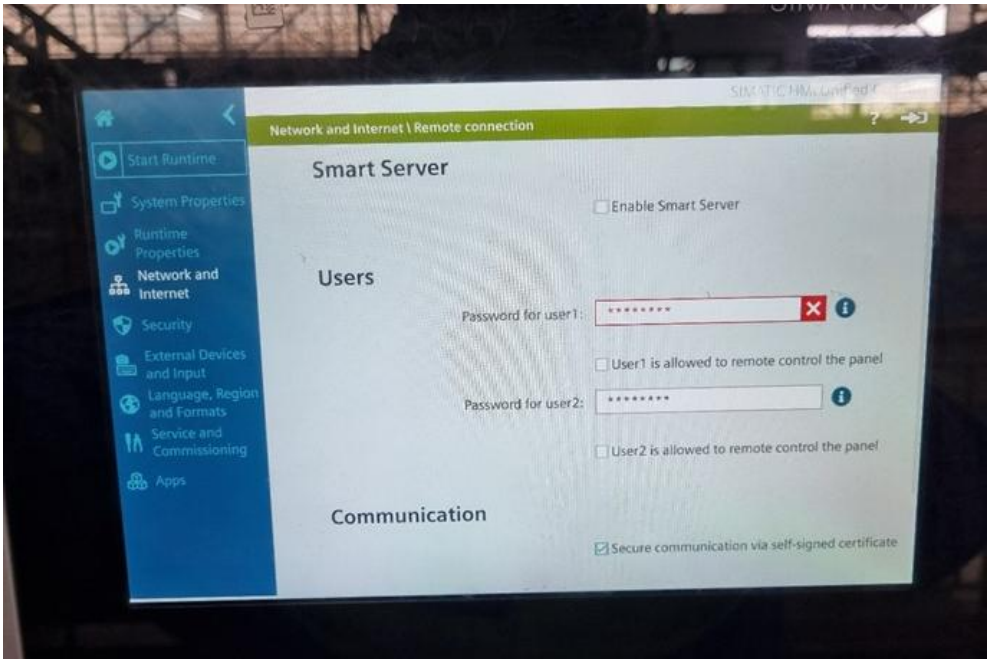
APÊNDICE C - Telas de configuração das propriedades de *Network and Internet*

Figura 191 - Janela de endereçamento das saídas *profinet*.



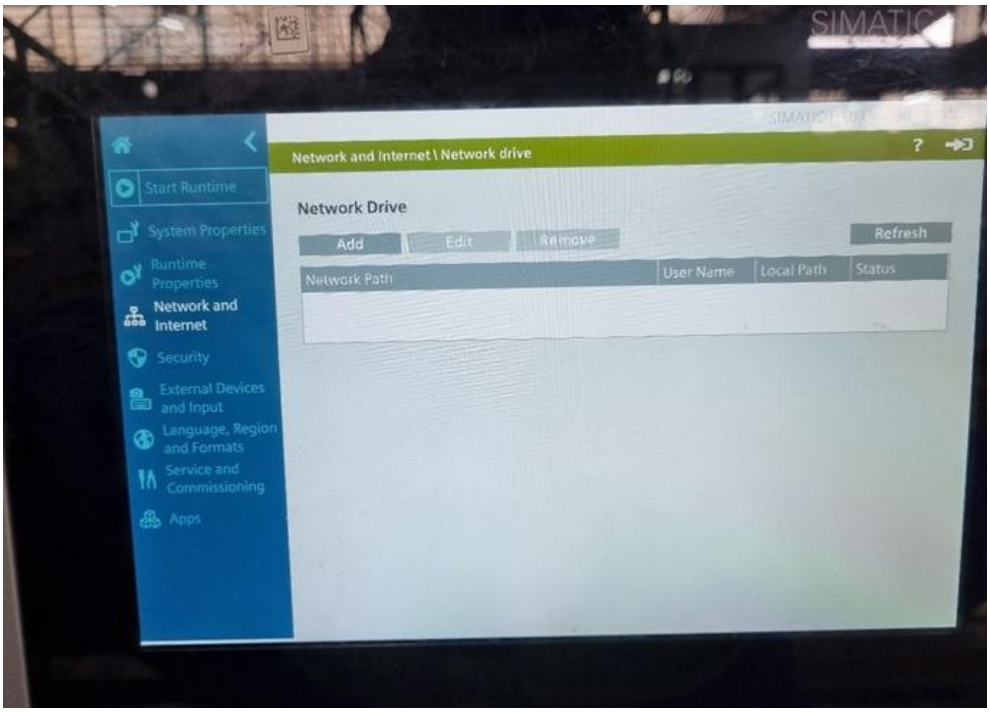
Fonte: Autoria própria.

Figura 192 - Janela de configuração do servidor de acesso remoto.



Fonte: Autoria própria.

Figura 193 - Janela de configuração de *drives* de comunicação.



Fonte: Autoria própria.

APÊNDICE D- Código das mensagens de estado na *faceplate* de comando das válvulas

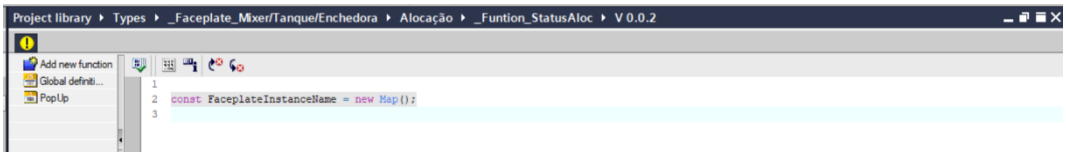
```
export function Text_box_4_Text_Trigger(item) {
   var value;
let tagTxt = Tags("Válvula.Status").Read();

switch (tagTxt) {
case 1:
  value = 'Segurança da Válvula\nDesabilitada';
  break;
case 5:
   value = 'Alarme Sensores Ativos\nAberto e Fechado';
  break;
case 10:
  value = 'Demora Pra Fechar\nSensor Aberto Ativo';
  break;
case 11:
  value = 'Alarme Demora\nPra Fechar';
  break;
case 15:
  value = 'Demora Pra Abrir\nSensor Fechado Ativo';
  break;
case 16:
  value = 'Alarme Demora\nPra Abrir';
  break;
```

```
        case 20:
          value = 'Falha Presente';
          break;
        case 21:
          value = 'Alarme';
          break;
        case 25:
          value = 'Abrindo\nVálvula';
          break;
        case 26:
          value = 'Válvula\nAberta';
          break;
        case 30:
          value = 'Fechando\nVálvula';
          break;
        case 31:
          value = 'Válvula\nFechada';
          break;
        default:
          value = 'Erro de leitura';
          break;
}
          return value;
}
```

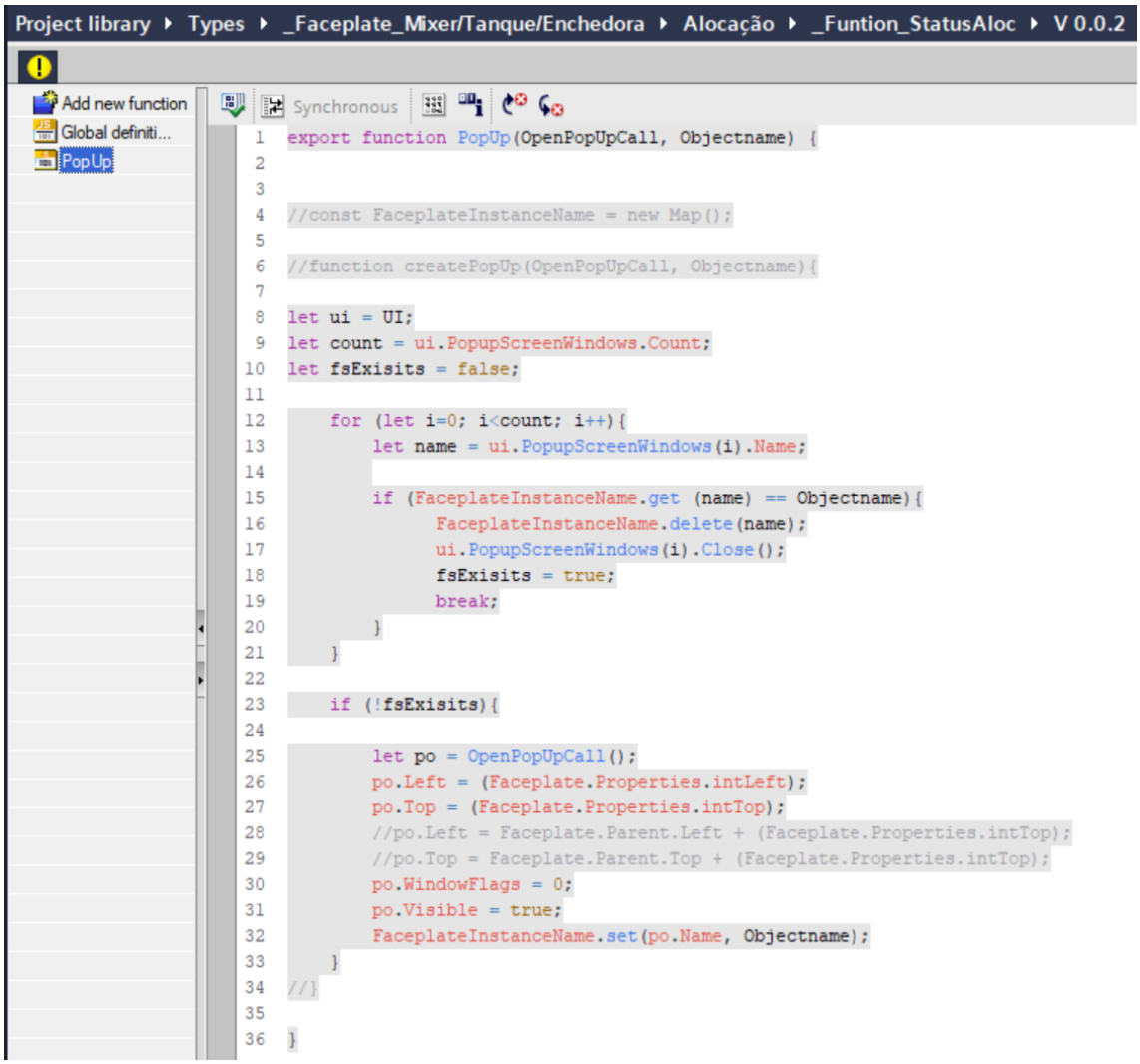
APÊNDICE E - Código de configuração de chamada de *faceplate* com parâmetro de posicionamento

Figura 194 - Código de instância do *faceplate* de comando.



Fonte: Autoria própria.

Figura 195 - Código de chamada do *faceplate* de comando.



Fonte: Autoria própria.

APÊNDICE F - Código das mensagens de cada passo da receita do *faceplate* de operação

```
export function DescAcompPasso_txt_Text_Trigger(item) {
  var value;

let tagAcompPasso = Tags("Slots.Acomp.Passo").Read();

  switch (tagAcompPasso) {
  case 0:
    value = 'DISPONÍVEL';
    break;
  case 5:
    value = 'AGUARDANDO START';
    break;
  case 10:
    value = 'ALOCANDO MIXER';
    break;
  case 20:
    value = 'ALOCANDO TANQUE';
    break;
  case 25:
    value = 'ALOCANDO ENCHEDORA';
    break;
  case 30:
```

```
            value = 'ALOCANDO SENSORES';
            break;
          case 40:
            value = 'ALOCANDO VÁLVULAS';
            break;
          case 50:
            value = 'AGUARDANDO CONFIRMAÇÃO DO PIG NA GARAGEM';
            break;
          case 60:
            value = 'VERIFICANDO O PIG NA POSIÇÃO';
            break;
          case 70:
            value = 'AGUARDANDO CONFIRMAÇÃO DE SENSORES\nDA PLACA DE
FLUXO POCISIONADOS';
            break;
          case 80:
            value = 'VERIFICANDO POSIÇÃO DOS SENSORES';
            break;
          case 90:
            value = 'INICIA A OPERAÇÃO';
            break;
          case 100:
            value = 'EM OPERAÇÃO'
            break;
          case 110:
```

```
      value = 'TEMPO PARA FINALIZAÇÃO DA OPERAÇÃO';
      break;
    case 120:
      value = 'TEMPO PARA FINALIZAÇÃO DA OPERAÇÃO';
      break;
    case 130:
      value = 'AGUARDANDO RESET PARA CONCLUIR FINALIZAÇÃO';
      break;
    default:
      value = 'DISPONÍVEL';
      break;
    }
    return value;
}
```

APÊNDICE G - Código das mensagens de cada passo do estágio do *PIG* no *faceplate* de operação

```
export function DescAcompEstagio_txt_Text_Trigger(item) {
   var value;
let tagAcompPasso = Tags("Slots.Acomp.Passo").Read();
let tagAcompEstagio = Tags("Slots.Acomp.Estagio").Read();
let tagSubRotina = Tags("Slots.Rota.Subtipo").Read();

switch (tagAcompEstagio) {
   case 0:
    if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
       value = 'PIG na Garagem, Pronto para Lançamento';
    }
    if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
       value = 'PIG na Garagem, Pronto para Lançamento';
    }
    break;
   case 10:
    if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
       value = 'Libera PIG na Tubulação de Lançamento';
    }
    if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
       value = 'Libera PIG na Tubulação de Lançamento';
    }
    break;
```

```
case 11:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = '';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Introdução da Bala na Tubulação de Lançamento';
  }
  break;
case 20:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Pressuriza Tubulação de envio do PIG';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Pressuriza Tubulação de envio do PIG';
  }
  break;
case 30:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'PIG à caminho do receptor';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'PIG à caminho do receptor';
  }
  break;
```

```
case 40:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Despressuriza Tubulação de envio do PIG e\nmantém fechada a Válvula de abastecimento do tanque destino';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Despressuriza Tubulação de envio do PIG e\nmantém fechada a Válvula de abastecimento do tanque destino';
  }
  break;
case 50:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Libera PIG na Tubulação de Retorno';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Libera PIG na Tubulação de Retorno';
  }
  break;
case 55:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Tempo de despressurização do trecho de envio';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = '';
  }
```

```
  break;
case 60:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Pressuriza Tubulação de Retorno do PIG';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Pressuriza Tubulação de Retorno do PIG';
  }
  break;
case 65:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = '';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'PIG à caminho do lançador';
  }
  break;
case 70:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'PIG chegando no lançador';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'PIG chegando no lançador';
  }
```

```
  break;
case 80:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Despressuriza Tubulação de Retorno';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Introdução do PIG na garagem';
  }
  break;
case 85:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Tempo de despressurização do trecho';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Despressurização do retorno';
  }
  break;
case 90:
  if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
     value = 'Finalizando do Ciclo do PIG';
  }
  if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
     value = 'Finalizando do Ciclo do PIG';
  }
```

```
      break;
    case 95:
      if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
         value = 'Remover bala';
      }
      if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
         value = '';
      }
      break;
    case 100:
      if ((tagSubRotina == 10) && (tagAcompPasso == 100)) {
         value = 'Finalização do Ciclo do PIG';
      }
      if ((tagSubRotina == 20) && (tagAcompPasso == 100)) {
         value = 'Finalização do Ciclo do PIG';
      }
      break;
    default:
      value = 'DISPONÍVEL';
      break;
  }
    return value;
  }
```

APÊNDICE H - Código das mensagens de falha no *faceplate* de operação

```
export function DescFalha_txt_Text_Trigger(item) {
   var value;

let tagAcompFalha = Tags("Slots.Acomp.DescFalha").Read();
let tagAviso = Tags("Slots.Acomp.DescAviso").Read();
let tagMixerFalha = Tags("Slots.RespCond.Mixer.Tag").Read();
let tagTanqueFalha = Tags("Slots.RespCond.Tanque.Tag").Read();
let tagEnchedoraFalha = Tags("Slots.RespCond.Enchedora.Tag").Read();
let tagEDsFalha = Tags("Slots.RespCond.Eds.Tag").Read();
let tagXVsFalha = Tags("Slots.RespCond.Xvs.Tag").Read();
let tagPos_EDs01Falha = Tags("Slots.RespCond.VerPos_Eds01.Tag").Read();
let tagPos_EDs02Falha = Tags("Slots.RespCond.VerPos_Eds02.Tag").Read();

let tagAcompEnvPig_Min = Tags("Slots.Rota.Parametros[0]").Read();
let tagAcompEnvPig_Seg = Tags("Slots.Rota.Parametros[1]").Read();
let tagAcompRetPig_Min = Tags("Slots.Rota.Parametros[2]").Read();
let tagAcompRetPig_Seg = Tags("Slots.Rota.Parametros[3]").Read();

   switch (tagAcompFalha) {

   case 0:
```

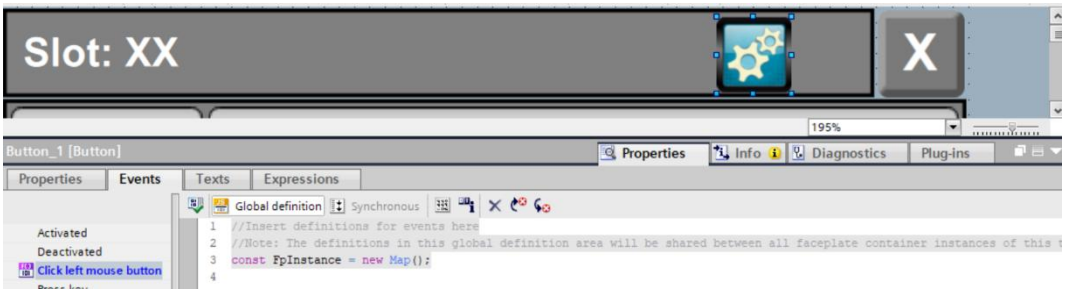
```
            value = 'Tempo de Envio: ' + tagAcompEnvPig_Min + '`' + tagAcompEnvPig_Seg
+ '`` - ' +
                    'Tempo de Retorno: '+ tagAcompRetPig_Min + '`' + tagAcompRetPig_Seg +
'`` ';
              break;
            case 5:
              value = 'Rota encerrada pelo Operador';
              break;
            case 10:
              value = 'Falha na verificação do Mixer: ' + tagMixerFalha;
              break;
            case 20:
              value = 'Falha na verificação do Tanque: ' + tagTanqueFalha;
              break;
            case 25:
              value = 'Falha na verificação da Enchedora: ' + tagEnchedoraFalha;
              break;
            case 30:
              value = 'Falha na verificação da Entrada Digitail: ' + tagEDsFalha;
              break;
            case 40:
              value = 'Falha na verificação da Válvula: ' + tagXVsFalha;
              break;
            /*case 50:
              value = '';
```

```
      break;*/
    case 60:
      value = 'Falha na verificação do Sensor do PIG na Garagem: ' + tagPos_EDs01Falha;
      break;
    /*case 70:
      value = '';*/
      break;
    case 80:
      value = 'Falha na verificação do Sensore da Placa de Fluxo: ' + tagPos_EDs02Falha;
      break;
    /*case 85:
      value = '';
      break;*/
    case 90:
      value = 'Detecção de queda de energia durante execução da rota';
      break;
    default:

      value = 'DISPONÍVEL'
      break;
    }
    return value;
}
```

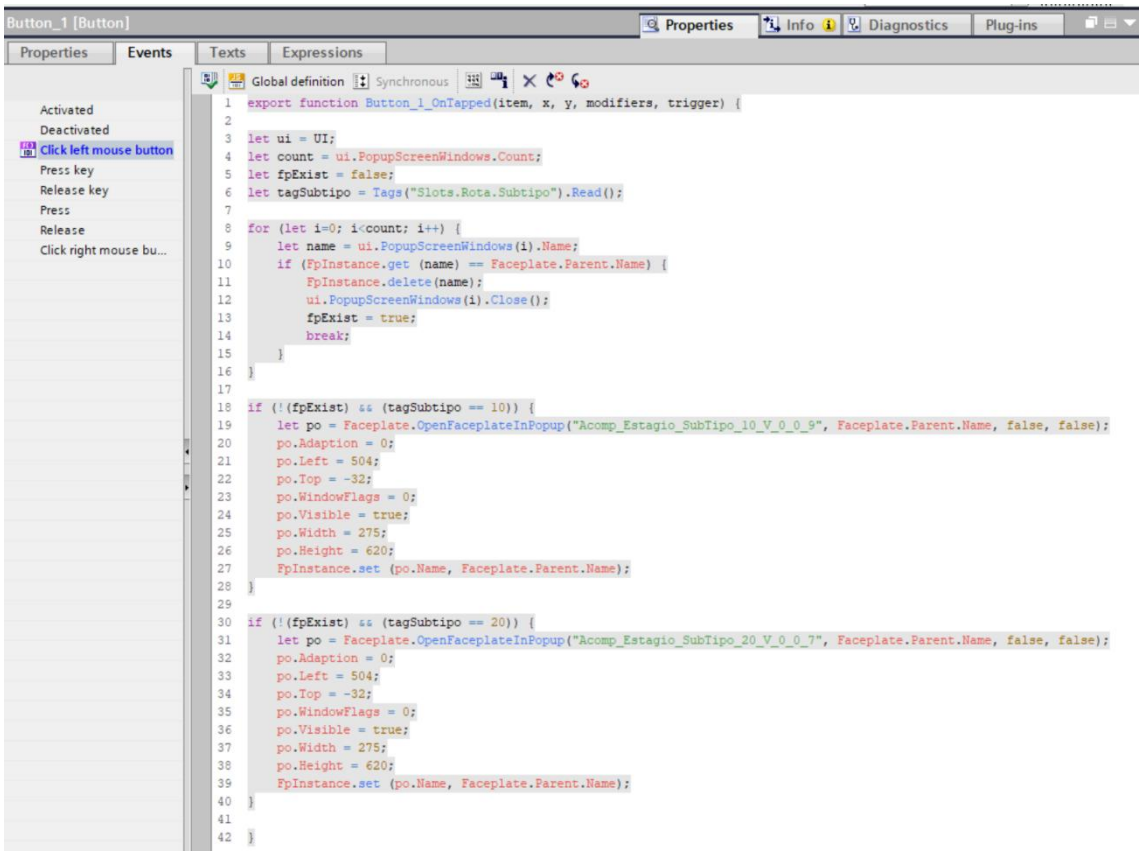
APÊNDICE I - Código de configuração de chamada de *faceplate* do fluxograma dos passos da receita em operação

Figura 196 - Código de instância do *faceplate* do fluxograma.



Fonte: Autoria própria.

Figura 197 - Código de chamada do *faceplate* do fluxograma.



Fonte: Autoria própria.