TIA PORTAL OPENNESS

(Funções)

Índice

[1. Estrutura 4](#_Toc30414692)

[1.1. TIAPortalOpennessDemo 5](#_Toc30414693)

[1.2. TiaOpennessHelper 6](#_Toc30414694)

[2. Organização 8](#_Toc30414695)

[3. Conexão ao TIA Portal 9](#_Toc30414696)

[4. Templates de código 10](#_Toc30414697)

[5. Funcionalidade “*Main folder files list*” 11](#_Toc30414698)

[6. Funcionalidade “*PLC DB Generator*” 12](#_Toc30414699)

[7. Funcionalidade “*PLC Tags*” 14](#_Toc30414700)

[8. Funcionalidade “*Generate Symbolic*” 15](#_Toc30414701)

[8.1. Select Robot 15](#_Toc30414702)

[8.2. Generate Symbolic 16](#_Toc30414703)

[9. Funcionalidade “*Sequence Generator*” 18](#_Toc30414704)

[9.1. Create Excel 19](#_Toc30414705)

[10. Funcionalidade “*Rename PLC*” 20](#_Toc30414706)

[11. Funcionalidade “*RobotList*” 22](#_Toc30414707)

[12. Funcionalidade “*Hardware Generator*” 23](#_Toc30414708)

[13. Classes importantes 24](#_Toc30414709)

[13.1. XLInterface 24](#_Toc30414710)

[13.2. PdfReader 26](#_Toc30414711)

[13.3. XmlParser 27](#_Toc30414712)

[13.4. CacheManager 28](#_Toc30414713)

[14. Notas finais 29](#_Toc30414714)

Índice de Imagens

[Figura 1 – Estrutura da Solution "TiaPortalOpennessDemo" 4](#_Toc30411571)

[Figura 2 - Estrutura do projeto "TIAPortalOpennessDemo" 5](#_Toc30411572)

[Figura 3 - Estrutura do projecto "TiaOpennessHelper" 7](#_Toc30411573)

[Figura 4 - Arquitetura MVVM 8](#_Toc30411574)

[Figura 5 - Função "ConnectToTia" 9](#_Toc30411575)

[Figura 6 - Função "LoadProjectTreeView" 9](#_Toc30411576)

[Figura 7 - Estrutura de pastas do programa 10](#_Toc30411577)

[Figura 8 - Pasta "Templates" na versão completa do programa 10](#_Toc30411578)

[Figura 9 - Funções dos botões da “Main folder files list” 11](#_Toc30411579)

[Figura 10 - Funções dos botões da “DBMaker” 13](#_Toc30411580)

[Figura 11 - Funções dos botões da view “PLC\_Taps” 14](#_Toc30411581)

[Figura 12 - Funções dos botões da view "SelectRobot" 15](#_Toc30411582)

[Figura 13 - Função "AddButtons" 15](#_Toc30411583)

[Figura 14 - Função "NewRobot" 17](#_Toc30411584)

[Figura 15 - Funções dos botões da view "TreeViewManager" 18](#_Toc30411585)

[Figura 16 - Funções dos botões da view "ExcelAsker" 19](#_Toc30411586)

[Figura 17 - Funções dos botões da view "RenamePLC" 20](#_Toc30411587)

[Figura 18 - Função "Rename" 21](#_Toc30411588)

[Figura 19 - Função do botão "RobotList" 22](#_Toc30411589)

[Figura 20 - Funções dos botões da view "HardwareGenerator" 23](#_Toc30411590)

[Figura 21 - Função "ExcelToMatrix" 24](#_Toc30411591)

[Figura 22 - Funções responsáveis por identificar o tipo de ficheiro 24](#_Toc30411592)

[Figura 23 - Funções de leitura de ficheiros "Symbolic" e "NetworkList" 25](#_Toc30411593)

[Figura 24 - Funções classe "PdfReader" 26](#_Toc30411594)

[Figura 25 - Funções de criação de ficheiros XML do tipo "PLCTag" 27](#_Toc30411595)

[Figura 26 - Funções da classe "CacheManager" 28](#_Toc30411596)

# Estrutura

Esta aplicação é composta por uma “*Solution*” com o nome “*TIAPortalOpennessDemo*” e dentro desta existem dois projetos com os nomes “*TiaOpennessHelper*” e “*TIAPortalOpennessDemo*”. É no projeto “*TiaOpennessHelper*” que estão presentes as principais classes que servem como auxílio ao projeto “*TIAPortalOpennessDemo*”.

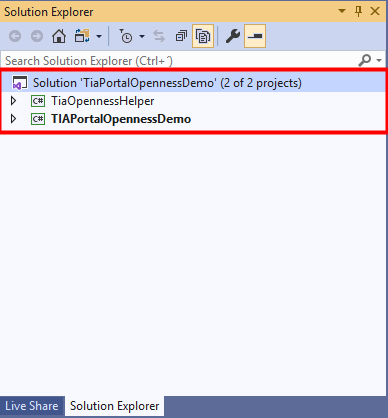


Figura 1 – Estrutura da Solution "TiaPortalOpennessDemo"

## TIAPortalOpennessDemo

Este projeto é composto por seis pastas: “*Commands”*, “*Images”*, “*Services”*, “*Utilities”*, “*ViewModels”* e “*Views”*.

Foi acrescentado conteúdo às pastas:

* **Views**: onde estão criadas as janelas da aplicação, foram acrescentadas novas janelas;
* **ViewModels**: onde se encontram as classes que comandam os elementos de uma determinada janela, foram criadas novas classes para as novas janelas criadas e um ficheiro *schema* para verificar se o ficheiro XML “*Config*”, importado na funcionalidade “*Rename PLC*”, é ou não válido;
* **Images**: onde estão as imagens utilizadas pelo programa, foram importadas novas imagens.

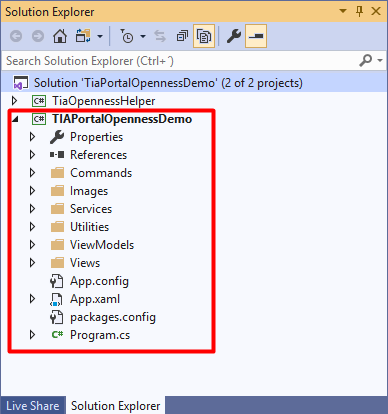


Figura 2 - Estrutura do projeto "TIAPortalOpennessDemo"

## TiaOpennessHelper

Este projeto é composto por oito pastas: “*Enums*”, “*SequenceGenerator*”, “*Models*”, “*PLCDbGenerator*”, “*SCLParser*”, “Utils”, “*VWSymbolic*” e “*XMLParser*”.

Foi acrescentado ao projeto as pastas:

* **SequenceGenerator**: Contém todas as classes/janelas utilizadas pela funcionalidade “*Sequence Generator*”;
* **PLCDbGenerator**: Onde estão as classes, e a janela, utilizadas pela funcionalidade “*PLC DB Generator”;*
* **VWSymbolic**: Composta por classes que servem de auxílio à funcionalidade “*Generate Symbolic*” e um ficheiro *schema*;

Para além das classes criadas nas novas pastas, foram ainda criadas outras para servir de auxílio nas diversas funcionalidades do programa:

* Na pasta “*Utils*” foram criadas as classes:
  + **CacheManager**: Utilizada pela funcionalidade “*Main folder files list*” onde estão as funções responsáveis pela manipulação de informação em *cache*;
  + **FolderInfo**: Utilizada como uma estrutura organizada de ficheiros permitindo guardar o caminho do ficheiro e o seu nome;
  + **uFindVisualChild**: Usada somente pela funcionalidade “*Generate Symbolic*”, para facilitar a busca por elementos selecionados do tipo “*Combobox*” nas tecnologias do robô.
* Na pasta “*XMLParser*” foi criada uma pasta com o nome “*XMLEditor*” com três classes que servem de ajuda para a criação do *hardware* do robô pela funcionalidade “*Hardware Generator*”.

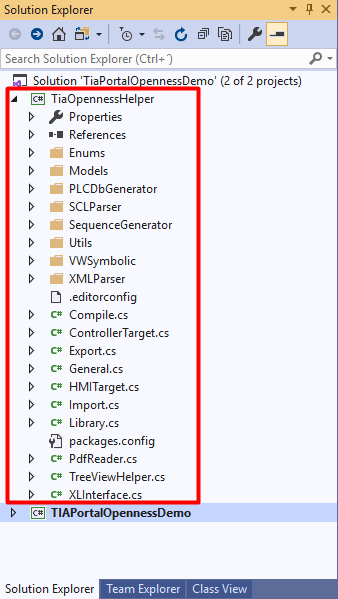


Figura 3 - Estrutura do projecto "TiaOpennessHelper"

# Organização

Sendo esta aplicação programada utilizando a arquitetura “*Model-View-ViewModel”*, ou “MVVM”, é necessário que todas as janelas tenham uma classe que controle os teus componentes e as suas funcionalidades.

Na pasta “*ViewModels*”, do projecto “*TIAPortalOpennessDemo*”, podemos encontrar as classes que controlam algumas das janelas presentes na pasta “*Views*”. Essas classes designam-se por “*ViewModels*”.

Algumas janelas, ou *views*, não têm associado um *viewmodel* próprio. Temos o caso das janelas “*CreateFolderDialog*”, “*FileBrowserControl*”, “*ImportCaxControl*”, “*Overlay*” e “*SettingsView*”, em que todas elas têm o *viewmodel* “*MainWindowViewModel*” como base.

As janelas “*PLC\_Taps*”, “*RobotView*”, “*TreeViewManager*”, “*ExcelAsker*”, “*DBMaker*” e “*OptionsRobotView*” não têm um *viewmodel* associado e, assim sendo, toda a sua programação é feita no código fonte, ou *sourcecode*, da *view*.

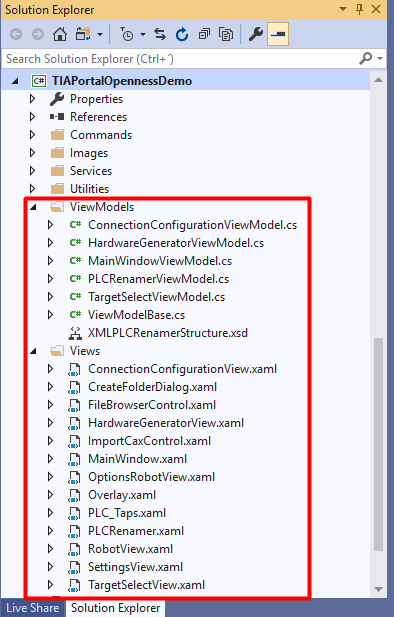


Figura 4 - Arquitetura MVVM

# Conexão ao TIA Portal

A estrutura de ficheiros carregada do TIA Portal para o TIA Portal Openness foi modificada.

Ao invés de serem carregados todos os elementos que constituem o projeto do TIA Portal, são apenas carregados os que serão utilizados pelas funcionalidades do TIA Portal Openness. Com esta modificação, existe uma melhoria significativa no tempo que leva a conexão a ser feita.

Os elementos carregados são organizados hierarquicamente com a ajuda de uma *treeview*. Esta *treeview* é gerada pela função *“LoadProjectTreeView*”, presente na classe “*MainWindowViewModel*”.

A conexão ao TIA Portal é feita pela função “*ConnectToTIA*”, também presente na classe “*MainWindowViewModel*”. Esta classe recebe como parâmetro de entrada uma variável do tipo booleano com o objetivo de indicar se a conexão irá ser feita exclusivamente para a funcionalidade “*Rename PLC*”. Se a conexão for feita apenas para renomear o PLC, então a *treeview* gerada quando se faz uma ligação ao TIA Portal, não será criada.

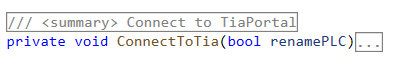


Figura 5 - Função "ConnectToTia"



Figura 6 - Função "LoadProjectTreeView"

# Templates de código

Para facilitar a criação de ficheiros XML e Excel por parte do programa, foram criados vários *templates*. Estes *templates* devem sempre acompanhar a pasta do programa para que tudo funcione corretamente.

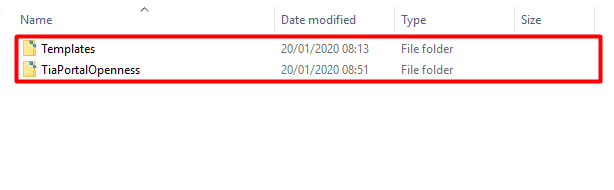


Figura 7 - Estrutura de pastas do programa

Na versão completa do programa, que inclui o projeto em *Visual Studio*, a pasta “*Templates*” encontra-se na pasta “*TiaPortalOpennessDemo*” -> “*bin*”.

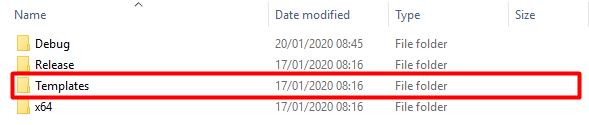


Figura 8 - Pasta "Templates" na versão completa do programa

# Funcionalidade “*Main folder files list*”

Todas as ações desta funcionalidade estão programadas na *viewmodel* “*MainWindowViewModel*”.

Existem quatro possíveis ações que podem ser realizadas nesta funcionalidade, todas elas são ativas por meio de botões.

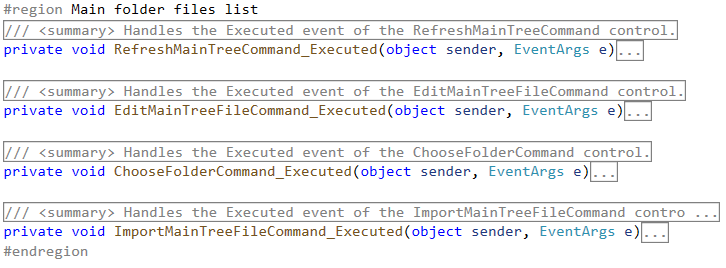


Figura 9 - Funções dos botões da “Main folder files list”

# Funcionalidade “*PLC DB Generator*”

Todas as ações desta funcionalidade estão programadas no *sourcecode* da *view* “*DBMaker*”, dentro da pasta “*PLCDbGenerator*”.

Existe um grande número de variáveis criadas nesta *view*, isto porque muitas delas não são utilizadas apenas localmente mas também por outras classes dentro da pasta “*PLCDbGenerator*”.

Dentro da pasta “*PLCDbGenerator*” podemos encontrar as classes:

* **EngAssist:** Funciona como uma estrutura de dados para armazenar as informações referentes à *Worksheet* “*EngAssist*”;
* **Frag:** Funciona como uma estrutura de dados para armazenar as informações do robô (perfil, estação, função) e o nome da *Worksheet* utilizada para recolher estas informações da folha de Excel “*Schnittstelle*”;
* **NetworkDBMaker:** Onde estão armazenadas as funções necessárias para a criação de todas as bases de dados (ficheiros XML) geradas por esta funcionalidade;
* **PLC\_Tap:** Funciona como uma estrutura de dados para armazenar as informações de uma *tag*;
* **ReplaceActions:** Funciona como estrutura de dados para organizar quais as ações que serão substituídas em cada *Worksheet* do ficheiro “*PLC DB*”, guardando qual o texto a substituir e o texto pelo qual será substituído;
* **UserConfig:** Funciona como estrutura de dados para armazenar o nome e o valor de cada entrada na *Worksheet* “*User Config*” do ficheiro “*PLC DB*”;
* **Variable:** Funciona como estrutura de dados para armazenar as variáveis presentes em cada *Worksheet* do ficheiro “*PLC DB*”, guardando o nome da variável, o tipo, o comentário e a ação.

Existem três possíveis ações que podem ser realizadas nesta funcionalidade, todas elas são ativas por meio de botões.

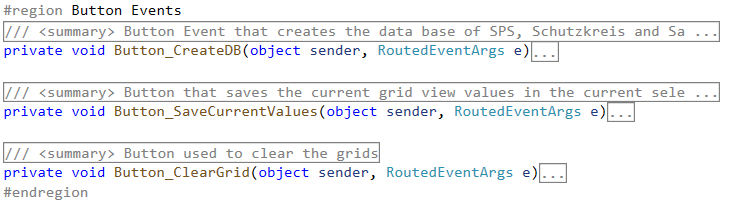


Figura 10 - Funções dos botões da “DBMaker”

# Funcionalidade “*PLC Tags*”

Todas as ações desta funcionalidade estão programadas no *sourcecode* da *view* “*PLC\_Taps*”, dentro da pasta “*Views*”.

Existem três possíveis ações que podem ser realizadas nesta funcionalidade, todas elas são ativas por meio de botões.

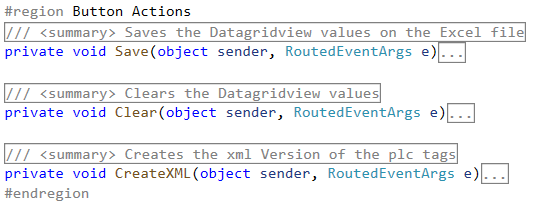


Figura 11 - Funções dos botões da view “PLC\_Taps”

# Funcionalidade “*Generate Symbolic*”

## Select Robot

Todas as ações desta funcionalidade estão programadas no *sourcecode* da *view* “*OptionsRobotView*”, dentro da pasta “*Views*”.

Todas as informações retiradas da folha Excel, são armazenadas em listas e enviadas para esta *view*.

Existem duas possíveis ações que podem ser realizadas nesta funcionalidade: Criar um novo robô ou selecionar um já criado na folha Excel.

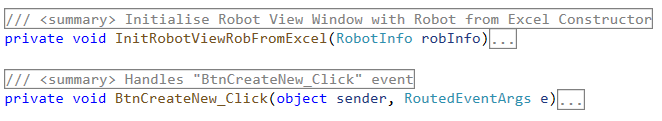


Figura 12 - Funções dos botões da view "SelectRobot"

Os botões são criado dinamicamente pela função “*AddButtons*” conforme a quantidade de robôs presente na folha Excel selecionada. Esta função trata de formatar corretamente os botões e apresenta-os no *design* da *view*.



Figura 13 - Função "AddButtons"

## Generate Symbolic

Todas as ações desta funcionalidade estão programadas no *sourcecode* da *view* “*OptionsRobotView*”, dentro da pasta “*RobotView*”.

Esta funcionalidade utiliza as classes dentro da pasta “*VWSymbolic*” como auxílio na execução das suas ações. Esta pasta é composta por um ficheiro *schema,* para verificar se a formatação de um ficheiro XML do tipo “*symbolic”,* aberto pela “*Main folder files list*”, é ou não válido, e pelas classes:

* **Robot:** Uma classe estática com o objetivo de preservar os valores inseridos nas listas utilizadas para gerar o simbólico de um robô. Os valores destas listas são preenchidos na *viewmodel* “*MainWindowViewModel*”, dentro da função “*LoadSymbolic*”;
* **RobotBase:** Funciona como estrutura de dados para armazenar as informações de uma *tag* presente na *Worksheet* “*Grund*” de um ficheiro de Excel do tipo “*Symbolic*”. Esta estrutura de dados guarda o simbólico da *tag*, o seu tipo, o endereço e o comentário;
* **RobotInfo:** Funciona como estrutura de dados para armazenar as informações de um simbólico já criado. Esta estrutura de dados guarda o nome do robô, o tipo de segurança (“*Range Monitoring*” ou “*Operation*”), o endereço inicial, as tecnologias utilizadas e o seu tipo (“*Basic Slave*” ou “*Laser Slave*”);
* **RobotSafeOperation:** Funciona como estrutura de dados para armazenar as informações de uma *tag* presente na *Worksheet* “*Rob Safe Operation*” de um ficheiro de Excel do tipo “*Symbolic*”. Esta estrutura de dados guarda o simbólico da *tag*, o seu tipo de dados, o endereço e o comentário;
* **RobotSafeRangeMonitoring:** Funciona como estrutura de dados para armazenar as informações de uma *tag* presente na *Worksheet* “*Rob Safe Range Monitoring*” de um ficheiro de Excel do tipo “*Symbolic*”. Esta estrutura de dados guarda o simbólico da *tag*, o seu tipo de dados, o endereço e o comentário;
* **RobotTecnologie:** Funciona como estrutura de dados para armazenar as informações de uma *tag* relacionada à tecnologia de um robô. Esta estrutura de dados guarda o nome da tecnologia, o número do FB, o tipo de segurança (“*Range Monitoring*” ou “*Operation*”), o simbólico, o tipo de dados, o endereço e o comentário;
* **SymbolicManager:** Utilizada como auxilio na execução das ações da *view* “*RobotView*”. É responsável por gerar os ficheiros XML dos simbólicos e importar os mesmos para o TIA Portal.

O simbólico é gerado pela função “*New Robot*”, presente na classe “*SymbolicManager*”, chamada depois de clicado o botão “*Create Robot*” na *view*.



Figura 14 - Função "NewRobot"

# Funcionalidade “*Sequence Generator*”

Todas as ações desta funcionalidade estão programadas no *sourcecode* da *view* “*TreeViewManager*”, dentro da pasta “*SequenceGenerator*”.

Nesta pasta podemos encontrar as classes:

* **ExcelManager:** Contém funções utilizadas maioritariamente na manipulação de ficheiros Excel do tipo “*Sequence*”;
* **GrafcetManager:** É composta por funções utilizadas para preparar o XML do *Graphcet*
* **NetworkManager:** É composta por funções utilizadas para gerar o XML do *Graphcet*;
* **Step:** Funciona como estrutura de dados para armazenar as informações de uma *step* do ficheiro Excel do tipo “*Sequence*”;
* **StepHandler:** Funciona como estrutura de dados para organizar e facilitar a criação do ficheiro XML gerado pela funcionalidade.
* **WorkSheet:** Funciona como estrutura de dados para facilitar a leitura de um ficheiro do tipo “*Sequence*”. Esta estrutura é responsável por guardar o nome de uma *Worksheet* e as steps criadas na mesma.

Existem quatro possíveis ações que podem ser realizadas nesta funcionalidade, todas elas são ativas por meio de botões.

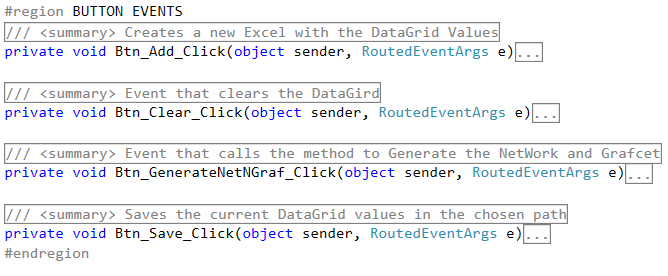


Figura 15 - Funções dos botões da view "TreeViewManager"

## Create Excel

Todas as ações desta funcionalidade estão programadas no *sourcecode* da *view* “*ExcelAsker*”, dentro da pasta “*SequenceGenerator*”.

Existem duas possíveis ações que podem ser realizadas nesta funcionalidade: “*Create”* e “*Cancel*”.

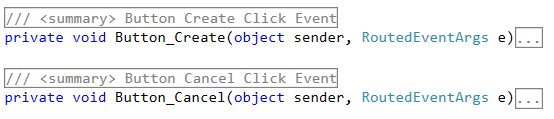


Figura 16 - Funções dos botões da view "ExcelAsker"

# Funcionalidade “*Rename PLC*”

Todas as ações desta funcionalidade estão programadas no *viewmodel* “*PLCRenameViewModel*”, dentro da pasta “*ViewModels*”.

Existem quatro possíveis ações que podem ser realizadas nesta funcionalidade, todas elas são ativas por meio de botões.

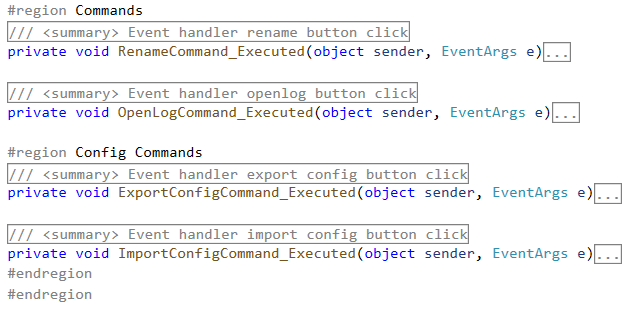


Figura 17 - Funções dos botões da view "RenamePLC"

A principal função desta funcionalidade é a “*RenamePLC*”, chamada quando clicado o botão “*Rename*”. Esta função é responsável por alterar os IP’s, exportar os elementos, eliminar os elementos do TIA Portal, editar o XML, renomear o *hardware* e importar os ficheiros alterados.

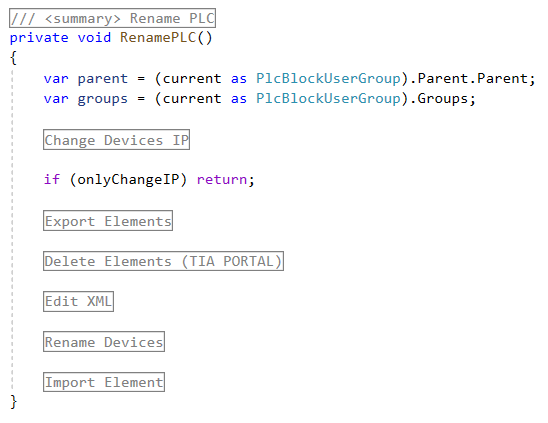


Figura 18 - Função "Rename"

# Funcionalidade “*RobotList*”

Todas as ações desta funcionalidade estão programadas no *viewmodel* “*MainWindowViewModel*”, dentro da pasta “*ViewModels*”.

Esta funcionalidade é acionada através do botão “*RobotList*”, que, quando clicado, utiliza a função “*GenerateRobotListCommand\_Executed*” que trata de gerar os ficheiros XML dos robôs da folha de Excel “*Schnittstelle*”.



Figura 19 - Função do botão "RobotList"

# Funcionalidade “*Hardware Generator*”

Todas as ações desta funcionalidade estão programadas no *viewmodel* “*HardwareGeneratorViewModel*”, dentro da pasta “*ViewModels*”.

Esta funcionalidade utiliza as classes dentro da pasta “*XMLParser*” -> “*XMLEditor*” como auxílio para gerar o *hardware* do PLC.

Existem três possíveis ações que podem ser realizadas nesta funcionalidade, todas elas são ativas por meio de botões.

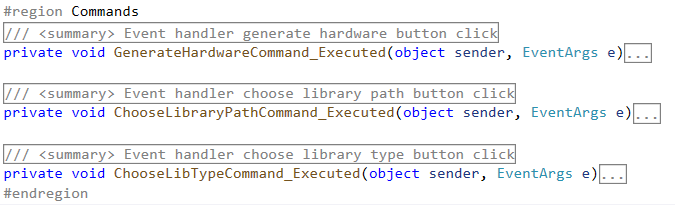


Figura 20 - Funções dos botões da view "HardwareGenerator"

# Classes importantes

## XLInterface

A classe “*XLInterface*” encontra-se dentro do projecto “*TiaOpennessHelper*” e é onde estão grande parte das funções de manipulação de ficheiros Excel.

Para aumentar a rapidez de leitura de cada *worksheet*, a informação contida na mesma, é guardada numa matriz. Esta conversão é feita pela função “*ExcelToMatrix*” que recebe como parâmetro de entrada uma *worksheet*.



Figura 21 - Função "ExcelToMatrix"

É também nesta classe onde se encontram as funções responsáveis por identificar um determinado tipo de ficheiro.

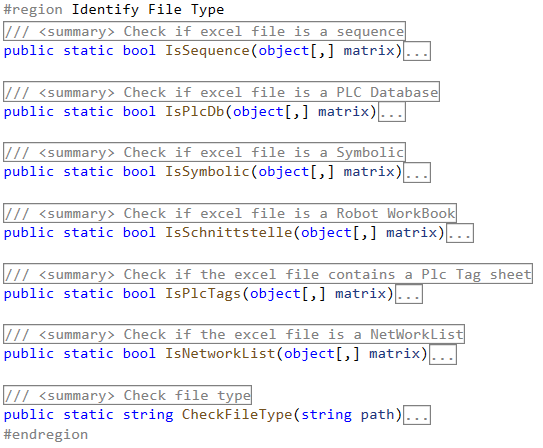


Figura 22 - Funções responsáveis por identificar o tipo de ficheiro

Esta classe também contém funções de manipulação de ficheiros do tipo “*Symbolic*” e “*NetworkList*”.

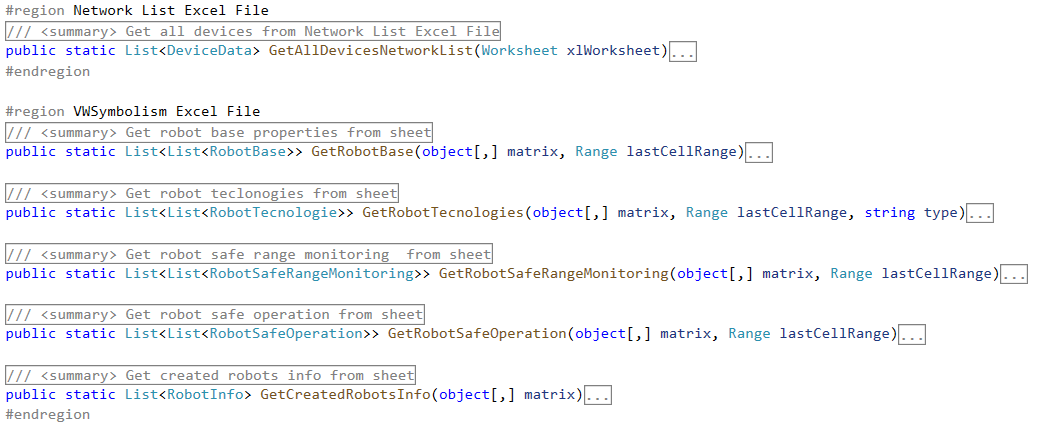


Figura 23 - Funções de leitura de ficheiros "Symbolic" e "NetworkList"

## PdfReader

A classe “*PdfReader*” encontra-se dentro do projecto “*TiaOpennessHelper*” e é onde se encontram as funções de manipulação de ficheiros PDF. É utilizada especificamente para leitura do *EPlan*.

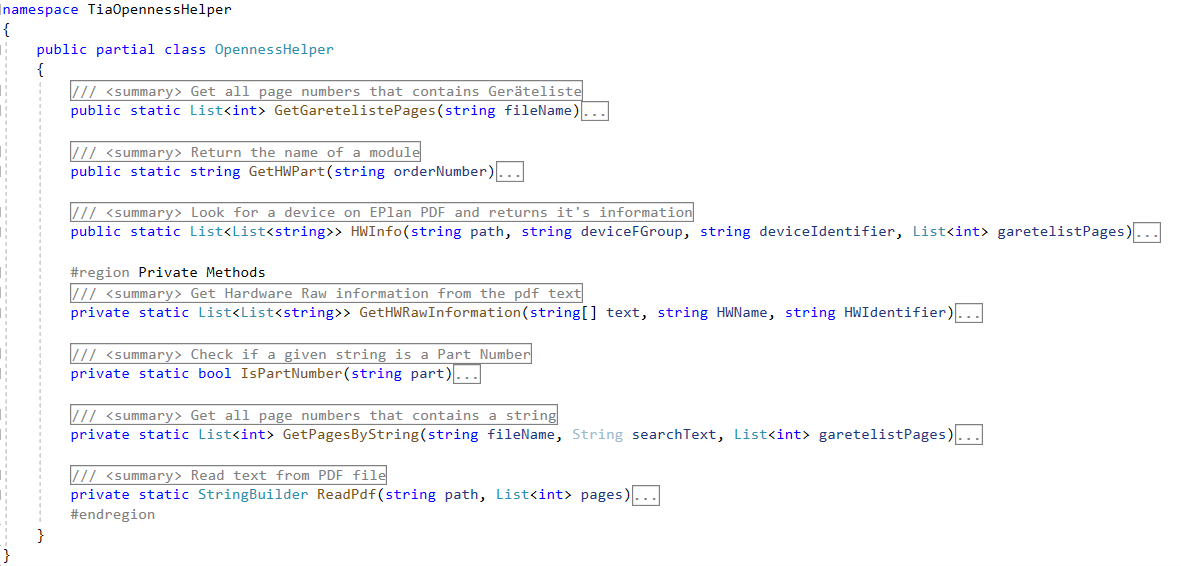


Figura 24 - Funções classe "PdfReader"

## XmlParser

A classe “*XmlParser*” encontra-se dentro do projecto “*TiaOpennessHelper*”, na pasta “*XMLParser*”, e é onde se encontram grande parte das funções de manipulação de ficheiros XML.

Para além das funções gerais, esta classe também contém funções responsáveis pela criação de ficheiros do tipo “*PLCTag*”.

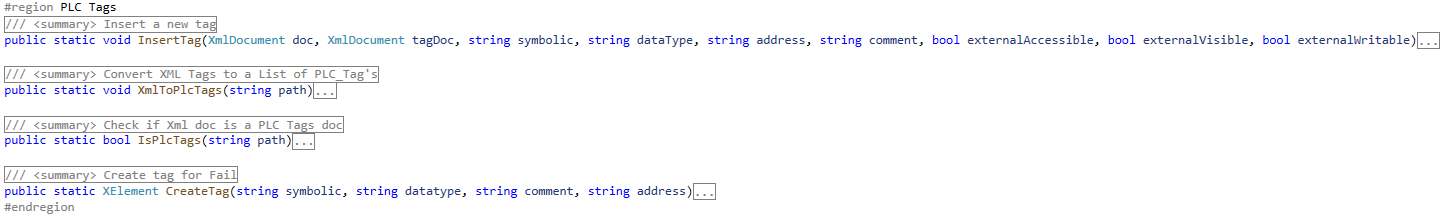


Figura 25 - Funções de criação de ficheiros XML do tipo "PLCTag"

## CacheManager

A classe “*CacheManager*” encontra-se dentro do projecto “*TiaOpennessHelper*”, na pasta “*Utils*”, e é onde se encontram as funções responsáveis pela manipulação de conteúdo em *cache*. É responsável por guardar, procurar, definir, limpar ou remover objetos em *cache*.

Esta classe é utilizada pela funcionalidade “*Main Folder Files List*”, para verificar se um determinado ficheiro já foi aberto e tem informações em *cache*.

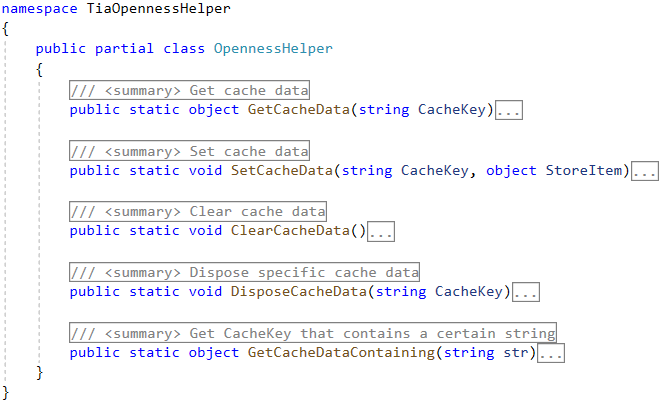


Figura 26 - Funções da classe "CacheManager"

# Notas finais

Existem muitas funções e classes presentes nos dois projetos que não foram mencionadas neste ficheiro porque são originais da aplicação “*TiaPortalOpenness*” e podem ser encontradas na documentação do “*TiaPortalOpenness*”.

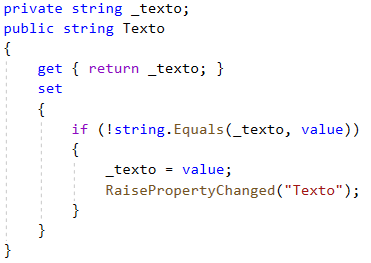
Grande parte das funções estão corretamente comentadas, sendo facilmente detetada a sua funcionalidade.

Sendo esta aplicação programada com a arquitetura MVVM, a maneira correta de aceder aos seus campos é pelo método “*Binding*”. Por exemplo, caso se queira aceder ao texto de um controlo do tipo “*TextBox*”, é necessário programar da seguinte maneira:

1. Declarar o controlo na *view*:

<TextBox Text="{Binding Texto}" />

1. Declarar na *viewmodel* a variável “Texto”:



1. Pode-se aceder ao texto dentro da *TextBox* criada pela variável “\_texto” e modificá-lo pela variável “Texto”.