WISMAS

1.1

Généré par Doxygen 1.7.6.1

Jeudi Juin 7 2018 20 :59 :17

Table des matières

1 Page principale du projet WISMAS

1.1 Introduction

Système d'information météo pour station multi activités WISMAS (Weather Information System Multi Activity Station)

1.2 Table des matières

Dépôt SVN : `https://svn.riouxsvn.com/wismas`

2 Changelog

---

r143 | pgrelet | 2018-06-07 17 :51 :26 +0200 (jeu. 07 juin 2018) | 1 ligne

Tag itération 1.1

---

r142 | pgrelet | 2018-06-07 17 :51 :12 +0200 (jeu. 07 juin 2018) | 1 ligne

Tag itération 1.1

---

r141 | opetrella | 2018-06-07 16 :19 :45 +0200 (jeu. 07 juin 2018) | 1 ligne

Correctif du nom de site SNOWKITE

---

r140 | opetrella | 2018-06-07 15 :42 :41 +0200 (jeu. 07 juin 2018) | 1 ligne

Correction des erreurs

---

r139 | opetrella | 2018-06-07 15 :22 :14 +0200 (jeu. 07 juin 2018) | 1 ligne

Correction des erreurs

---

r138 | pgrelet | 2018-06-07 12 :12 :20 +0200 (jeu. 07 juin 2018) | 1 ligne

Optimisation de la suppression

---

r137 | pgrelet | 2018-06-07 10 :48 :03 +0200 (jeu. 07 juin 2018) | 1 ligne

Correctif qDebug

---

r136 | pgrelet | 2018-06-07 10 :46 :12 +0200 (jeu. 07 juin 2018) | 1 ligne

Correctif sur les méthodes de suppression

---

r135 | pgrelet | 2018-06-06 15 :38 :56 +0200 (mer. 06 juin 2018) | 1 ligne

Correctif sur les paramétrages de déplacement

---

r134 | pgrelet | 2018-06-06 14 :06 :39 +0200 (mer. 06 juin 2018) | 1 ligne

Préparation démo

---

r133 | pgrelet | 2018-06-06 13 :40 :57 +0200 (mer. 06 juin 2018) | 1 ligne

Correction de la suppression : Nb_enregistrements

---

r132 | pgrelet | 2018-06-06 12 :17 :19 +0200 (mer. 06 juin 2018) | 1 ligne

Ajout/Correction de la suppression : VerifierDossierEnregistrements()/Nb_-enregistrements

---

r131 | opetrella | 2018-06-06 10 :43 :16 +0200 (mer. 06 juin 2018) | 1 ligne

modification chemin des videos fichier .ini

---

r130 | opetrella | 2018-06-05 08 :36 :43 +0200 (mar. 05 juin 2018) | 1 ligne

Ajout des qDebug pour la demo

---

r129 | opetrella | 2018-05-30 17 :49 :49 +0200 (mer. 30 mai 2018) | 1 ligne

Ajout des informations complémentaires au panneau lumineux

---

r128 | pgrelet | 2018-05-25 15 :52 :34 +0200 (ven. 25 mai 2018) | 1 ligne

Tag itération-1.0

---

r127 | pgrelet | 2018-05-25 15 :14 :56 +0200 (ven. 25 mai 2018) | 1 ligne

Fichiers annexes MàJ

---

r126 | pgrelet | 2018-05-25 14 :52 :37 +0200 (ven. 25 mai 2018) | 1 ligne

Modification sur des variables int

---

r125 | pgrelet | 2018-05-25 14 :41 :29 +0200 (ven. 25 mai 2018) | 1 ligne

Correction bug : déplacements des caméras

---

r124 | opetrella | 2018-05-25 14 :09 :02 +0200 (ven. 25 mai 2018) | 1 ligne

changement de version en 1.0

---

r123 | opetrella | 2018-05-25 12 :03 :57 +0200 (ven. 25 mai 2018) | 1 ligne

replacement des methodes dans WISMASIHM

---

r122 | pgrelet | 2018-05-24 12 :27 :32 +0200 (jeu. 24 mai 2018) | 1 ligne

Modification sur la suppression des enregistrements

---

r121 | pgrelet | 2018-05-24 09 :49 :10 +0200 (jeu. 24 mai 2018) | 1 ligne

Modification sur le déplacement des caméras

---

r120 | tvaira | 2018-05-24 08 :09 :27 +0200 (jeu. 24 mai 2018) | 1 ligne

Script d'installation de la base de donnéees

---

r119 | pgrelet | 2018-05-23 17 :50 :07 +0200 (mer. 23 mai 2018) | 1 ligne

Modification syntaxe IHM

---

r118 | tvaira | 2018-05-23 11 :32 :19 +0200 (mer. 23 mai 2018) | 1 ligne

Maj fichier SQL

---

r117 | pgrelet | 2018-04-26 15 :32 :46 +0200 (jeu. 26 avril 2018) | 1 ligne

Correctif sur le rafraichissement du flux vidéo

---

r116 | pgrelet | 2018-04-26 14 :01 :09 +0200 (jeu. 26 avril 2018) | 1 ligne

Préparation pour itération 1.0

---

r115 | pgrelet | 2018-04-26 14 :00 :33 +0200 (jeu. 26 avril 2018) | 1 ligne

Préparation pour itération 1.0

---

r114 | pgrelet | 2018-04-26 13 :43 :25 +0200 (jeu. 26 avril 2018) | 1 ligne

Correction sur les déplacements caméra

---

r113 | pgrelet | 2018-04-26 12 :44 :51 +0200 (jeu. 26 avril 2018) | 1 ligne

Ajout fonctionnalité activé/désactivé depuis l'IHMcd

---

r112 | pgrelet | 2018-04-25 16 :34 :56 +0200 (mer. 25 avril 2018) | 1 ligne

Réorganisation du code

---

r111 | pgrelet | 2018-04-25 12 :20 :14 +0200 (mer. 25 avril 2018) | 1 ligne

Correctif sur les déplacements caméras

---

r110 | pgrelet | 2018-04-24 20 :36 :19 +0200 (mar. 24 avril 2018) | 1 ligne

Itération-0.9 tag

---

r109 | pgrelet | 2018-04-24 20 :26 :06 +0200 (mar. 24 avril 2018) | 1 ligne

MàJ README et Changelog - itération-0.9

---

r108 | opetrella | 2018-04-23 17 :49 :28 +0200 (lun. 23 avril 2018) | 1 ligne

adaptation taille de l'ecran television

---

r107 | pgrelet | 2018-04-23 17 :21 :43 +0200 (lun. 23 avril 2018) | 1 ligne

Préparation pour la revue 3

---

r106 | opetrella | 2018-04-23 16 :41 :55 +0200 (lun. 23 avril 2018) | 1 ligne

remplacement des fonds de couleur de l'interface

---

r105 | opetrella | 2018-04-23 14 :48 :18 +0200 (lun. 23 avril 2018) | 1 ligne

adaptation de l'affichage

---

r104 | opetrella | 2018-04-23 14 :07 :39 +0200 (lun. 23 avril 2018) | 1 ligne

Mise en place du chargement de la video la plus recente pour chaque site

---

r103 | pgrelet | 2018-04-23 10 :56 :28 +0200 (lun. 23 avril 2018) | 1 ligne

Modification sur le script cvlc.sh

---

r102 | pgrelet | 2018-04-22 19 :05 :27 +0200 (dim. 22 avril 2018) | 1 ligne

Correction suppression des enregistrements

---

r101 | pgrelet | 2018-04-22 17 :39 :35 +0200 (dim. 22 avril 2018) | 1 ligne

Syntaxe à jour ; Doxygen

---

r100 | pgrelet | 2018-04-22 03 :14 :12 +0200 (dim. 22 avril 2018) | 1 ligne

Changelog à jour

---

r99 | pgrelet | 2018-04-22 03 :12 :26 +0200 (dim. 22 avril 2018) | 1 ligne

Correctif sur la sauvegarde des positions en déplacement fixe

---

r98 | pgrelet | 2018-04-22 03 :02 :08 +0200 (dim. 22 avril 2018) | 1 ligne

Correctif suppression niveau 2

---

r97 | pgrelet | 2018-04-19 21 :44 :39 +0200 (jeu. 19 avril 2018) | 1 ligne

MàJ de la doxygen et des commentaires

---

r96 | pgrelet | 2018-04-19 21 :18 :04 +0200 (jeu. 19 avril 2018) | 1 ligne

Développement : suppression des enregistrements

---

r95 | pgrelet | 2018-04-19 12 :14 :24 +0200 (jeu. 19 avril 2018) | 1 ligne

Modification de la fonction enregistrer vidéos

---

r94 | pgrelet | 2018-04-18 17 :57 :14 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout : suppression des vidéos enregistrées

---

r93 | pgrelet | 2018-04-18 12 :04 :17 +0200 (mer. 18 avril 2018) | 1 ligne

Développement : désactiver les caméras

---

r92 | pgrelet | 2018-04-18 11 :45 :51 +0200 (mer. 18 avril 2018) | 1 ligne

Développement : désactiver les caméras

---

r91 | pgrelet | 2018-04-18 10 :50 :15 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout : Désactiver les caméras

---

r90 | tvaira | 2018-04-16 13 :56 :31 +0200 (lun. 16 avril 2018) | 1 ligne

Verification des TODO

---

r89 | tvaira | 2018-04-13 10 :04 :56 +0200 (ven. 13 avril 2018) | 1 ligne

Ajout du script cvlc.sh

---

r88 | pgrelet | 2018-04-12 22 :26 :12 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout répertoire videos

---

r87 | pgrelet | 2018-04-12 19 :27 :58 +0200 (jeu. 12 avril 2018) | 1 ligne

Edition du chemin d'enregistrement des vidéos

---

r86 | pgrelet | 2018-04-12 19 :22 :26 +0200 (jeu. 12 avril 2018) | 1 ligne

Développement du chemin où enregistrer les vidéos

---

r85 | opetrella | 2018-04-12 18 :13 :09 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout du compteur pour enregistrer toute les heures en base de données

---

r84 | pgrelet | 2018-04-12 12 :29 :28 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout de la suppression automatique des vidéos

---

r83 | opetrella | 2018-04-12 12 :28 :33 +0200 (jeu. 12 avril 2018) | 1 ligne

creation de la methode de changement de page sur le panneau lumineux

---

r82 | opetrella | 2018-04-11 13 :44 :07 +0200 (mer. 11 avril 2018) | 1 ligne

recherche panneau lumineux (essayer de changer les pages du panneau lumineux)

---

r81 | pgrelet | 2018-04-11 12 :32 :16 +0200 (mer. 11 avril 2018) | 1 ligne

Edition des qDebugs

---

r80 | pgrelet | 2018-04-11 12 :14 :20 +0200 (mer. 11 avril 2018) | 1 ligne

Ajout paramètre résolution et optimisation timerCamera

---

r79 | pgrelet | 2018-04-06 12 :29 :06 +0200 (ven. 06 avril 2018) | 1 ligne

Développement des bars de progression

---

r78 | pgrelet | 2018-04-06 12 :23 :59 +0200 (ven. 06 avril 2018) | 1 ligne

Développement de l'enregistrement terminé

---

r77 | opetrella | 2018-04-06 12 :21 :25 +0200 (ven. 06 avril 2018) | 1 ligne

Rajout de parametres panneau lumineux fichier ini plus structure

---

r76 | opetrella | 2018-04-05 17 :52 :09 +0200 (jeu. 05 avril 2018) | 1 ligne

Ajout d'option au panneau lumineux

---

r75 | pgrelet | 2018-04-05 17 :51 :56 +0200 (jeu. 05 avril 2018) | 1 ligne

Développement de l'enregistrement

---

r74 | pgrelet | 2018-04-05 12 :31 :12 +0200 (jeu. 05 avril 2018) | 1 ligne

Développement Enregistrement vidéo

---

r73 | opetrella | 2018-04-04 19 :30 :44 +0200 (mer. 04 avril 2018) | 1 ligne

Creation de la classe PortPanneau

---

r72 | opetrella | 2018-04-04 19 :30 :19 +0200 (mer. 04 avril 2018) | 1 ligne

Creation de la classe PortPanneau

---

r71 | pgrelet | 2018-04-04 17 :57 :15 +0200 (mer. 04 avril 2018) | 1 ligne

Modification enregistrement vidéo

---

r70 | pgrelet | 2018-04-04 17 :28 :18 +0200 (mer. 04 avril 2018) | 1 ligne

Ajout de attendre() fonction d'attente pour deplacerCamera()

---

r69 | pgrelet | 2018-04-04 15 :54 :02 +0200 (mer. 04 avril 2018) | 1 ligne

Ajout fonctionnalité enregistrer vidéos

---

r68 | pgrelet | 2018-04-04 15 :23 :30 +0200 (mer. 04 avril 2018) | 1 ligne

Déplacement panoramique

---

r67 | opetrella | 2018-04-04 15 :17 :00 +0200 (mer. 04 avril 2018) | 1 ligne

Ajout des moyens horaire en base de données

---

r66 | tvaira | 2018-04-02 21 :08 :32 +0200 (lun. 02 avril 2018) | 1 ligne

Retour sur la revue 2 pour la partie Acquisition

---

r65 | tvaira | 2018-04-02 16 :22 :27 +0200 (lun. 02 avril 2018) | 1 ligne

Retour sur la revue 2

---

r64 | pgrelet | 2018-03-30 12 :29 :27 +0200 (ven. 30 mars 2018) | 1 ligne

Développement acquisition vidéo : boucle

---

r63 | opetrella | 2018-03-30 12 :28 :19 +0200 (ven. 30 mars 2018) | 1 ligne

Ajout de Phonon

---

r62 | opetrella | 2018-03-29 17 :54 :46 +0200 (jeu. 29 mars 2018) | 1 ligne

ajout des conseils de fartage + gestion des erreurs sur l'interface

---

r61 | pgrelet | 2018-03-29 17 :54 :46 +0200 (jeu. 29 mars 2018) | 1 ligne

Dévellopement de l'acquisition vidéo : déplacement

---

r60 | pgrelet | 2018-03-28 19 :52 :54 +0200 (mer. 28 mars 2018) | 1 ligne

Modification IHM : paramétrage système

---

r57 | pgrelet | 2018-03-28 17 :54 :37 +0200 (mer. 28 mars 2018) | 1 ligne

Fonctionnalités : Enregistrer la video

---

r56 | opetrella | 2018-03-28 17 :12 :30 +0200 (mer. 28 mars 2018) | 1 ligne

décodage des trames + affichage

---

r55 | pgrelet | 2018-03-28 12 :27 :45 +0200 (mer. 28 mars 2018) | 1 ligne

include QDebug Video.h

---

r52 | pgrelet | 2018-03-28 12 :12 :42 +0200 (mer. 28 mars 2018) | 1 ligne

Ajout de la sauvegarde des paramètres et amélioration de l'IHM

---

r49 | opetrella | 2018-03-27 16 :23 :17 +0200 (mar. 27 mars 2018) | 1 ligne

rectification de la version du programme en version 0.8

---

r48 | pgrelet | 2018-03-26 17 :51 :52 +0200 (lun. 26 mars 2018) | 1 ligne

Doxygen à jour

---

r47 | pgrelet | 2018-03-26 16 :20 :39 +0200 (lun. 26 mars 2018) | 1 ligne

Correction capture vidéo QWebView

---

r46 | pgrelet | 2018-03-25 23 :49 :45 +0200 (dim. 25 mars 2018) | 1 ligne

Modification des paramètres

---

r45 | tvaira | 2018-03-24 17 :01 :39 +0100 (sam. 24 mars 2018) | 1 ligne

Test du simulateur

---

r44 | tvaira | 2018-03-24 12 :20 :06 +0100 (sam. 24 mars 2018) | 1 ligne

Mise a jour Doxygen

---

r43 | tvaira | 2018-03-24 12 :18 :16 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout du simulateur de Station Meteo

---

r42 | tvaira | 2018-03-24 12 :03 :46 +0100 (sam. 24 mars 2018) | 1 ligne

Mise a jour Changelog Doxygen

---

r41 | tvaira | 2018-03-24 12 :01 :53 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout parametrage Doxygen

---

r40 | pgrelet | 2018-03-23 12 :29 :23 +0100 (ven. 23 mars 2018) | 1 ligne

Codage des déplacements caméras

---

r39 | pgrelet | 2018-03-22 17 :56 :04 +0100 (jeu. 22 mars 2018) | 1 ligne

Codage des déplacements caméras

---

r38 | opetrella | 2018-03-22 17 :54 :17 +0100 (jeu. 22 mars 2018) | 1 ligne

Décodage des trames de la station météo

---

r37 | pgrelet | 2018-03-22 14 :57 :41 +0100 (jeu. 22 mars 2018) | 1 ligne

Lecture Configuration cameras fichier INI

---

r36 | opetrella | 2018-03-22 12 :35 :52 +0100 (jeu. 22 mars 2018) | 1 ligne

Affichage station interface

---

r35 | pgrelet | 2018-03-22 12 :31 :05 +0100 (jeu. 22 mars 2018) | 1 ligne

Creation du projet Acquisition Video

---

r34 | pgrelet | 2018-03-22 12 :30 :19 +0100 (jeu. 22 mars 2018) | 1 ligne

---

r33 | pgrelet | 2018-03-21 17 :56 :48 +0100 (mer. 21 mars 2018) | 1 ligne

Codage des paramètres, écriture et lecture

---

r32 | opetrella | 2018-03-21 17 :54 :39 +0100 (mer. 21 mars 2018) | 1 ligne

Creation de la structure de fonctionement du fichier INI

---

r31 | opetrella | 2018-03-21 17 :54 :02 +0100 (mer. 21 mars 2018) | 1 ligne

Creation de la structure de fonctionement du fichier INI

---

r30 | pgrelet | 2018-03-19 15 :09 :14 +0100 (lun. 19 mars 2018) | 1 ligne

Codage des signaux/slots et création des paramètres

---

r29 | pgrelet | 2018-03-16 11 :28 :41 +0100 (ven. 16 mars 2018) | 1 ligne

Modification sur IHM paramétrage des caméras

---

r28 | pgrelet | 2018-03-15 16 :59 :06 +0100 (jeu. 15 mars 2018) | 1 ligne

Modification sur la navigation entre les caméras

---

r27 | opetrella | 2018-03-15 16 :57 :01 +0100 (jeu. 15 mars 2018) | 1 ligne

suppression de la classe sites

---

r26 | opetrella | 2018-03-15 16 :55 :05 +0100 (jeu. 15 mars 2018) | 1 ligne

renommage de la classe StationsMeteo

---

r25 | pgrelet | 2018-03-15 16 :54 :53 +0100 (jeu. 15 mars 2018) | 1 ligne

Déclaration des classes système d'acquisition vidéo et codage des signaux/slots

---

r24 | pgrelet | 2018-03-15 16 :53 :31 +0100 (jeu. 15 mars 2018) | 1 ligne

Déclaration des classes système d'acquisition vidéo et codage des signaux/slots

---

r23 | opetrella | 2018-03-15 16 :52 :50 +0100 (jeu. 15 mars 2018) | 1 ligne

Ajout de la bibliothèque SQL

---

r22 | opetrella | 2018-03-02 09 :52 :52 +0100 (ven. 02 mars 2018) | 1 ligne

suppression des fichiers à ignorer

---

r21 | opetrella | 2018-03-02 09 :48 :11 +0100 (ven. 02 mars 2018) | 1 ligne

interface client

---

r20 | pgrelet | 2018-02-22 17 :47 :40 +0100 (jeu. 22 févr. 2018) | 1 ligne

Changelog à jour

---

r19 | pgrelet | 2018-02-22 17 :46 :52 +0100 (jeu. 22 févr. 2018) | 1 ligne

Ajout choix cameras IHM

---

r18 | pgrelet | 2018-02-22 17 :46 :11 +0100 (jeu. 22 févr. 2018) | 1 ligne

Ajout choix cameras IHM

---

r17 | opetrella | 2018-02-22 17 :03 :43 +0100 (jeu. 22 févr. 2018) | 1 ligne

finalisation des slots

---

r16 | pgrelet | 2018-02-22 16 :44 :26 +0100 (jeu. 22 févr. 2018) | 1 ligne

Ajout IHM Mes vidéos

---

r15 | opetrella | 2018-02-22 15 :53 :51 +0100 (jeu. 22 févr. 2018) | 1 ligne

AJout des signaux et des slots

---

r14 | pgrelet | 2018-02-22 14 :18 :30 +0100 (jeu. 22 févr. 2018) | 1 ligne

Nouvelle structure IHM

---

r13 | pgrelet | 2018-02-22 14 :03 :36 +0100 (jeu. 22 févr. 2018) | 1 ligne

Renommage des variables IHM

---

r12 | pgrelet | 2018-02-22 13 :56 :24 +0100 (jeu. 22 févr. 2018) | 1 ligne

Renommage des variables IHM

---

r11 | pgrelet | 2018-02-21 13 :59 :39 +0100 (mer. 21 févr. 2018) | 1 ligne

Ajout de la fenêtre IHM accueil

---

r10 | opetrella | 2018-02-21 13 :46 :28 +0100 (mer. 21 févr. 2018) | 1 ligne

restoration de l'interface

---

r9 | pgrelet | 2018-02-21 12 :34 :24 +0100 (mer. 21 févr. 2018) | 1 ligne

Nouvelle structure de l'IHM

---

r8 | opetrella | 2018-02-21 12 :14 :16 +0100 (mer. 21 févr. 2018) | 1 ligne

Nouvelle interface client

---

r7 | pgrelet | 2018-02-21 09 :09 :02 +0100 (mer. 21 févr. 2018) | 1 ligne

Renommage des variables de l'IHM

---

r6 | pgrelet | 2018-02-21 08 :52 :28 +0100 (mer. 21 févr. 2018) | 1 ligne

Renommage des variables de l'IHM

---

r5 | pgrelet | 2018-02-21 08 :41 :51 +0100 (mer. 21 févr. 2018) | 1 ligne

MàJ compilateur simulateur-wismas

---

r4 | pgrelet | 2018-02-18 23 :29 :30 +0100 (dim. 18 févr. 2018) | 1 ligne

MàJ du README

---

r3 | pgrelet | 2018-02-18 23 :20 :53 +0100 (dim. 18 févr. 2018) | 1 ligne

Ajout du prototypage IHM

---

r2 | tvaira | 2018-02-03 11 :13 :33 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout initial (tv)

---

r1 | www-data | 2018-02-03 11 :04 :04 +0100 (sam. 03 févr. 2018) | 1 ligne

Creating initial repository structure ----------

---

3 Recette IR

Étudiant 3 : Grelet Pierre

- Le système d'acquisition vidéo est paramétrable
- Les déplacements panoramique et manuel d'une caméra fonctionnent
- La vidéo s'enregistre sur un serveur NFS
- Une acquisition périodique des caméras est possible
- Les enregistrements sont supprimables

Étudiant 4 : Petrella Olivier

– Le protocole de communication est spécifié et mis en œuvre ;
– Les mesures météorologiques d'un site sont relevées
– Les mesures météorologiques relevées sont enregistrées
– Les informations sur la station sont consultables sur l'écran
– Les conditions météorologiques d'un site sont consultables sur le panneau lumineux

4 Base de données MySQL

CREATE DATABASE IF NOT EXISTS ‘WISMAS_2018‘ ;

USE ‘WISMAS_2018‘ ;

CREATE TABLE IF NOT EXISTS ‘sites‘ (‘idSite‘ int(11) NOT NULL, ‘type‘ int(11) NOT NULL, ‘nom‘ varchar(255) NOT NULL, ‘etat‘ int(11) NOT NULL, PRIMARY KEY (‘idSite‘)) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE IF NOT EXISTS ‘mesures‘ (‘idMesure‘ int(11) NOT NULL AUTO_INCREMENT, ‘idSite‘ int(11) NOT NULL, ‘dateMesure‘ date NOT NULL, ‘heure‘ int(4), ‘temperatureAir‘ FLOAT, ‘temperatureNeige‘ FLOAT, ‘hauteurNeige‘ int(4), ‘humidite‘ int(4), ‘pressionAir‘ int(4), ‘vitesseVent‘ FLOAT, ‘directionVent‘ VARCHAR(3), PRIMARY KEY (‘idMesure‘), CONSTRAINT mesures_fk_1 FOREIGN KEY (‘idSite‘) REFERENCES sites(‘idSite‘) ON DELETE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE IF NOT EXISTS ‘informations‘ (‘idInfo‘ int(11) NOT NULL AUTO_INCREMENT, ‘idSite‘ int(11) NOT NULL, ‘tarifs‘ varchar(255) NOT NULL, ‘horaire‘ varchar(255) NOT NULL, ‘nbPisteOuverte‘ int(4), PRIMARY KEY (‘idInfo‘), CONSTRAINT informations_fk_1 FOREIGN KEY (‘idSite‘) REFERENCES sites(‘idSite‘) ON DELETE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

5 Fichiers de configuration

– configuration-cameras.ini

[General] nb_cameras=3 nb_videos=3 niveau_suppression=Nb_enregistrements

[Camera1] etat=1 nom=Alpin chemin_video=./videos/alpin/ adresse_IP=192.168.52.221 numero_port=99 identifiant=admin mot_de_passe= type_deplacement=Panoramique duree=15 periode=90 resolution=480

[Camera2] etat=1 nom=Fond chemin_video=./videos/fond/ adresse_IP=192.168.52.93 numero_port=99 identifiant=admin mot_de_passe= type_deplacement=Fixe duree=15 periode=60 resolution=480

[Camera3] etat=0 nom=SnowKite chemin_video=./videos/snowKite/ adresse_IP=192.168.52.150 numero_port=81 identifiant=admin mot_de_passe= type_deplacement=Panoramique duree=15 periode=30 resolution=720

– WISMAS_station.ini

[Configuration] nb_station=3 periode=40

[Enregistrement] periode=60

[Station_configuration] port=ttyUSB0 baud_rate=9600 data_bit=8 stop=1 parity=0 periode=1000

[Panneau_configuration] port=ttyUSB1 baud_rate=9600 data_bit=8 stop=1 parity=0 periode=1000

[Panneau_affichage] titre=Station WISMAS taille_text=normal temps_text=6 effet_transition=normal nom_station=true date_heure=true temperature_air=true hauteur_neige=true humidite=false pression=false vitesse_vent=true

[Station1] id=101 type=1 nom="Vallée de la mort" chemin_video="/videos/alpin/" couleur_text_panneau="rouge"

[Station2] id=201 type=2 nom="Mont Blanc" chemin_video="/videos/fond/" couleur_text_panneau="vert"

[Station3] id=301 type=3 nom="Mont noir" chemin_video="/videos/snowKite/" couleur_text_panneau="tricolor"

6 A propos

Auteur

Grelet Pierre <pierre.grelet@outlook.fr>
Petrella Olivier <olivier.petrella@gmail.com>

Version

1.1

Date

2018

7 Licence GPL

8 Liste des choses à faire

Membre WISMASIHM : :chargerVideo ()
charger la dernière video.

9 Documentation des classes

9.1 Référence de la structure AFFICHAGE_PANNEAU

fixe la configuration de l'affichage des informations sur le panneau lumineux. titre QString, int periode, bool temperature_air, bool temperature_neige, bool hauteur_neige, bool humidite, bool pression, bool vitesse_vent,

`#include <structures.h>`

Attributs publics

- QString titre
- int periode
- QString taille_text
- int temps_text
- QString effet_transition
- bool nom_station
- bool date_heure
- bool temperature_air
- bool temperature_neige
- bool hauteur_neige
- bool humidite
- bool pression
- bool vitesse_vent

9.1.1 Documentation des données membres

9.1.1.1 bool AFFICHAGE_PANNEAU : :date_heure

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.2 QString AFFICHAGE_PANNEAU : :effet_transition

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.3 bool AFFICHAGE_PANNEAU : :hauteur_neige

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.4 bool AFFICHAGE_PANNEAU : :humidite

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.5 bool AFFICHAGE_PANNEAU : :nom_station

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.6 int AFFICHAGE_PANNEAU : :periode

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.7 bool AFFICHAGE_PANNEAU : :pression

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.8 QString AFFICHAGE_PANNEAU : :taille_text

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.9 bool AFFICHAGE_PANNEAU : :temperature_air

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.10 bool AFFICHAGE_PANNEAU : :temperature_neige

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.11 int AFFICHAGE_PANNEAU : :temps_text

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.12 QString AFFICHAGE_PANNEAU : :titre

Référencé par WISMASIHM : :chargerFichierConfig().

9.1.1.13 bool AFFICHAGE_PANNEAU : :vitesse_vent

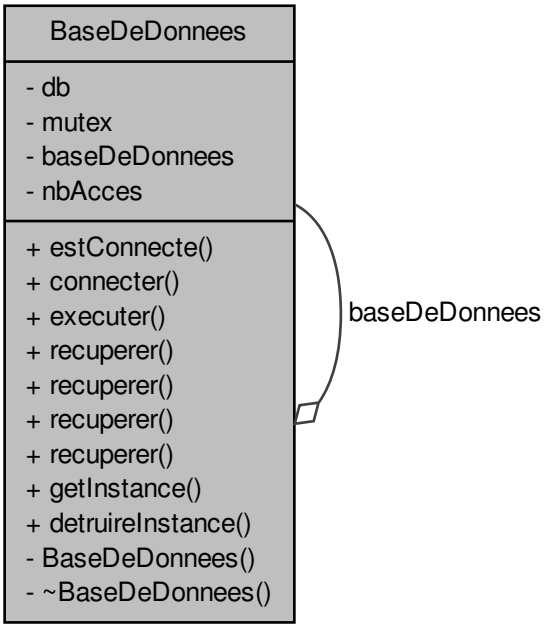
Référencé par WISMASIHM : :chargerFichierConfig().

La documentation de cette structure a été générée à partir du fichier suivant :
– structures.h

9.2 Référence de la classe BaseDeDonnees

```
#include <basededonnees.h>
```

Graphe de collaboration de BaseDeDonnees :



Fonctions membres publiques

- bool estConnecte ()
- bool connecter (QString nomBase=DATABASENAME)
- bool executer (QString requete)
- bool recuperer (QString requete, QString &donnees)
- bool recuperer (QString requete, QStringList &donnees)
- bool recuperer (QString requete, QVector< QString > &donnees)
- bool recuperer (QString requete, QVector< QStringList > &donnees)

Fonctions membres publiques statiques

- static BaseDeDonnees * getInstance ()
- static void detruireInstance ()

Fonctions membres privées

- BaseDeDonnees ()
- ~BaseDeDonnees ()

Attributs privés

– QSqlDatabase db
– QMutex mutex

Attributs privés statiques

– static BaseDeDonnees * baseDeDonnees = NULL
– static int nbAcces = 0

9.2.1 Documentation des constructeurs et destructeur

9.2.1.1 BaseDeDonnees : :BaseDeDonnees ( ) [private]

Références db.

Référencé par getInstance().

```
{
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::BaseDeDonnees()>";
    #endif
    db = QSqlDatabase::addDatabase("QMYSQL");
}
```

9.2.1.2 BaseDeDonnees : :~BaseDeDonnees ( ) [private]

```
{
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::~BaseDeDonnees()>";
    #endif
}
```

9.2.2 Documentation des fonctions membres

9.2.2.1 bool BaseDeDonnees : :connecter (QString *nomBase* = DATABASENAME)

Références db, HOSTNAME, mutex, PASSWORD, et USERNAME.

Référencé par StationMeteo : :StationMeteo().

```
{
    QMutexLocker verrou(&mutex);
    if(!db.isOpen())
    {
       db.setHostName(HOSTNAME);
       db.setUserName(USERNAME);
       db.setPassword(PASSWORD);
       db.setDatabaseName(nomBase);

       #ifdef DEBUG_BASEDEDONNEES
       qDebug() << "HostName : " << db.hostName();
       qDebug() << "UserName : " << db.userName();
       qDebug() << "DatabaseName : " << db.databaseName();
       #endif
       if(db.open())
       {
           #ifdef DEBUG_BASEDEDONNEES
           qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()>
```

```
        connexion réussie à %1").arg(db.hostName());
            #endif

            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()> erreur :
        impossible de se connecter à la base de données !");

            QMessageBox::critical(0, QString::fromUtf8("Test MO"),
       QString::fromUtf8("Impossible de se connecter à la base de données !"));

            return false;
        }
    }
    else
        return true;
}
```

9.2.2.2 void BaseDeDonnees : :detruireInstance ( ) [static]

Références baseDeDonnees, et nbAcces.

Référencé par StationMeteo : :~StationMeteo().

```
{
    // instance ?
    if(baseDeDonnees != NULL)
    {
        nbAcces--;
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << "<BaseDeDonnees::detruireInstance()> nbAcces restants = " <
      < nbAcces;
        #endif
        // dernier ?
        if(nbAcces == 0)
            delete baseDeDonnees;
    }
}
```

9.2.2.3 bool BaseDeDonnees : :estConnecte ( )

Références db.

Référencé par StationMeteo : :enregistrerDonnees(), et StationMeteo : :getInformation-Complementaire().

```
{
    return db.isOpen();
}
```

9.2.2.4 bool BaseDeDonnees : :executer (QString *requete*)

Références db, et mutex.

Référencé par StationMeteo : :enregistrerDonnees(), StationMeteo : :StationMeteo(), et StationMeteo : :~StationMeteo().

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
```

```
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::executer()> retour %1
       pour la requete : %2").arg(QString::number(retour)).arg(requete);
        #endif
        if(retour)
        {
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::executer()> erreur :
       %1 pour la requête %2").arg(r.lastError().text()).arg(requete);

            return false;
        }
    }
    else
        return false;

}
```

9.2.2.5 BaseDeDonnees * BaseDeDonnees : :getInstance ( ) [static]

Références BaseDeDonnees(), baseDeDonnees, et nbAcces.

Référencé par StationMeteo : :StationMeteo().

```
{
    if(baseDeDonnees == NULL)
        baseDeDonnees = new BaseDeDonnees();

    nbAcces++;
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::getInstance()> nbAcces = " << nbAcces;
    #endif

    return baseDeDonnees;
}
```

9.2.2.6 bool BaseDeDonnees : :recuperer (QString *requete,* QString & *donnees*)

Références db, et mutex.

Référencé par StationMeteo : :getInformationComplementaire(), et StationMeteo : :-StationMeteo().

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
       QString)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(requete);
        #endif
        if(retour)
        {
            // on se positionne sur l'enregistrement
```

```
            r.first();

            // on vérifie l'état de l'enregistrement retourné
            if(!r.isValid())
            {
                #ifdef DEBUG_BASEDEDONNEES
                qDebug() << QString::fromUtf8("
    <BaseDeDonnees::recuperer(QString, QString)> résultat non valide !");
                #endif
                return false;
            }

            // on récupère sous forme de QString la valeur du champ
            if(r.isNull(0))
            {
                #ifdef DEBUG_BASEDEDONNEES
                qDebug() << QString::fromUtf8("
    <BaseDeDonnees::recuperer(QString, QString)> résultat vide !");
                #endif
                return false;
            }
            donnees = r.value(0).toString();
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString, QString)>
     enregistrement -> " << donnees;
            #endif
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
     QString)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg(requete)
    ;

            return false;
        }
    }
    else
        return false;
}
```

9.2.2.7 bool BaseDeDonnees : :recuperer (QString *requete,* QStringList & *donnees*)

Références db, et mutex.

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
     QStringList)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(
    requete);
        #endif
        if(retour)
        {
            // on se positionne sur l'enregistrement
            r.first();

            // on vérifie l'état de l'enregistrement retourné
            if(!r.isValid())
            {
                #ifdef DEBUG_BASEDEDONNEES
                qDebug() << QString::fromUtf8("
    <BaseDeDonnees::recuperer(QString, QStringList)> résultat non valide !");
                #endif
                return false;
```

```
            }

            // on récupère sous forme de QString la valeur de tous les champs
        sélectionnés
            // et on les stocke dans une liste de QString
            for(int i=0;i<r.record().count();i++)
                if(!r.isNull(i))
                    donnees << r.value(i).toString();
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString, QStringList)>
        enregistrement -> " << donnees;
            #endif
            return true;
         }
         else
         {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
        QStringList)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg(
       requete);
            return false;
         }
    }
    else
        return false;
}
```

9.2.2.8 bool BaseDeDonnees : :recuperer (QString *requete,* QVector< QString > & *donnees*)

Références db, et mutex.

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;
    QString data;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
       QVector<QString>)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(
      requete);
        #endif
        if(retour)
        {
            // pour chaque enregistrement
            while ( r.next() )
            {
                // on récupère sous forme de QString la valeur du champs
        sélectionné
                data = r.value(0).toString();

                #ifdef DEBUG_BASEDEDONNEES
                //qDebug() << "<BaseDeDonnees::recuperer(QString,
        QVector<QString>)> enregistrement -> " << data;
                #endif

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);
            }
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString, QVector<QString>)>
        enregistrement -> " << donnees;
            #endif
            return true;
         }
         else
         {
```

```
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
       QVector<QString>)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg
      (requete);

            return false;
        }
    }
    else
        return false;
}
```

9.2.2.9 bool BaseDeDonnees : :recuperer (QString *requete,* QVector< QStringList > & *donnees*)

Références db, et mutex.

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;
    QStringList data;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
       QVector<QStringList>)> retour %1 pour la requete : %2").arg(QString::number(retour)).
      arg(requete);
        #endif
        if(retour)
        {
            // pour chaque enregistrement
            while ( r.next() )
            {
                // on récupère sous forme de QString la valeur de tous les
       champs sélectionnés
                // et on les stocke dans une liste de QString
                for(int i=0;i<r.record().count();i++)
                    data << r.value(i).toString();

                #ifdef DEBUG_BASEDEDONNEES
                //qDebug() << "<BaseDeDonnees::recuperer(QString,
       QVector<QStringList>)> enregistrement -> " << data;
                /*for(int i=0;i<r.record().count();i++)
                    qDebug() << r.value(i).toString();*/
                #endif

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);

                // on efface la liste de QString pour le prochain
       enregistrement
                data.clear();
            }
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString,
       QVector<QStringList>)> enregistrement -> " << donnees;
            #endif
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
       QVector<QStringList>)> erreur : %1 pour la requête %2").arg(r.lastError().text())
      .arg(requete);

            return false;
        }
    }
```

```
    else
        return false;
}
```

9.2.3 Documentation des données membres

9.2.3.1 BaseDeDonnees * BaseDeDonnees : :baseDeDonnees = NULL [static, private]

Référencé par detruireInstance(), et getInstance().

9.2.3.2 QSqlDatabase BaseDeDonnees : :db [private]

Référencé par BaseDeDonnees(), connecter(), estConnecte(), executer(), et recuperer().

9.2.3.3 QMutex BaseDeDonnees : :mutex [private]

Référencé par connecter(), executer(), et recuperer().

9.2.3.4 int BaseDeDonnees : :nbAcces = 0 [static, private]

Référencé par detruireInstance(), et getInstance().

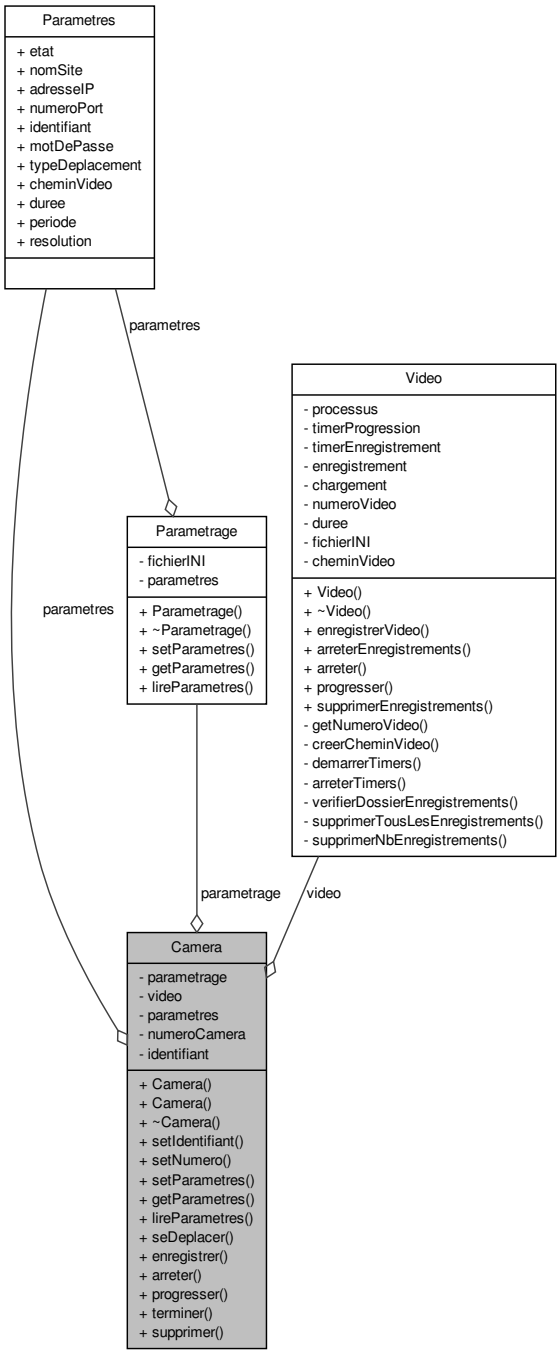
La documentation de cette classe a été générée à partir des fichiers suivants :
– basededonnees.h
– basededonnees.cpp

9.3 Référence de la classe Camera

Acquérir le système vidéo.

`#include <Camera.h>`

Graphe de collaboration de Camera :



Connecteurs publics

- void enregistrer () const
  Enregistrer les images vidéos sur une durée.
- void arreter () const
  Arrêter l'enregistrement.
- void progresser (int pct) const
  Envoyer la progression des enregistrements.
- void terminer () const
  Réinitialiser les bars de progression de l'acquisition.
- void supprimer () const
  Supprimer les enregistrements.

Signaux

- void afficherMessage (QString message) const
- void progression (int chargement) const

Fonctions membres publiques

- Camera ()
  Constructeur.
- Camera (int numeroCamera, QString identifiant)
  Constructeur.
- ∼Camera ()
  Destructeur.
- void setIdentifiant (QString identifiant)
- void setNumero (int numeroCamera)
- void setParametres (Parametres parametres)
  Mutateur de la structure parametres.
- Parametres getParametres () const
  Accesseur de la structure parametres.
- void lireParametres ()
  Lire les paramètres de la caméra.
- QString seDeplacer () const
  Déplacer la caméra vers la position initiale.

Attributs privés

- Parametrage ∗ parametrage
- Video ∗ video
- Parametres parametres
- int numeroCamera
- QString identifiant

9.3.1 Description détaillée

Auteur

Pierre GRELET

Version

1.1

Date

15 mars 2018

9.3.2 Documentation des constructeurs et destructeur

9.3.2.1 Camera : :Camera ( )

Constructeur par défaut

Références parametrage, progresser(), progression(), terminer(), et video.

```
                : parametrage(NULL), video(NULL), numeroCamera(0), identifiant("
    ")
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    video = new Video;
    connect(video, SIGNAL(progression(int)), this, SLOT(progresser(int)));
    connect(video, SIGNAL(fini()), this, SLOT(terminer()));

    parametrage = new Parametrage;
}
```

9.3.2.2 Camera : :Camera (int *numeroCamera,* QString *identifiant*)

Paramètres

numero-Camera	int le numéro de la caméra
identifiant	QString le nom de la caméra

Références lireParametres(), parametrage, progresser(), progression(), terminer(), et video.

```
                                                : parametrage(NULL), video(
    NULL), numeroCamera(numeroCamera), identifiant(identifiant)
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    video = new Video;
    connect(video, SIGNAL(progression(int)), this, SLOT(progresser(int)));
    connect(video, SIGNAL(fini()), this, SLOT(terminer()));

    parametrage = new Parametrage;
    lireParametres();
}
```

9.3.2.3 Camera : :~Camera ( )

Références parametrage, et video.

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
```

```
    #endif

    delete parametrage;
    delete video;
}
```

9.3.3 Documentation des fonctions membres

9.3.3.1 void Camera : :afficherMessage (QString *message*) const [signal]

Référencé par enregistrer().

9.3.3.2 void Camera : :arreter ( ) const [slot]

L'administrateur a demandé d'arrêter l'acquisition

Références Video : :arreter(), et video.

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    video->arreter();
}
```

9.3.3.3 void Camera : :enregistrer ( ) const [slot]

Références Parametres : :adresseIP, afficherMessage(), Parametres : :duree, Video-: :enregistrerVideo(), Parametres : :etat, Parametres : :identifiant, Parametres : :motDe-Passe, Parametres : :nomSite, numeroCamera, Parametres : :numeroPort, parametres, et video.

```
{
    if(parametres.etat == "1")
    {
        emit afficherMessage(QString::fromUtf8("Démarrage enregistrement caméra
       %1 (durée %2 s)").arg(parametres.nomSite).arg(parametres.duree));
        video->enregistrerVideo(numeroCamera, parametres.nomSite, parametres.
      adresseIP, parametres.numeroPort, parametres.identifiant, parametres.motDePasse
      , parametres.duree.toInt());
    }

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << numeroCamera << parametres.nomSite << parametres
      .adresseIP << parametres.numeroPort;
    #endif
}
```

9.3.3.4 Parametres Camera : :getParametres ( ) const

Les paramètres de la caméra

Renvoie

un type *Parametres* qui est la structure de Parametres

Références Parametrage : :getParametres(), et parametrage.

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    return parametrage->getParametres();
}
```

9.3.3.5 void Camera : :lireParametres ( )

Références Parametrage : :getParametres(), Parametrage : :lireParametres(), numeroCamera, parametrage, et parametres.

Référencé par Camera().

```
{
    parametrage->lireParametres("Camera" + QString::number(numeroCamera));
    parametres = parametrage->getParametres();
}
```

9.3.3.6 void Camera : :progresser (int *pct*) const [slot]

Paramètres

pct	int la valeur de progression d'enregistrement

Références progression().

Référencé par Camera().

```
{
    #ifdef DEBUG
    //qDebug() << Q_FUNC_INFO;
    #endif

    emit progression(pct);
}
```

9.3.3.7 void Camera : :progression (int *chargement*) const [signal]

Référencé par Camera(), progresser(), et terminer().

9.3.3.8 QString Camera : :seDeplacer ( ) const

Renvoie

un type *QString* qui est la requête

Références Parametres : :adresseIP, Parametres : :identifiant, Parametres : :motDePasse, Parametres : :numeroPort, et parametres.

```
{
    //
      http://192.168.52.93:99/decoder_control.cgi?command=33&onestep=500&user=admin&pwd=
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif
```

```
    QString URL = "http://" + parametres.adresseIP + ":" + parametres.numeroPort
       + "/decoder_control.cgi?command=31&onestep=500" + "&user=" + parametres.
      identifiant + "&pwd=" + parametres.motDePasse;

    return URL;
}
```

9.3.3.9 void Camera : :setIdentifiant (QString *identifiant*)

9.3.3.10 void Camera : :setNumero (int *numeroCamera*)

9.3.3.11 void Camera : :setParametres (Parametres *parametres*)

Paramètres

parametres	Parametres les paramètres de la caméra

Références parametrage, et Parametrage : :setParametres().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    this->parametrage->setParametres(parametres);
}
```

9.3.3.12 void Camera : :supprimer ( ) const [slot]

Supprimer la vidéo la plus ancienne

Références Video : :supprimerEnregistrements(), et video.

```
{
    video->supprimerEnregistrements();
}
```

9.3.3.13 void Camera : :terminer ( ) const [slot]

Références progression().

Référencé par Camera().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    progression(0);
}
```

9.3.4 Documentation des données membres

9.3.4.1 QString Camera : :identifiant [private]

9.3.4.2 int Camera : :numeroCamera [private]

Référencé par enregistrer(), et lireParametres().

9.3.4.3 Parametrage* Camera : :parametrage [private]

Référencé par Camera(), getParametres(), lireParametres(), setParametres(), et ~Camera().

9.3.4.4 Parametres Camera : :parametres [private]

Référencé par enregistrer(), lireParametres(), et seDeplacer().

9.3.4.5 Video* Camera : :video [private]

Référencé par arreter(), Camera(), enregistrer(), supprimer(), et ~Camera().

La documentation de cette classe a été générée à partir des fichiers suivants :
– Camera.h
– Camera.cpp

9.4 Référence de la structure CONFIG_ENREGISTREMENT

`#include <structures.h>`

Attributs publics

– int periode

9.4.1 Documentation des données membres

9.4.1.1 int CONFIG_ENREGISTREMENT : :periode

Référencé par WISMASIHM : :chargerFichierConfig().

La documentation de cette structure a été générée à partir du fichier suivant :
– structures.h

9.5 Référence de la structure CONFIG_LOGICIEL

fixe la configuration de départ de l'application.

`#include <structures.h>`

Attributs publics

– int nb_station
– long periode

9.5.1 Documentation des données membres

9.5.1.1 int CONFIG_LOGICIEL : :nb_station

défini le nombre de station

Référencé par WISMASIHM : :chargerFichierConfig().

9.5.1.2 long CONFIG_LOGICIEL : :periode

change l'affichage d'une station en seconde

Référencé par WISMASIHM : :chargerFichierConfig(), et WISMASIHM : :rafraichir().

La documentation de cette structure a été générée à partir du fichier suivant :
– structures.h

9.6 Référence de la structure DONNEES_STATION

fixe les données d'un station

```
#include <structures.h>
```

Attributs publics

– QString dateDonnes
– QString heureDonnes
– QString tarifs
– QString horaire
– QString nbPisteOuverte
– QString directionVent
– QString vitesseVent
– QString temperatureAir
– QString temperatureNeige
– QString hauteurNeige
– QString humidite
– QString pressionAir

9.6.1 Documentation des données membres

9.6.1.1 QString DONNEES_STATION : :dateDonnes

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), et StationMeteo : :preparerDonnees().

9.6.1.2 QString DONNEES_STATION : :directionVent

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

9.6.1.3 QString DONNEES_STATION : :hauteurNeige

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

9.6.1.4 QString DONNEES_STATION : :heureDonnes

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), et StationMeteo : :preparerDonnees().

9.6.1.5 QString DONNEES_STATION : :horaire

9.6.1.6 QString DONNEES_STATION : :humidite

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

9.6.1.7 QString DONNEES_STATION : :nbPisteOuverte

9.6.1.8 QString DONNEES_STATION : :pressionAir

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

9.6.1.9 QString DONNEES_STATION : :tarifs

9.6.1.10 QString DONNEES_STATION : :temperatureAir

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

9.6.1.11 QString DONNEES_STATION : :temperatureNeige

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

9.6.1.12 QString DONNEES_STATION : :vitesseVent

Référencé par WISMASProtocole : :decoderTrame(), StationMeteo : :enregistrerDonnees(), StationMeteo : :preparerDonnees(), et StationMeteo : :setDonneesTrame().

La documentation de cette structure a été générée à partir du fichier suivant :
– structures.h

9.7 Référence de la classe IHMWismas

Interface administrateur pour gérer le système vidéo.

`#include <IHMWismas.h>`

Connecteurs publics

– void parametrer ()
  Afficher la fenêtre de paramétrage des caméras.
– void demarrer ()
  Afficher la fenêtre d'acquisition des vidéos.
– void gererBoutonEtat ()

Gérer l'état des caméras au niveau de l'interface.
- void gererTypeDeplacement () const
  Gérer les déplacements au niveau de l'interface.
- void activerBoutonsCamera () const
  Activer le type de déplacement fixe.
- void desactiverBoutonsCamera () const
  Activer le type de déplacement panoramique.
- void deplacerGauche () const
  Déplacer la caméra à gauche.
- void deplacerDroite () const
  Déplacer la caméra à droite.
- void deplacerHaut () const
  Déplacer la caméra en haut.
- void deplacerBas () const
  Déplacer la caméra en bas.
- void sauvegarderParametres ()
  Sauvegarder les paramètres entrés via l'IHM sur le fichier de configuration .ini.
- void afficherParametresCamera () const
  Afficher les paramètres d'une caméra.
- void rafraichirFluxVideo (const QString identifiantCamera, Parametres &parametres)
  Rafraîchir les paramètres caméra.
- void afficherFluxVideo () const
  Afficher le flux vidéo de la caméra sélectionnée.
- void sauvegarderPositionInitiale () const
  Paramétrer la position initiale de la caméra.
- void sauvegarderPositionFinale () const
  Paramétrer la position finale de la caméra.
- void initialiserPosition () const
  Paramétrer la position finale de la caméra.
- void demarrerAcquisitionVideo ()
  Lancer la vidéo.
- void deplacerPositionInitiale ()
  Déplacer la prise vidéo position intiale.
- void deplacerPositionFinale ()
  Déplacer la prise vidéo position finale.
- void arreterAcquisitionVideo ()
  Arrêter la prise vidéo.
- void afficherMessage (const QString message) const
  Afficher des messages sur les enregistrements.

Signaux

- void fini ()

Fonctions membres publiques

- IHMWismas (QWidget *parent=0)
  Constructeur de la calsse IHMWismas.
- ~IHMWismas ()
  Destructeur.

Fonctions membres privées

- void connecter () const

Connecter les signaux aux slots.
- void creerCameras ()
  Lire les paramètres du fichier de configuration .ini pour créer les caméras ainsi que leur timer.
- QUrl creerUrlCamera () const
  Créer un url caméra pour récupérer son flux vidéo.
- void demarrerTimers (const int numeroCamera, const Parametres &parametres)
  Démarrer la période des caméras.
- void arreterTimers () const
  Arrêter la période des caméras.

Attributs privés

- Ui : :IHMWismas * ui
  relation vers la classe IHM
- QString fichierINI
  le fichier de parametrage
- QVector< QTimer * > timerCamera
  les timers de chaque caméra
- QTimer * timerInitial
- QTimer * timerFinal
- QVector< Camera * > cameras
  les caméras à gérer
- QNetworkAccessManager * manager
  pour émettre des demandes de deplacement de la camera
- QAction * actionQuitter

9.7.1 Description détaillée

La fenêtre principale de l'application WISMAS_video.

Auteur

Pierre GRELET

Version

1.1

Date

18 février 2018

9.7.2 Documentation des constructeurs et destructeur

9.7.2.1 IHMWismas : :IHMWismas (QWidget * *parent* = 0) [explicit]

Paramètres

parent	QObject Adresse objet Qt parent (ici 0 pour une fenêtre principale)

Références actionQuitter, afficherFluxVideo(), connecter(), creerCameras(), fichierINI, initialiserPosition(), manager, parametrer(), timerFinal, timerInitial, et ui.

```
                                    : QWidget(parent), ui(new Ui::IHMWismas)
{
    ui->setupUi(this);

    fichierINI = QApplication::applicationDirPath() + "/" + "
      configuration-cameras.ini";

    // Les timers pour les déplacements périodiques
    timerInitial = new QTimer(this);
    timerFinal = new QTimer(this);
    manager = new QNetworkAccessManager(this);

    actionQuitter = new QAction(this);
    actionQuitter->setShortcut(QKeySequence(QKeySequence::Quit));
    addAction(actionQuitter);

    creerCameras();
    afficherFluxVideo();
    initialiserPosition();

    // Connecte les signaux aux slots
    connecter();

    // Affiche la fenêtre de paramétrage des caméras
    parametrer();

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << fichierINI;
    #endif
}
```

9.7.2.2 IHMWismas : :~IHMWismas ( )

Références arreterAcquisitionVideo(), Video : :arreterEnregistrements(), cameras, et ui.

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    arreterAcquisitionVideo();

    for(int compteur = 0; compteur < cameras.count(); compteur++)
    {
        Camera* camera = cameras.at(compteur);
        delete camera;
    }

    Video::arreterEnregistrements();

    delete ui;
}
```

9.7.3 Documentation des fonctions membres

9.7.3.1 void IHMWismas : :activerBoutonsCamera ( ) const [slot]

Références ui.

Référencé par afficherParametresCamera(), gererTypeDeplacement(), et sauvegarderParametres().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << "Déplacer caméra : activé";
```

```
    #endif

    ui->deplacementGaucheParametrage->setEnabled(true);
    ui->deplacementDroiteParametrage->setEnabled(true);
    ui->deplacementHautParametrage->setEnabled(true);
    ui->deplacementBasParametrage->setEnabled(true);
    ui->enregistrerPositionInitiale->setEnabled(true);
    ui->enregistrerPositionFinale->setEnabled(false);
}
```

9.7.3.2 void IHMWismas : :afficherFluxVideo ( ) const [slot]

Références cameras, creerUrlCamera(), Parametres : :resolution, et ui.

Référencé par connecter(), et IHMWismas().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();
    Parametres parametres = cameras.at(numeroCamera)->getParametres();

    int resolution = parametres.resolution.toInt();

    switch(resolution)
    {
        case 480:
            ui->visualisationParametrage->setMaximumSize(720,480);
            break;
        case 720:
            ui->visualisationParametrage->setMaximumSize(1280, 720);
            break;
        default:
            ui->visualisationParametrage->setMaximumSize(720, 480);
            break;
    }
    ui->visualisationParametrage->load(creerUrlCamera());
}
```

9.7.3.3 void IHMWismas : :afficherMessage (const QString *message*) const [slot]

Paramètres

message	QString log sur les enregistrements vidéo

Références ui.

Référencé par arreterAcquisitionVideo(), creerCameras(), et demarrerAcquisitionVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    ui->messagesEnregistrement->append(message);
}
```

9.7.3.4 void IHMWismas : :afficherParametresCamera ( ) const [slot]

Références activerBoutonsCamera(), Parametres : :adresseIP, cameras, desactiverBoutonsCamera(), Parametres : :duree, Parametres : :etat, Parametres : :numeroPort, Parametres : :periode, Parametres : :typeDeplacement, et ui.

Référencé par connecter(), gererBoutonEtat(), et parametrer().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();

        if(parametres.etat == "0")
        {
            ui->parametres->setDisabled(true);
            ui->acquisition->setDisabled(true);
            ui->visualisationParametrage->stop();
            ui->etat->setChecked(false);
            ui->etat->setText("Désactivé");
        }
        else
        {
            ui->parametres->setEnabled(true);
            ui->acquisition->setEnabled(true);
            ui->etat->setChecked(true);
            ui->etat->setText("Activé");
        }

        ui->champAdresseIP->setText(parametres.adresseIP);
        ui->champPort->setText(parametres.numeroPort);
        ui->champDuree->setText(parametres.duree);
        ui->champPeriode->setText(parametres.periode);

        if(parametres.typeDeplacement == "Panoramique")
        {
            ui->typeDeplacementPanoramique->setChecked(true);
            activerBoutonsCamera();
        }
        else if(parametres.typeDeplacement == "Fixe")
        {
            ui->typeDeplacementFixe->setChecked(true);
            desactiverBoutonsCamera();
        }
        else
        {
            ui->typeDeplacementFixe->setChecked(true);
            desactiverBoutonsCamera();
        }
    }
}
```

9.7.3.5 void IHMWismas : :arreterAcquisitionVideo ( ) [slot]

La vidéo est arrêtée par l'administrateur

Références afficherMessage(), arreterTimers(), cameras, Parametres : :etat, fini(), initialiserPosition(), Parametres : :nomSite, et ui.

Référencé par connecter(), et ∼IHMWismas().

```
{
```

```
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    ui->parametrage->setEnabled(true);
    ui->demarrageAcquisitionVideo->setEnabled(true);
    ui->arretAcquisitionVideo->setEnabled(false);

    arreterTimers();

    emit fini(); // pour pévenir les caméras

    for(int compteur = 0; compteur < cameras.count(); compteur++)
    {
        Parametres parametres = cameras.at(compteur)->getParametres();

        if(parametres.etat == "0")
            afficherMessage(QString::fromUtf8("Fin de la programmation des
       enregistrements caméra %1").arg(parametres.nomSite));
    }
    initialiserPosition();
}
```

9.7.3.6 void IHMWismas : :arreterTimers ( ) const [private]

Arrêt du système d'acquisition vidéo

Références fichierINI, timerCamera, timerFinal, et timerInitial.

Référencé par arreterAcquisitionVideo(), et demarrerAcquisitionVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    QSettings configuration(fichierINI, QSettings::IniFormat);

    int nbTimers = configuration.value("nb_cameras", "0").toInt();

    for(int compteur = 0; compteur < nbTimers; compteur++)
    {
        if(timerCamera.at(compteur)->isActive())
            timerCamera.at(compteur)->stop();
    }
    if(timerInitial->isActive())
        timerInitial->stop();
    if(timerFinal->isActive())
        timerFinal->stop();
}
```

9.7.3.7 void IHMWismas : :connecter ( ) const [private]

Références actionQuitter, afficherFluxVideo(), afficherParametresCamera(), arreterAcquisitionVideo(), demarrer(), demarrerAcquisitionVideo(), deplacerBas(), deplacerDroite(), deplacerGauche(), deplacerHaut(), gererBoutonEtat(), gererTypeDeplacement(), parametrer(), sauvegarderParametres(), sauvegarderPositionFinale(), sauvegarderPositionInitiale(), et ui.

Référencé par IHMWismas().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
```

```
    #endif

    connect(actionQuitter, SIGNAL(triggered()), this, SLOT(close()));

    connect(ui->parametrage, SIGNAL(clicked()), this, SLOT(parametrer()));
    connect(ui->acquisition, SIGNAL(clicked()), this, SLOT(demarrer()));

    connect(ui->listeConfigurationCameras, SIGNAL(currentIndexChanged(int)),
      this, SLOT(afficherFluxVideo()));
    connect(ui->listeConfigurationCameras, SIGNAL(currentIndexChanged(int)),
      this, SLOT(afficherParametresCamera()));
    connect(ui->sauvegarde, SIGNAL(clicked()), this, SLOT(sauvegarderParametres
      ()));

    connect(ui->etat, SIGNAL(stateChanged(int)), this, SLOT(gererBoutonEtat()))
      ;

    connect(ui->typeDeplacementPanoramique, SIGNAL(clicked()), this, SLOT(
      gererTypeDeplacement()));
    connect(ui->typeDeplacementFixe, SIGNAL(clicked()), this, SLOT(
      gererTypeDeplacement()));
    connect(ui->deplacementGaucheParametrage, SIGNAL(clicked()), this, SLOT(
      deplacerGauche()));
    connect(ui->deplacementDroiteParametrage, SIGNAL(clicked()), this, SLOT(
      deplacerDroite()));
    connect(ui->deplacementHautParametrage, SIGNAL(clicked()), this, SLOT(
      deplacerHaut()));
    connect(ui->deplacementBasParametrage, SIGNAL(clicked()), this, SLOT(
      deplacerBas()));

    connect(ui->enregistrerPositionInitiale, SIGNAL(clicked()), this, SLOT(
      sauvegarderPositionInitiale()));
    connect(ui->enregistrerPositionFinale, SIGNAL(clicked()), this, SLOT(
      sauvegarderPositionFinale()));

    connect(ui->demarrageAcquisitionVideo, SIGNAL(clicked()), this, SLOT(
      demarrerAcquisitionVideo()));
    connect(ui->arretAcquisitionVideo, SIGNAL(clicked()), this, SLOT(
      arreterAcquisitionVideo()));
}
```

9.7.3.8 void IHMWismas : :creerCameras ( ) [private]

Références actionQuitter, afficherMessage(), cameras, fichierINI, fini(), timerCamera, et ui.

Référencé par IHMWismas().

```
{
    QSettings configuration(fichierINI, QSettings::IniFormat);
    int nbCameras = configuration.value("nb_cameras", "0").toInt();
    QString identifiant;
    QString nom;

    for(int compteur = 0; compteur < nbCameras; compteur++)
    {
        identifiant = configuration.value(QString("Camera%1").arg(compteur+1) +
       "/identifiant").toString();
        nom = configuration.value(QString("Camera%1").arg(compteur+1) + "/nom")
      .toString();

        #ifdef DEBUG
        qDebug() << Q_FUNC_INFO << identifiant << " " << nom;
        #endif

        if(!identifiant.isEmpty())
        {
            Camera* camera = new Camera(compteur+1, identifiant);
            cameras.push_back(camera);
```

```
            QTimer* timer = new QTimer();
            timerCamera.push_back(timer);

            ui->listeConfigurationCameras->addItem(nom);

            connect(this, SIGNAL(fini()), camera, SLOT(arreter()));
            connect(camera, SIGNAL(afficherMessage(QString)), this, SLOT(
    afficherMessage(QString)));
            connect(actionQuitter, SIGNAL(triggered()), camera, SLOT(supprimer(
    )));

            switch(compteur)
            {
            case 0:
                ui->labelEnregistrementCamera1->setText(QString::fromUtf8("
    Caméra %1 :").arg(nom));
                connect(timerCamera.at(compteur), SIGNAL(timeout()), camera,
    SLOT(enregistrer()));
                connect(camera, SIGNAL(progression(int)), ui->
    progessionEnregistrementCamera1, SLOT(setValue(int)));
                break;
            case 1:
                ui->labelEnregistrementCamera2->setText(QString::fromUtf8("
    Caméra %1 :").arg(nom));
                connect(timerCamera.at(compteur), SIGNAL(timeout()), camera,
    SLOT(enregistrer()));
                connect(camera, SIGNAL(progression(int)), ui->
    progessionEnregistrementCamera2, SLOT(setValue(int)));
                break;
            case 2:
                ui->labelEnregistrementCamera3->setText(QString::fromUtf8("
    Caméra %1 :").arg(nom));
                connect(timerCamera.at(compteur), SIGNAL(timeout()), camera,
    SLOT(enregistrer()));
                connect(camera, SIGNAL(progression(int)), ui->
    progessionEnregistrementCamera3, SLOT(setValue(int)));
                break;
            }
        }
    }
}
```

9.7.3.9 QUrl IHMWismas : :creerUrlCamera ( ) const [private]

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, Parametres-: :motDePasse, Parametres : :numeroPort, et ui.

Référencé par afficherFluxVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();
    Parametres parametres = cameras.at(numeroCamera)->getParametres();

    QUrl URL("http://" + parametres.adresseIP + ":" + parametres.numeroPort + "
      /mobile.htm");
    URL.setUserName(parametres.identifiant);
    URL.setPassword(parametres.motDePasse);

    return URL;
}
```

9.7.3.10 void IHMWismas : :demarrer ( ) [slot]

Références demarrerAcquisitionVideo(), et ui.

Référencé par connecter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    ui->fenetre->setCurrentIndex(1);

    demarrerAcquisitionVideo();
}
```

9.7.3.11 void IHMWismas : :demarrerAcquisitionVideo ( ) [slot]

La vidéo est lancée pour une durée prédéfinie

Références afficherMessage(), arreterTimers(), cameras, demarrerTimers(), deplacerPositionFinale(), Parametres : :etat, Parametres : :nomSite, Parametres : :periode, timerFinal, et ui.

Référencé par connecter(), et demarrer().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    ui->progessionEnregistrementCamera1->setValue(0);
    ui->progessionEnregistrementCamera2->setValue(0);
    ui->progessionEnregistrementCamera3->setValue(0);
    ui->parametrage->setEnabled(false);
    ui->demarrageAcquisitionVideo->setEnabled(false);
    ui->arretAcquisitionVideo->setEnabled(true);

    arreterTimers();

    timerFinal->singleShot(2500, this, SLOT(deplacerPositionFinale()));

    for(int compteur = 0; compteur < cameras.count(); compteur++)
    {
        Parametres parametres = cameras.at(compteur)->getParametres();

        if(parametres.etat == "1")
        {
            afficherMessage(QString::fromUtf8("Programmation enregistrement
       caméra %1 : toutes les %2  s").arg(parametres.nomSite).arg(parametres.periode));
            cameras.at(compteur)->enregistrer();
            demarrerTimers(compteur, parametres);
        }
    }
}
```

9.7.3.12 void IHMWismas : :demarrerTimers (const int *numeroCamera,* const Parametres & *parametres*) [private]

A une période 0, l'acquisition vidéo est à nouveau lancée

Paramètres

numero-Camera	int le numéro de la caméra
parametres	Parametres les paramètres de la caméra

Références Parametres : :periode, et timerCamera.

Référencé par demarrerAcquisitionVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    timerCamera.at(numeroCamera)->start(parametres.periode.toInt()*1000);
}
```

9.7.3.13 void IHMWismas : :deplacerBas ( ) const [slot]

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, manager, Parametres : :motDePasse, Parametres : :numeroPort, et ui.

Référencé par connecter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=2&onestep=1" + "&user=" + parametres
      .identifiant + "&pwd=" + parametres.motDePasse;
        manager->get(QNetworkRequest(QUrl(URL)));
    }
}
```

9.7.3.14 void IHMWismas : :deplacerDroite ( ) const [slot]

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, manager, Parametres : :motDePasse, Parametres : :numeroPort, et ui.

Référencé par connecter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=6&onestep=1" + "&user=" + parametres
      .identifiant + "&pwd=" + parametres.motDePasse;
        manager->get(QNetworkRequest(QUrl(URL)));
    }
}
```

9.7.3.15 void IHMWismas : :deplacerGauche ( ) const [slot]

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, manager, Parametres : :motDePasse, Parametres : :numeroPort, et ui.

Référencé par connecter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=4&onestep=1" + "&user=" + parametres
      .identifiant + "&pwd=" + parametres.motDePasse;
        manager->get(QNetworkRequest(QUrl(URL)));
    }
}
```

9.7.3.16 void IHMWismas : :deplacerHaut ( ) const [slot]

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, manager, Parametres : :motDePasse, Parametres : :numeroPort, et ui.

Référencé par connecter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=0&onestep=1" + "&user=" + parametres
      .identifiant + "&pwd=" + parametres.motDePasse;
        manager->get(QNetworkRequest(QUrl(URL)));
    }
}
```

9.7.3.17 void IHMWismas : :deplacerPositionFinale ( ) [slot]

Pour un déplacement panoramique

Références Parametres : :adresseIP, cameras, deplacerPositionInitiale(), Parametres : :identifiant, manager, Parametres : :motDePasse, Parametres : :numeroPort, et timerInitial.

Référencé par demarrerAcquisitionVideo(), et deplacerPositionInitiale().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    for(int compteur = 0; compteur < cameras.count(); compteur++)
    {
        Parametres parametres = cameras.at(compteur)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=33&onestep=500" + "&user=" +
      parametres.identifiant + "&pwd=" + parametres.motDePasse;
```

```
        manager->get(QNetworkRequest(QUrl(URL)));
    }

    timerInitial->singleShot(10000, this, SLOT(deplacerPositionInitiale()));
}
```

9.7.3.18 void IHMWismas : :deplacerPositionInitiale ( ) [slot]

Pour un déplacement panoramique

Références cameras, deplacerPositionFinale(), manager, et timerFinal.

Référencé par deplacerPositionFinale().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    for(int compteur = 0; compteur < cameras.count(); compteur++)
    {
        QString URL = cameras.at(compteur)->seDeplacer();
        manager->get(QNetworkRequest(QUrl(URL)));
    }

    timerFinal->singleShot(15000, this, SLOT(deplacerPositionFinale()));
}
```

9.7.3.19 void IHMWismas : :desactiverBoutonsCamera ( ) const [slot]

Références ui.

Référencé par afficherParametresCamera(), gererTypeDeplacement(), et sauvegarderParametres().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << "Déplacer caméra : désactivé";
    #endif

    ui->deplacementGaucheParametrage->setEnabled(true);
    ui->deplacementDroiteParametrage->setEnabled(true);
    ui->deplacementHautParametrage->setEnabled(true);
    ui->deplacementBasParametrage->setEnabled(true);
    ui->enregistrerPositionInitiale->setEnabled(false);
    ui->enregistrerPositionFinale->setEnabled(false);
}
```

9.7.3.20 void IHMWismas : :fini ( ) [signal]

Référencé par arreterAcquisitionVideo(), et creerCameras().

9.7.3.21 void IHMWismas : :gererBoutonEtat ( ) [slot]

Activer/désactiver la caméra

Références afficherParametresCamera(), cameras, fichierINI, rafraichirFluxVideo(), et ui.

Référencé par connecter().

```
{
    int numeroCamera = ui->listeConfigurationCameras->currentIndex();
    Parametres parametres = cameras.at(numeroCamera)->getParametres();
    QString identifiantCamera = "Camera" + QString::number(numeroCamera+1);

    QSettings configuration(fichierINI, QSettings::IniFormat);

    if(ui->etat->isChecked())
    {
        ui->etat->setText("Activé");
        configuration.setValue(identifiantCamera + "/etat", "1");
    }
    else
    {
        ui->etat->setText("Désactivé");
        configuration.setValue(identifiantCamera + "/etat", "0");;
    }
    rafraichirFluxVideo(identifiantCamera, parametres);
    cameras.at(numeroCamera)->lireParametres();
    afficherParametresCamera();
}
```

9.7.3.22 void IHMWismas : :gererTypeDeplacement ( ) const [slot]

Activer/désactiver boutons de déplacement

Références activerBoutonsCamera(), desactiverBoutonsCamera(), et ui.

Référencé par connecter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    if(ui->typeDeplacementPanoramique-> isChecked())
        activerBoutonsCamera();
    else

        desactiverBoutonsCamera();
}
```

9.7.3.23 void IHMWismas : :initialiserPosition ( ) const [slot]

Pour un déplacement panoramique

Références cameras, et manager.

Référencé par arreterAcquisitionVideo(), IHMWismas(), et sauvegarderPositionFinale().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    for(int compteur = 0; compteur < cameras.count(); compteur++)
    {
        QString URL = cameras.at(compteur)->seDeplacer();
        manager->get(QNetworkRequest(QUrl(URL)));
    }
}
```

9.7.3.24 void IHMWismas : :parametrer ( ) [slot]

Références afficherParametresCamera(), et ui.

Référencé par connecter(), et IHMWismas().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    ui->fenetre->setCurrentIndex(0);

    afficherParametresCamera();
}
```

9.7.3.25 void IHMWismas : :rafraichirFluxVideo (const QString *identifiantCamera*, Parametres & *parametres*) [slot]

Une modification des paramètres qui est ensuite sauvegardée

Paramètres

identifiant-Camera	QString le nom de la caméra
parametres	Parametres les paramètres de la caméra

Références Parametres : :adresseIP, fichierINI, Parametres : :identifiant, Parametres-: :motDePasse, Parametres : :numeroPort, et ui.

Référencé par gererBoutonEtat(), et sauvegarderParametres().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    QSettings configuration(fichierINI, QSettings::IniFormat);
    parametres.adresseIP = configuration.value(identifiantCamera + "/adresse_IP
      ", "").toString();
    parametres.numeroPort = configuration.value(identifiantCamera + "
      /numero_port","").toString();

    QUrl URL("http://" + parametres.adresseIP + ":" + parametres.numeroPort + "
      /mobile.htm");
    URL.setUserName(parametres.identifiant);
    URL.setPassword(parametres.motDePasse);

    ui->visualisationParametrage->load(URL);
    ui->visualisationParametrage->update();
}
```

9.7.3.26 void IHMWismas : :sauvegarderParametres ( ) [slot]

Références activerBoutonsCamera(), cameras, desactiverBoutonsCamera(), fichierINI, rafraichirFluxVideo(), sauvegarderPositionFinale(), sauvegarderPositionInitiale(), et ui.

Référencé par connecter().

```
{
    int numeroCamera = ui->listeConfigurationCameras->currentIndex();
    QString identifiantCamera = "Camera" + QString::number(numeroCamera+1);
    Parametres parametres = cameras.at(numeroCamera)->getParametres();
    QSettings configuration(fichierINI, QSettings::IniFormat);
```

```
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << identifiantCamera;
    #endif

    configuration.setValue(identifiantCamera + "/adresse_IP", ui->
      champAdresseIP->text());
    configuration.setValue(identifiantCamera + "/numero_port", ui->champPort->
      text());
    configuration.setValue(identifiantCamera + "/duree", ui->champDuree->text()
      );
    configuration.setValue(identifiantCamera + "/periode", ui->champPeriode->
      text());

    if(ui->typeDeplacementFixe->isChecked())
    {
        desactiverBoutonsCamera();

        configuration.setValue(identifiantCamera + "/type_deplacement", "Fixe")
      ;
        sauvegarderPositionInitiale();
        sauvegarderPositionFinale();
    }
    if(ui->typeDeplacementPanoramique->isChecked())
    {
        activerBoutonsCamera();
        configuration.setValue(identifiantCamera + "/type_deplacement", "
      Panoramique");
    }

    rafraichirFluxVideo(identifiantCamera, parametres);
    cameras.at(numeroCamera)->lireParametres();
}
```

9.7.3.27 void IHMWismas : :sauvegarderPositionFinale ( ) const [slot]

Pour un déplacement panoramique

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, initialiserPosition(), manager, Parametres : :motDePasse, Parametres : :numeroPort, et ui.

Référencé par connecter(), et sauvegarderParametres().

```
{
    //ENREGISTRER POSITION DETECTION
    //
      http://192.168.52.93:99/decoder_control.cgi?command=32&onestep=500&user=admin&pwd=
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=32&onestep=500" + "&user=" +
      parametres.identifiant + "&pwd=" + parametres.motDePasse;
        manager->get(QNetworkRequest(QUrl(URL)));
    }
    ui->enregistrerPositionInitiale->setEnabled(true);
    ui->enregistrerPositionFinale->setEnabled(false);

    initialiserPosition();
}
```

9.7.3.28 void IHMWismas : :sauvegarderPositionInitiale ( ) const [slot]

Pour un déplacement panoramique

Références Parametres : :adresseIP, cameras, Parametres : :identifiant, manager, Parametres : :motDePasse, Parametres : :numeroPort, et ui.

Référencé par connecter(), et sauvegarderParametres().

```
{
    //ENREGISTRER POSITION INITIALE
    //
      http://192.168.52.93:99/decoder_control.cgi?command=30&onestep=500&user=admin&pwd=
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    int numeroCamera = ui->listeConfigurationCameras->currentIndex();

    if(numeroCamera >=0 && numeroCamera < cameras.count())
    {
        Parametres parametres = cameras.at(numeroCamera)->getParametres();
        QString URL = "http://" + parametres.adresseIP + ":" + parametres.
      numeroPort + "/decoder_control.cgi?command=30&onestep=500" + "&user=" +
      parametres.identifiant + "&pwd=" + parametres.motDePasse;
        manager->get(QNetworkRequest(QUrl(URL)));
    }
    ui->enregistrerPositionInitiale->setEnabled(false);
    ui->enregistrerPositionFinale->setEnabled(true);
}
```

9.7.4 Documentation des données membres

9.7.4.1 QAction* IHMWismas : :actionQuitter [private]

Référencé par connecter(), creerCameras(), et IHMWismas().

9.7.4.2 QVector<Camera*> IHMWismas : :cameras [private]

Référencé par afficherFluxVideo(), afficherParametresCamera(), arreterAcquisitionVideo(), creerCameras(), creerUrlCamera(), demarrerAcquisitionVideo(), deplacerBas(), deplacerDroite(), deplacerGauche(), deplacerHaut(), deplacerPositionFinale(), deplacerPositionInitiale(), gererBoutonEtat(), initialiserPosition(), sauvegarderParametres(), sauvegarderPositionFinale(), sauvegarderPositionInitiale(), et ~IHMWismas().

9.7.4.3 QString IHMWismas : :fichierINI [private]

Référencé par arreterTimers(), creerCameras(), gererBoutonEtat(), IHMWismas(), rafraichirFluxVideo(), et sauvegarderParametres().

9.7.4.4 QNetworkAccessManager* IHMWismas : :manager [private]

Référencé par deplacerBas(), deplacerDroite(), deplacerGauche(), deplacerHaut(), deplacerPositionFinale(), deplacerPositionInitiale(), IHMWismas(), initialiserPosition(), sauvegarderPositionFinale(), et sauvegarderPositionInitiale().

9.7.4.5 QVector<QTimer*> IHMWismas : :timerCamera [private]

Référencé par arreterTimers(), creerCameras(), et demarrerTimers().

9.7.4.6 QTimer* IHMWismas : :timerFinal [private]

Référencé par arreterTimers(), demarrerAcquisitionVideo(), deplacerPositionInitiale(), et IHMWismas().

9.7.4.7 QTimer* IHMWismas : :timerInitial [private]

Référencé par arreterTimers(), deplacerPositionFinale(), et IHMWismas().

9.7.4.8 Ui : :IHMWismas* IHMWismas : :ui [private]

Référencé par activerBoutonsCamera(), afficherFluxVideo(), afficherMessage(), afficherParametresCamera(), arreterAcquisitionVideo(), connecter(), creerCameras(), creerUrlCamera(), demarrer(), demarrerAcquisitionVideo(), deplacerBas(), deplacerDroite(), deplacerGauche(), deplacerHaut(), desactiverBoutonsCamera(), gererBoutonEtat(), gererTypeDeplacement(), IHMWismas(), parametrer(), rafraichirFluxVideo(), sauvegarderParametres(), sauvegarderPositionFinale(), sauvegarderPositionInitiale(), et ~IHMWismas().

La documentation de cette classe a été générée à partir des fichiers suivants :
- IHMWismas.h
- IHMWismas.cpp

9.8 Référence de la structure PARAM_COMMUNICATION

`#include <structures.h>`

Attributs publics

- QString port
- int baud_rate
- int data_bit
- int stop
- int parity

9.8.1 Documentation des données membres

9.8.1.1 int PARAM_COMMUNICATION : :baud_rate

Référencé par WISMASIHM : :chargerFichierConfig().

9.8.1.2 int PARAM_COMMUNICATION : :data_bit

Référencé par WISMASIHM : :chargerFichierConfig().

9.8.1.3 int PARAM_COMMUNICATION : :parity

Référencé par WISMASIHM : :chargerFichierConfig().

9.8.1.4 QString PARAM_COMMUNICATION : :port

nom du port serie d'entrée du périphérique

Référencé par WISMASIHM : :chargerFichierConfig().

9.8.1.5 int PARAM_COMMUNICATION : :stop

Référencé par WISMASIHM : :chargerFichierConfig().

La documentation de cette structure a été générée à partir du fichier suivant :
– structures.h

9.9 Référence de la structure PARAM_PANNEAU

fixe les paramètres du périphérique panneau lumineux. port QString, int baud_rate, int data_bit, int stop, int parity

```
#include <structures.h>
```

Attributs publics

– QString port
– int baud_rate
– int data_bit
– int stop
– int parity

9.9.1 Documentation des données membres

9.9.1.1 int PARAM_PANNEAU : :baud_rate

Référencé par WISMASIHM : :chargerFichierConfig().

9.9.1.2 int PARAM_PANNEAU : :data_bit

Référencé par WISMASIHM : :chargerFichierConfig().

9.9.1.3 int PARAM_PANNEAU : :parity

Référencé par WISMASIHM : :chargerFichierConfig().

9.9.1.4 QString PARAM_PANNEAU : :port

nom du port serie d'entrée du panneau lumineux

Référencé par WISMASIHM : :chargerFichierConfig().

9.9.1.5 int PARAM_PANNEAU : :stop

Référencé par WISMASIHM : :chargerFichierConfig().

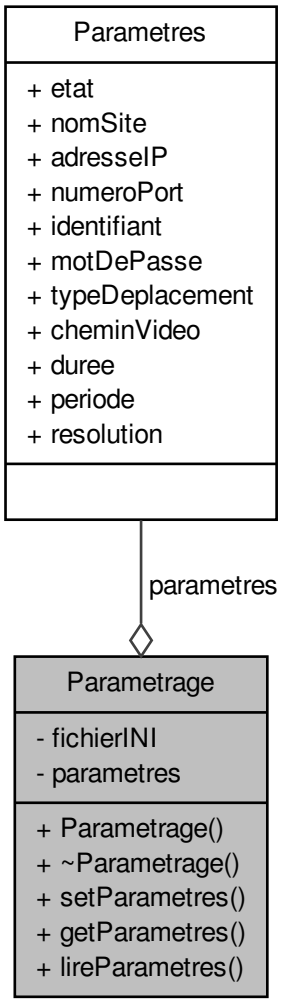
La documentation de cette structure a été générée à partir du fichier suivant :
– structures.h

9.10 Référence de la classe Parametrage

Paramétrer le système vidéo.

`#include <Parametrage.h>`

Graphe de collaboration de Parametrage :



Fonctions membres publiques

- Parametrage ()
  Constructeur.
- ~Parametrage ()
  Destructeur.
- void setParametres (Parametres &parametres)
  Mutateur de la structure parametres.

- Parametres getParametres () const
  Accesseur de la structure parametres.
- void lireParametres (QString identifiantCamera)
  Attribuer les paramètres à la caméra sélectionnée.

Attributs privés

- QString fichierINI
- Parametres parametres

9.10.1 Description détaillée

Auteur

Pierre GRELET

Version

1.1

Date

15 mars 2018

9.10.2 Documentation des constructeurs et destructeur

9.10.2.1 Parametrage : :Parametrage ( )

Références fichierINI.

```
{
    fichierINI = QApplication::applicationDirPath() + "/" + "
      configuration-cameras.ini";

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << fichierINI;
    #endif
}
```

9.10.2.2 Parametrage : :~Parametrage ( )

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif
}
```

9.10.3 Documentation des fonctions membres

9.10.3.1 Parametres Parametrage : :getParametres ( ) const

Les paramètres de la caméra

Renvoie

un type *Parametres* qui est la structure de Parametres

Références parametres.

Référencé par Camera : :getParametres(), et Camera : :lireParametres().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    return parametres;
}
```

9.10.3.2 void Parametrage : :lireParametres (QString *identifiantCamera*)

Paramètres

identifiantCamera	QString la caméra à paramétrer

Références Parametres : :adresseIP, Parametres : :cheminVideo, Parametres : :duree, Parametres : :etat, fichierINI, Parametres : :identifiant, Parametres : :motDePasse, Parametres : :nomSite, Parametres : :numeroPort, parametres, Parametres : :periode, Parametres : :resolution, et Parametres : :typeDeplacement.

Référencé par Camera : :lireParametres().

```
{
    QSettings configuration(fichierINI, QSettings::IniFormat);

    parametres.etat = configuration.value(identifiantCamera + "/etat", "1").
      toString();
    parametres.nomSite = configuration.value(identifiantCamera + "/nom", "").
      toString();
    parametres.adresseIP = configuration.value(identifiantCamera + "/adresse_IP
      ", "").toString();
    parametres.numeroPort = configuration.value(identifiantCamera + "
      /numero_port","").toString();
    parametres.identifiant = configuration.value(identifiantCamera + "
      /identifiant", "admin").toString();
    parametres.motDePasse = configuration.value(identifiantCamera + "
      /mot_de_passe", "").toString();
    parametres.typeDeplacement = configuration.value(identifiantCamera + "
      /type_deplacement", "Fixe").toString();
    parametres.cheminVideo = configuration.value(identifiantCamera + "
      /chemin_video", "./videos").toString();
    parametres.duree = configuration.value(identifiantCamera + "/duree", "30").
      toString();
    parametres.periode = configuration.value(identifiantCamera +"/periode", "
      3600").toString();
    parametres.resolution = configuration.value(identifiantCamera + "
      /resolution", "480").toString();

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    qDebug() << "Etat : " + parametres.etat;
    qDebug() << "Camera courante : " + identifiantCamera;
    qDebug() << "Adresse IP : " + parametres.adresseIP;
    qDebug() << "Port : " + parametres.numeroPort;
    qDebug() << "Login : " + parametres.identifiant;
    qDebug() << "Mot de passe : " + parametres.motDePasse;
    qDebug() << "Chemin vidéo : " + parametres.cheminVideo;
    qDebug() << "Durée : " + parametres.duree;
```

```
    qDebug() << "Période : " + parametres.periode;
    qDebug() << "Résolution : " + parametres.resolution;
    qDebug() << "Type de déplacement : " + parametres.typeDeplacement;
    #endif
}
```

9.10.3.3 void Parametrage : :setParametres (Parametres & *parametres*)

Paramètres

parametres	Parametres les paramètres de la caméra

Références parametres.

Référencé par Camera : :setParametres().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    this->parametres = parametres;
}
```

9.10.4 Documentation des données membres

9.10.4.1 QString Parametrage : :fichierINI [private]

Référencé par lireParametres(), et Parametrage().

9.10.4.2 Parametres Parametrage : :parametres [private]

Référencé par getParametres(), lireParametres(), et setParametres().

La documentation de cette classe a été générée à partir des fichiers suivants :
- Parametrage.h
- Parametrage.cpp

9.11 Référence de la structure Parametres

paramètres d'une caméra

`#include <Structures.h>`

Attributs publics

- QString etat
- QString nomSite
- QString adresseIP
- QString numeroPort
- QString identifiant
- QString motDePasse
- QString typeDeplacement
- QString cheminVideo
- QString duree
- QString periode

– QString resolution

9.11.1 Documentation des données membres

9.11.1.1 QString Parametres : :adresseIP

Référencé par IHMWismas : :afficherParametresCamera(), IHMWismas : :creerUrlCamera(), IHMWismas : :deplacerBas(), IHMWismas : :deplacerDroite(), IHMWismas : :deplacerGauche(), IHMWismas : :deplacerHaut(), IHMWismas : :deplacerPositionFinale(), Camera : :enregistrer(), Parametrage : :lireParametres(), IHMWismas : :rafraichirFluxVideo(), IHMWismas : :sauvegarderPositionFinale(), IHMWismas : :sauvegarderPositionInitiale(), et Camera : :seDeplacer().

9.11.1.2 QString Parametres : :cheminVideo

Référencé par Parametrage : :lireParametres().

9.11.1.3 QString Parametres : :duree

Référencé par IHMWismas : :afficherParametresCamera(), Camera : :enregistrer(), et Parametrage : :lireParametres().

9.11.1.4 QString Parametres : :etat

Référencé par IHMWismas : :afficherParametresCamera(), IHMWismas : :arreterAcquisitionVideo(), IHMWismas : :demarrerAcquisitionVideo(), Camera : :enregistrer(), et Parametrage : :lireParametres().

9.11.1.5 QString Parametres : :identifiant

Référencé par IHMWismas : :creerUrlCamera(), IHMWismas : :deplacerBas(), IHMWismas : :deplacerDroite(), IHMWismas : :deplacerGauche(), IHMWismas : :deplacerHaut(), IHMWismas : :deplacerPositionFinale(), Camera : :enregistrer(), Parametrage : :lireParametres(), IHMWismas : :rafraichirFluxVideo(), IHMWismas : :sauvegarderPositionFinale(), IHMWismas : :sauvegarderPositionInitiale(), et Camera : :seDeplacer().

9.11.1.6 QString Parametres : :motDePasse

Référencé par IHMWismas : :creerUrlCamera(), IHMWismas : :deplacerBas(), IHMWismas : :deplacerDroite(), IHMWismas : :deplacerGauche(), IHMWismas : :deplacerHaut(), IHMWismas : :deplacerPositionFinale(), Camera : :enregistrer(), Parametrage : :lireParametres(), IHMWismas : :rafraichirFluxVideo(), IHMWismas : :sauvegarderPositionFinale(), IHMWismas : :sauvegarderPositionInitiale(), et Camera : :seDeplacer().

9.11.1.7 QString Parametres : :nomSite

Référencé par IHMWismas : :arreterAcquisitionVideo(), IHMWismas : :demarrerAcquisitionVideo(), Camera : :enregistrer(), et Parametrage : :lireParametres().

9.11.1.8 QString Parametres : :numeroPort

Référencé par IHMWismas : :afficherParametresCamera(), IHMWismas : :creerUrlCamera(), IHMWismas : :deplacerBas(), IHMWismas : :deplacerDroite(), IHMWismas : :deplacerGauche(), IHMWismas : :deplacerHaut(), IHMWismas : :deplacerPositionFinale(), Camera : :enregistrer(), Parametrage : :lireParametres(), IHMWismas : :rafraichirFluxVideo(), IHMWismas : :sauvegarderPositionFinale(), IHMWismas : :sauvegarderPositionInitiale(), et Camera : :seDeplacer().

9.11.1.9 QString Parametres : :periode

Référencé par IHMWismas : :afficherParametresCamera(), IHMWismas : :demarrerAcquisitionVideo(), IHMWismas : :demarrerTimers(), et Parametrage : :lireParametres().

9.11.1.10 QString Parametres : :resolution

Référencé par IHMWismas : :afficherFluxVideo(), et Parametrage : :lireParametres().

9.11.1.11 QString Parametres : :typeDeplacement

Référencé par IHMWismas : :afficherParametresCamera(), et Parametrage : :lireParametres().

La documentation de cette structure a été générée à partir du fichier suivant :
– Structures.h

9.12 Référence de la classe PortPanneau

Communique avec les stations météo des sites via un port série virtuel.

`#include <portpanneau.h>`

Connecteurs publics

– void afficherMessage (QString message)
– void changerPage (int nb_page)
– void lireTrame ()

Fonctions membres publiques

– PortPanneau (QObject ∗parent=0)
  Constructeur de la classe PortPanneau.
– ∼PortPanneau ()
  Destructeur.

Fonctions membres privées

– QString calculerChecksum (QString message)
– QString creerTrame (QStringList messages, int nb_page)
– void envoyerTrame (QString trame)
– bool lireAcquittement (QString trame)

lit l'acquittement du panneau
- QString configure_page (uint8_t _mode)
- QString configure_taille_police (uint8_t _mode)
- QString configure_temps (int temps)
- QString configure_couleur (uint8_t _mode)
- QString configure_position (uint8_t _mode, QString message)

Attributs privés

- QextSerialPort * port
  relation vers la classe QextSerialPort

9.12.1 Documentation des constructeurs et destructeur

9.12.1.1 PortPanneau : :PortPanneau (QObject * *parent* = 0) [explicit]

Paramètres

parent	Adresse de l'objet parent Qt (sinon 0)

Références lireTrame(), port, et PORT_PANNEAU.

```
                                        : QObject(parent)
{
    port = new QextSerialPort(QLatin1String(PORT_PANNEAU),
      QextSerialPort::EventDriven, this);
    port->setBaudRate(BAUD9600);
    port->setDataBits(DATA_8);
    port->setStopBits(STOP_1);
    port->open(QIODevice::ReadWrite);
    if(port->isOpen())
    {
        connect(port, SIGNAL(readyRead()), this, SLOT(lireTrame()));
    }
    else
    {
        qDebug() << Q_FUNC_INFO << "Erreur : port" << QLatin1String(PORT_PANNEAU
      ) << "non ouvert !";
    }
}
```

9.12.1.2 PortPanneau : :~PortPanneau ( )

Références port.

```
{
    if(port->isOpen())
    {
        port->close();
    }
}
```

9.12.2 Documentation des fonctions membres

9.12.2.1 void PortPanneau : :afficherMessage (QString *message*) [slot]

Paramètres

message	

Références creerTrame(), et envoyerTrame().

Référencé par WISMASIHM : :demarrerAffichagePanneau(), et WISMASIHM : :WISMASIHM().

```
{
    QStringList messages = message.split(";");
    for(int i = 0; i < messages.count(); i++)
    {
        QString trame = creerTrame(messages, i);
        envoyerTrame(trame);
    }
}
```

9.12.2.2 QString PortPanneau : :calculerChecksum (QString *trame*) [private]

Paramètres

trame	QString la trame

Renvoie

QString le checksum calculé

Référencé par changerPage(), et creerTrame().

```
{
    uint8_t checksum = 0;

    for(int i=0;i<trame.length();i++)
        checksum ^= trame.at(i).toAscii(); // ^= ou exclusif de chaque
       caractere de la trame

    QString Checksum = QString("%1").arg(checksum, 2, 16, QChar('0')).toUpper()
      ;

    #ifdef DEMO
        qDebug("checksum :\t0x%02X", checksum);
        qDebug() << Q_FUNC_INFO << Checksum;
    #endif

    return Checksum;
}
```

9.12.2.3 void PortPanneau : :changerPage (int *nb_page*) [slot]

Paramètres

nb_page	

Références calculerChecksum(), envoyerTrame(), lireAcquittement(), PANNEAU_PAGE_A, PANNEAU_PAGE_B, PANNEAU_PAGE_C, PANNEAU_PAGE_D, PANNEAU_PAGE_E, PANNEAU_PAGE_F, PANNEAU_PAGE_G, PANNEAU_PAGE_H, et PANNEAU_PAGE_I.

Référencé par WISMASIHM : :afficherInformationsMeteoStation().

```
{
    char page = 0;

    switch(nb_page)
    {
        case 0:
            page = PANNEAU_PAGE_A;
            break;
        case 1:
            page = PANNEAU_PAGE_B;
            break;
        case 2:
            page = PANNEAU_PAGE_C;
            break;
        case 3:
            page = PANNEAU_PAGE_D;
            break;
        case 4:
            page = PANNEAU_PAGE_E;
            break;
        case 5:
            page = PANNEAU_PAGE_F;
            break;
        case 6:
            page = PANNEAU_PAGE_G;
            break;
        case 7:
            page = PANNEAU_PAGE_H;
            break;
        case 8:
            page = PANNEAU_PAGE_I;
            break;
    }

    QString trame;
    //trame = "<SC>";
    trame = QString("<RP%1>").arg(page);
    trame += calculerChecksum(trame);
    trame += "<E>";
    trame = "<ID01>" + trame;
    this->envoyerTrame(trame);
    #ifdef DEMO
        qDebug() << Q_FUNC_INFO << trame << lireAcquittement(trame);
    #endif
}
```

9.12.2.4 QString PortPanneau : :configure_couleur (uint8_t *_mode*) [private]

Référencé par creerTrame().

```
{
    QString mode;
    if(_mode >= 'A' && _mode <= 'S')
        mode = QString("<C%1>").arg(QChar(_mode));
    return mode;
 //
     //sprintf(protocole, "%s<C%c>", protocole, mode);
}
```

9.12.2.5 QString PortPanneau : :configure_page (uint8_t *_mode*) [private]

Référencé par creerTrame().

```
{
    QString mode;
    if(_mode >='A' && _mode <='I')
        mode = QString("<P%1>").arg(QChar(_mode));
```

```
    return mode;
}
```

9.12.2.6 QString PortPanneau : :configure_position (uint8_t *_mode,* QString *message*) [private]

9.12.2.7 QString PortPanneau : :configure_taille_police (uint8_t *_mode*) [private]

Référencé par creerTrame().

```
{
    QString mode;
    if(_mode >= 'A' && _mode <= 'E')
        mode = QString("<A%1>").arg(QChar(_mode));
    return mode;
}
```

9.12.2.8 QString PortPanneau : :configure_temps (int *temps*) [private]

Référencé par creerTrame().

```
{
    QString mode;
    if(temps >= 0 && temps <= 25)
        mode = QString("<W%1>").arg(QString('A' + temps));
    return mode;
}
```

9.12.2.9 QString PortPanneau : :creerTrame (QStringList *messages,* int *nb_page*) [private]

Paramètres

messages	
nb_page	

Renvoie

QString

Références calculerChecksum(), configure_couleur(), configure_page(), configure_taille_police(), configure_temps(), PANNEAU_PAGE_A, PANNEAU_PAGE_B, PANNEAU_PAGE_C, PANNEAU_PAGE_D, PANNEAU_PAGE_E, PANNEAU_PAGE_F, PANNEAU_PAGE_G, PANNEAU_PAGE_H, PANNEAU_PAGE_I, et PANNEAU_SIZE_5X7.

Référencé par afficherMessage().

```
{
    //Exemple de trame : "<ID01><L1><PA><FE><MA><WC><FE>message1F<E>"

    QString protocole = "<L1>";//QString("<L%1>").arg(nb_page+1);
    QString options   = "<FE><MA>";
    QString trame;
```

```
    //Conversion des caractères spéciaux
    if(messages[nb_page].contains("-"))
        messages[nb_page].replace("-", "<U0B> ");

    if(messages[nb_page].contains("°"))
        messages[nb_page].replace("°", "<U3A>");

    if(messages[nb_page].contains("é"))
        messages[nb_page].replace("é", "<U69>");

    if(messages[nb_page].contains("€"))
        messages[nb_page].replace("€", "<U00>");

    if(messages[nb_page].contains("à"))
        messages[nb_page].replace("à", "<U60>");

    char page = 0;

    switch(nb_page)
    {
        case 0:
            page = PANNEAU_PAGE_A;
            break;
        case 1:
            page = PANNEAU_PAGE_B;
            break;
        case 2:
            page = PANNEAU_PAGE_C;
            break;
        case 3:
            page = PANNEAU_PAGE_D;
            break;
        case 4:
            page = PANNEAU_PAGE_E;
            break;
        case 5:
            page = PANNEAU_PAGE_F;
            break;
        case 6:
            page = PANNEAU_PAGE_G;
            break;
        case 7:
            page = PANNEAU_PAGE_H;
            break;
        case 8:
            page = PANNEAU_PAGE_I;
            break;
    }


    trame  = protocole;

    trame += configure_page(page);

    trame += options;

    trame += configure_temps(10);

    trame += "<FE>";
    trame += configure_taille_police(PANNEAU_SIZE_5X7);
    trame += configure_couleur('A');
    //trame += configure_position(PANNEAU_POSITION_CENTER, messages[nb_page]);
    trame += messages[nb_page];
    trame += calculerChecksum(trame);
    trame += "<E>";
    trame = "<ID01>" + trame;

    #ifdef DEMO
        qDebug() << Q_FUNC_INFO << trame;
    #endif


    return trame;
}
```

9.12.2.10 void PortPanneau : :envoyerTrame (QString *trame*) [private]

Paramètres

trame	QString la trame

Références port.

Référencé par afficherMessage(), et changerPage().

```
{
    port->write(trame.toLocal8Bit().constData());
}
```

9.12.2.11 bool PortPanneau : :lireAcquittement (QString *trame*) [private]

Paramètres

trame	QString la trame

Renvoie

bool true si acquittement sinon false

Références PANNEAU_ACK, et PANNEAU_NACK.

Référencé par changerPage().

```
{
    if(trame.contains(PANNEAU_NACK))
        return false;
    else if(trame.contains(PANNEAU_ACK))
        return true;
    else
        return false;
}
```

9.12.2.12 void PortPanneau : :lireTrame ( ) [slot]

Références port.

Référencé par PortPanneau().

```
{
    QByteArray donneesRecues;

    if(port->isOpen())
    {
        while(port->bytesAvailable())
        {
            donneesRecues += port->readAll();
            usleep(10000);
        }

        QString trame(donneesRecues);


    }
}
```

9.12.3 Documentation des données membres

9.12.3.1 QextSerialPort* PortPanneau : :port [private]

Référencé par envoyerTrame(), lireTrame(), PortPanneau(), et ~PortPanneau().

La documentation de cette classe a été générée à partir des fichiers suivants :
– portpanneau.h
– portpanneau.cpp

9.13 Référence de la classe PortXBee

Communique avec les stations météo des sites via un port série virtuel.

`#include <portxbee.h>`

Connecteurs publics

– void recevoirTrame ()
Receptionne les trames en provenance des stations météos.

Signaux

– void nouvelleTrame (QString trame)

Fonctions membres publiques

– PortXBee (QObject *parent=0)
Constructeur de la classe PortXBee.
– ~PortXBee ()
Destructeur.

Attributs privés

– QextSerialPort * port
relation vers la classe QextSerialPort

9.13.1 Documentation des constructeurs et destructeur

9.13.1.1 PortXBee : :PortXBee (QObject * *parent* = 0) [explicit]

Constructeur.

Paramètres

parent	Adresse de l'objet parent Qt (sinon 0)
parent	QObject

Références port, PORT_XBEE, et recevoirTrame().

```
                                    : QObject(parent)
{
    port = new QextSerialPort(QLatin1String(PORT_XBEE),
      QextSerialPort::EventDriven, this);
    port->setBaudRate(BAUD9600);
    port->setDataBits(DATA_8);
    port->setStopBits(STOP_1);
    port->open(QIODevice::ReadWrite);
    if(port->isOpen())
    {
        connect(port, SIGNAL(readyRead()), this, SLOT(recevoirTrame()));
    }
    else
    {
        qDebug() << Q_FUNC_INFO << "Erreur : port" << QLatin1String(PORT_XBEE)
      << "non ouvert !";
    }
}
```

9.13.1.2 PortXBee : :~PortXBee ( )

Références port.

```
{
    if(port->isOpen())
    {
        port->close();
    }
}
```

9.13.2 Documentation des fonctions membres

9.13.2.1 void PortXBee : :nouvelleTrame (QString *trame*) [signal]

Référencé par recevoirTrame().

9.13.2.2 PortXBee : :recevoirTrame ( ) [slot]

Références nouvelleTrame(), et port.

Référencé par PortXBee().

```
{
    QByteArray donneesRecues;

    if(port->isOpen())
    {
        while(port->bytesAvailable())
        {
            donneesRecues += port->readAll();
            usleep(10000);
        }

        QString trame(donneesRecues);

        //qDebug() << Q_FUNC_INFO << trame;

        emit nouvelleTrame(trame);
    }
    else
    {
        // Simulation
        //QString trame =
       "$ID.101,TS.1,DV.SE,VV.11,TA.20,TN.-15,H.30,HY.80,B.1010;";
```

```
        //emit nouvelleTrame(trame);
    }
}
```

9.13.3 Documentation des données membres

9.13.3.1 QextSerialPort* PortXBee : :port [private]

Référencé par PortXBee(), recevoirTrame(), et ~PortXBee().

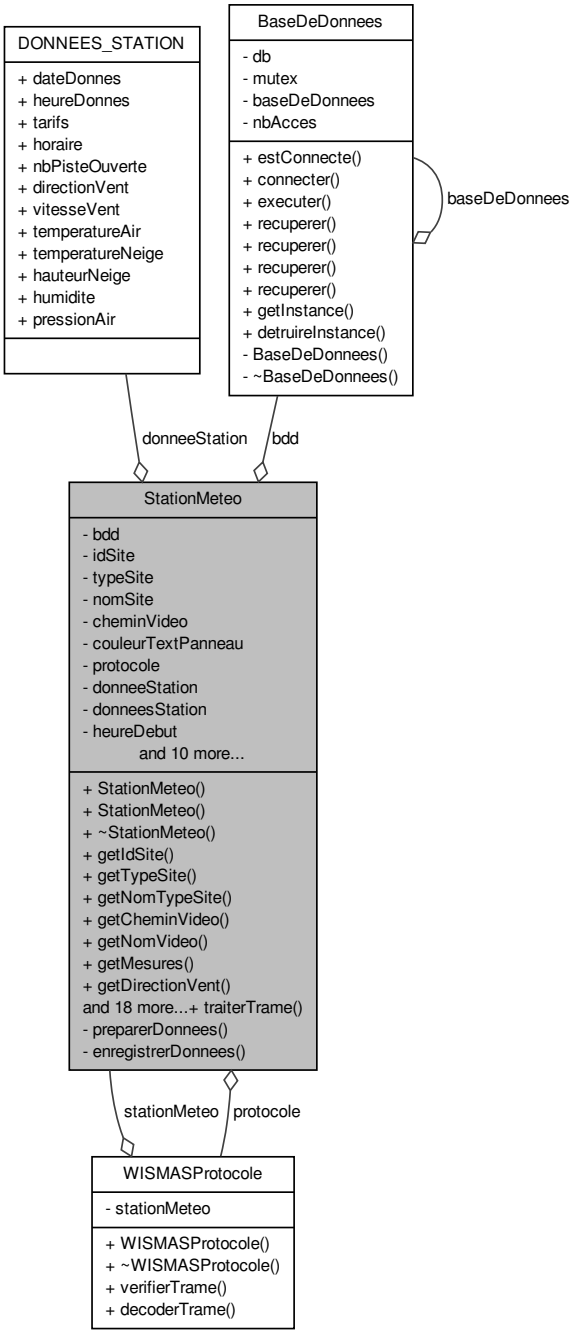
La documentation de cette classe a été générée à partir des fichiers suivants :
– portxbee.h
– portxbee.cpp

9.14 Référence de la classe StationMeteo

Gére les données météos d'une station d'un site.

`#include <stationmeteo.h>`

Graphe de collaboration de StationMeteo :



Connecteurs publics

- void traiterTrame (QString trame)

Fonctions membres publiques

- StationMeteo (QObject *parent=0)
  Constructeur.
- StationMeteo (int idSite, int typeSite, QString nomSite, QString cheminVideo, QString couleurTextPanneau, QObject *parent=0)
  StationMeteo : :StationMeteo.
- ~StationMeteo ()
  StationMeteo : :~StationMeteo.
- int getIdSite () const
  StationMeteo : :getIdSite.
- int getTypeSite () const
  StationMeteo : :getTypeSite.
- QString getNomTypeSite () const
  StationMeteo : :getNomTypeSite.
- QString getCheminVideo () const
  StationMeteo : :getCheminVideo.
- QString getNomVideo () const
- QString getMesures (int typeMesure)
  StationMeteo : :getMesures.
- QString getDirectionVent () const
  StationMeteo : :getDirectionVent.
- QString getVitesseVent () const
  StationMeteo : :getVitesseVent.
- QString getTemperatureAir ()
  StationMeteo : :getTemperatureAir.
- QString getTemperatureNeige ()
  StationMeteo : :getTemperatureNeige.
- QString getHauteurNeige ()
  StationMeteo : :getHauteurNeige.
- QString getHumidite () const
  StationMeteo : :getHumidite.
- QString getPressionAir () const
  StationMeteo : :getPressionAir.
- QString getConseilsFartage (int marque) const
  StationMeteo : :getConseilsFartage.
- QStringList getInformationComplementaire (int idSite)
- void setIdSite (int)
  StationMeteo : :setId.
- void setTypeSite (int)
  StationMeteo : :setTypeSite.
- void setDonneesTrame (DONNEES_STATION)
- void setNomVideo (QString)
  StationMeteo : :setNomVideo.
- void setDirectionVent (QString)
  StationMeteo : :setDirectionVent.
- void setVitesseVent (QString)
  StationMeteo : :setVitesseVent.
- void setTemperatureAir (QString)
  StationMeteo : :setTemperatureAir.
- void setTemperatureNeige (QString)
  StationMeteo : :setTemperatureNeige.
- void setHauteurNeige (QString)
  StationMeteo : :setHauteurNeige.

- void setHumidite (QString)
  StationMeteo : :setHumidite.
- void setPressionAir (QString)
  StationMeteo : :setPressionAir.

Fonctions membres privées

- bool preparerDonnees (DONNEES_STATION &donneesEnregistrement)
  preparerDonnees() prepare les données pour l'enregistrement
- void enregistrerDonnees ()
  enregistrerDonnees() enregistre les données dans la base de données.

Attributs privés

- BaseDeDonnees * bdd
  Association vers la classe BaseDeDonnees.
- int idSite
- int typeSite
- QString nomSite
- QString cheminVideo
- QString couleurTextPanneau
- WISMASProtocole * protocole
  Association vers la classe WISMASProtocole.
- DONNEES_STATION donneeStation
- QVector< DONNEES_STATION > donneesStation
- QTime heureDebut
- QTime heureFin
- QTimer * timer_baseDeDonnees
- QString directionVent
- QString vitesseVent
- QString temperatureAir
- QString temperatureNeige
- QString hauteurNeige
- QString humidite
- QString pressionAir
- QString nomVideo
- int heureCourante

9.14.1 Documentation des constructeurs et destructeur

9.14.1.1 StationMeteo : :StationMeteo (QObject * *parent* = 0)

Constructeur par défaut

Paramètres

parent	QObject L'adresse d'un objet Qt parent

Références bdd, BaseDeDonnees : :connecter(), BaseDeDonnees : :getInstance(), heureCourante, heureDebut, heureFin, et protocole.

```
                                        : QObject(parent), idSite(0),
    typeSite(0), directionVent("--"), vitesseVent("--"), temperatureAir("--"),
    temperatureNeige("--"), hauteurNeige("--"), humidite("--"), pressionAir("--
      "), nomVideo("")
{
    bdd = BaseDeDonnees::getInstance();
```

```
    bdd->connecter();

    this->protocole = new WISMASProtocole(this);

    heureDebut = QTime::currentTime();
    //heureFin = QTime::currentTime().addSecs(3600);
    heureFin = QTime::currentTime().addSecs(60);

    heureCourante = QDateTime::currentDateTime().toString("HH").toInt();
}
```

9.14.1.2 StationMeteo : :StationMeteo (int *idSite,* int *typeSite,* QString *nomSite,* QString *cheminVideo,* QString *couleurTextPanneau,* QObject ∗ *parent* = 0)

Constructeur par défaut

Références bdd, BaseDeDonnees : :connecter(), BaseDeDonnees : :executer(), BaseDeDonnees : :getInstance(), heureCourante, heureDebut, heureFin, protocole, et BaseDeDonnees : :recuperer().

```
                                                                : QObject(parent), idSite
      (idSite),
    typeSite(typeSite), nomSite(nomSite), cheminVideo(cheminVideo),
      couleurTextPanneau(couleurTextPanneau), directionVent("--"), vitesseVent("--"),
       temperatureAir("--"),
    temperatureNeige("--"), hauteurNeige("--"), humidite("--"), pressionAir("--
      ")
{
    bdd = BaseDeDonnees::getInstance();
    bdd->connecter();

    QStringList infosSite;
    QString requete;
    bool retour;

    // Récupère les information du site
    requete = "SELECT * FROM sites WHERE idSite = '" + QString::number(idSite)
      + "'";
    retour = bdd->recuperer(requete, infosSite);
    if(retour)
    {
        #ifdef DEBUG

            qDebug() << QString::fromUtf8("Site : ") << infosSite.at(0) <<
      infosSite.at(1) << infosSite.at(2) << infosSite.at(3);
        #endif

    }
    else
    {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << requete;
        #endif

    }

    // La station est active
    requete = "UPDATE sites SET etat = '1' WHERE idSite = '" + QString::number(
      idSite) + "'";
    retour = bdd->executer(requete);
    if(!retour)
    {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << requete;
        #endif
```

```
    }

    this->protocole = new WISMASProtocole(this);

    heureDebut = QTime::currentTime();
    //heureFin = QTime::currentTime().addSecs(3600);
    heureFin = QTime::currentTime().addSecs(60);

    heureCourante = QDateTime::currentDateTime().toString("HH").toInt();
}
```

9.14.1.3 StationMeteo : :~StationMeteo ( )

Références bdd, BaseDeDonnees : :detruireInstance(), BaseDeDonnees : :executer(), et idSite.

```
{
    // La station est inactive
    QString requete = "UPDATE sites SET etat = '0' WHERE idSite = '" +
      QString::number(idSite) + "'";
    bool retour = bdd->executer(requete);
    if(!retour)
    {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << requete;
        #endif

    }

    BaseDeDonnees::detruireInstance();
}
```

9.14.2 Documentation des fonctions membres

9.14.2.1 void StationMeteo : :enregistrerDonnees ( ) [private]

Renvoie

void

Références bdd, DONNEES_STATION : :dateDonnes, DONNEES_STATION- : :directionVent, BaseDeDonnees : :estConnecte(), BaseDeDonnees : :executer(), getIdSite(), DONNEES_STATION : :hauteurNeige, DONNEES_STATION : :heure-Donnes, DONNEES_STATION : :humidite, preparerDonnees(), DONNEES_STAT-ION : :pressionAir, DONNEES_STATION : :temperatureAir, DONNEES_STATION- : :temperatureNeige, et DONNEES_STATION : :vitesseVent.

Référencé par traiterTrame().

```
{
    DONNEES_STATION donneesEnregistrement;

    if(!preparerDonnees(donneesEnregistrement))
        return;

    if(bdd->estConnecte())
    {
        bool retour;
        QString requete;

        QDate dateMesure = QDate::fromString(donneesEnregistrement.dateDonnes,
```

```
        "dd/MM/yyyy");

          requete = "INSERT INTO
         mesures(idSite,dateMesure,heure,temperatureAir,temperatureNeige,hauteurNeige,humidite,pressionAir,vitesseV
         VALUES('" + QString::number(this->getIdSite()) + "','" + dateMesure.toString("
        yyyy-MM-dd") + "','" + donneesEnregistrement.heureDonnes + "','" +
        donneesEnregistrement.temperatureAir + "','" + donneesEnregistrement.temperatureNeige + "','" +
        donneesEnregistrement.hauteurNeige + "','" + donneesEnregistrement.humidite + "','
        " + donneesEnregistrement.pressionAir + "','" + donneesEnregistrement.vitesseVent
         + "','" + donneesEnregistrement.directionVent + "')";

          retour = bdd->executer(requete);
          qDebug() << Q_FUNC_INFO << requete << retour;
          if(!retour)
          {
              qDebug() << Q_FUNC_INFO << "Erreur " << requete;
          }
    }
}
```

9.14.2.2 QString StationMeteo : :getCheminVideo ( ) const

Renvoie

QString cheminVideo.

Références cheminVideo.

```
{
    return cheminVideo;
}
```

9.14.2.3 QString StationMeteo : :getConseilsFartage (int *marque*) const

Paramètres

marque	int marque de farte

Renvoie

QString la couleur du farte

Références donneesStation, et temperatureNeige.

```
{
    if(donneesStation.count() == 0)
        return "--";

    switch(marque)
    {
        case 1:
            if(this->temperatureNeige.toInt() >= 0 && this->temperatureNeige.
      toInt() <= 10)    { return "jaune"; }
            else if(abs(this->temperatureNeige.toInt()) >= 4 && abs(this->
      temperatureNeige.toInt()) <= abs(-4))    { return "rouge"; }
            else if(abs(this->temperatureNeige.toInt()) >= abs(-2) && abs(
      this->temperatureNeige.toInt()) <= abs(-8))  { return "violet";}
            else if(abs(this->temperatureNeige.toInt()) >= abs(-6) && abs(
      this->temperatureNeige.toInt()) <= abs(-12)) { return "bleu";  }
            else if(abs(this->temperatureNeige.toInt()) >= abs(-10) && abs(
      this->temperatureNeige.toInt()) <= abs(-32)){ return "vert";  }
            else
            {
                #ifdef DEBUG
```

```
            qDebug() << Q_FUNC_INFO << "Temperature de neige non prise
 en charge pour la marque SWIX";
        #endif

            return "--";
        }
        break;

    case 2:
        if(this->temperatureNeige.toInt() >= 0 && this->temperatureNeige.
toInt() <= 6)    { return "jaune"; }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-4) && abs(
this->temperatureNeige.toInt()) <= abs(-12)) { return "rouge"; }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-10) && abs(
this->temperatureNeige.toInt()) <= abs(-30)){ return "bleu";  }
        else
        {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << "Temperature de neige non prise
 en charge pour la marque TOKO";
        #endif

            return "--";
        }
        break;
    case 3:
        if(this->temperatureNeige.toInt() >= 14 && abs(this->
temperatureNeige.toInt()) <= abs(-2))  { return "jaune"; }
        else if(this->temperatureNeige.toInt() >= 0 && abs(this->
temperatureNeige.toInt()) <= abs(-5))   { return "rouge"; }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-2) && abs(
this->temperatureNeige.toInt()) <= abs(-10)) { return "violet";}
        else if(abs(this->temperatureNeige.toInt()) >= abs(-7) && abs(
this->temperatureNeige.toInt()) <= abs(-15)) { return "bleu";  }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-10) && abs(
this->temperatureNeige.toInt()) <= abs(-25)){ return "vert";  }
        else
        {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << "Temperature de neige non prise
 en charge pour la marque VOLA";
        #endif

            return "--";
        }
        break;
    case 4:
        if(this->temperatureNeige.toInt() >= 0 && abs(this->temperatureNeige
.toInt()) <= abs(-4))   { return "jaune"; }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-4) && abs(
this->temperatureNeige.toInt()) <= abs(-8))  { return "rouge"; }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-6) && abs(
this->temperatureNeige.toInt()) <= abs(-12)) { return "violet";}
        else if(abs(this->temperatureNeige.toInt()) >= abs(-10) && abs(
this->temperatureNeige.toInt()) <= abs(-18)){ return "bleu";  }
        else if(abs(this->temperatureNeige.toInt()) >= abs(-10) && abs(
                this->temperatureNeige.toInt()) <= abs(-30)){ return "
vert";  }
        else
        {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << "Temperature de neige non prise
 en charge pour la marque BRIKO / MAPLUS";
        #endif

            return "--";
        }
        break;
    case 5:
        if(this->temperatureNeige.toInt() >= 10 && this->temperatureNeige.
```

```
toInt() <= 0)  { return "jaune"; }
      else if(this->temperatureNeige.toInt() >= 2 && abs(this->
temperatureNeige.toInt()) <= abs(-8))  { return "rouge"; }
      else if(abs(this->temperatureNeige.toInt()) >= abs(-5) && abs(
this->temperatureNeige.toInt()) <= abs(-15)){ return "bleu";  }
      else
      {
          #ifdef DEBUG

              qDebug() << Q_FUNC_INFO << "Temperature de neige non prise
 en charge pour la marque DRAGON SKI";
          #endif

          return "--";
      }
      break;
   case 6:
      if(this->temperatureNeige.toInt() >= 10 && this->temperatureNeige.
toInt() <= 0)  { return "jaune"; }
      else if(abs(this->temperatureNeige.toInt()) >= abs(-3) && abs(
this->temperatureNeige.toInt()) <= abs(-7)) { return "rouge"; }
      else if(abs(this->temperatureNeige.toInt()) >= abs(-5) && abs(
this->temperatureNeige.toInt()) <= abs(-15)){ return "bleu";  }
      else
      {
          #ifdef DEBUG

              qDebug() << Q_FUNC_INFO << "Temperature de neige non prise
 en charge pour la marque XC SOLUTION";
          #endif

          return "--";
      }
      break;

    default:
      #ifdef DEBUG

          qDebug() << Q_FUNC_INFO << "Temperature de neige non prise en
 charge";
      #endif

      return "--";
      break;
  }
}
```

9.14.2.4 QString StationMeteo : :getDirectionVent ( ) const

Renvoie

QString

Références directionVent.

Référencé par getMesures().

```
{
    if(this->directionVent == "500")
    {
        return "--";
    }
    else
    {
        return this->directionVent;
    }
}
```

9.14.2.5 QString StationMeteo : :getHauteurNeige ( )

Renvoie

QString

Références hauteurNeige.

Référencé par getMesures().

```
{
    if(this->hauteurNeige == "500")
    {
        return "-- cm";
    }
    else
    {
        return this->hauteurNeige + "cm";
    }
}
```

9.14.2.6 QString StationMeteo : :getHumidite ( ) const

Renvoie

QString

Références humidite.

Référencé par getMesures().

```
{
    if(this->humidite == "500")
    {
        return "-- %";
    }
    else
    {
        return this->humidite +" %";
    }
}
```

9.14.2.7 int StationMeteo : :getIdSite ( ) const

Renvoie

int id l'identifiant du site sur lequel la station météo est installée.

Références idSite.

Référencé par WISMASProtocole : :decoderTrame(), et enregistrerDonnees().

```
{
    return idSite;
}
```

9.14.2.8 QStringList StationMeteo : :getInformationComplementaire (int *idSite*)

Références bdd, BaseDeDonnees : :estConnecte(), et BaseDeDonnees : :recuperer().

Référencé par WISMASIHM : :demarrerAffichagePanneau(), et WISMASIHM : :WISMASIHM().

```
{
    if(bdd->estConnecte())
    {
        bool retour;
        QString requete;
        QStringList donnees;

        requete = "SELECT tarifs, horaire, nbPisteOuverte FROM informations
       WHERE idSite = " + QString::number(idSite);

        retour = bdd->recuperer(requete, donnees);

        if(retour)
        {
            return donnees;
        }
    }
}
```

9.14.2.9 QString StationMeteo : :getMesures (int *typeMesure*)

Renvoie

QString

Références DIRECTION_VENT, getDirectionVent(), getHauteurNeige(), getHumidite(), getPressionAir(), getTemperatureAir(), getTemperatureNeige(), getVitesseVent(), HAUTEUR_NEIGE, HUMIDITE, PRESSION_AIR, TEMPERATURE_AIR, TEMPERATURE_NEIGE, et VITESSE_VENT.

```
{
    switch(typeMesure)
    {
        case DIRECTION_VENT:
            return getDirectionVent();
            break;

        case VITESSE_VENT:
            return getVitesseVent();
            break;

        case TEMPERATURE_AIR:
            return getTemperatureAir();
            break;

        case TEMPERATURE_NEIGE:
            return getTemperatureNeige();
            break;

        case HAUTEUR_NEIGE:
            return getHauteurNeige();
            break;

        case HUMIDITE:
            return getHumidite();
            break;

        case PRESSION_AIR:
            return getPressionAir();
            break;
    }
}
```

9.14.2.10 QString StationMeteo : :getNomTypeSite ( ) const

Renvoie

QString nom le nom du type de site.

Références SKI_ALPIN, SKI_DE_FOND, SNOW_SKITE, et typeSite.

```
{
    switch(typeSite)
    {
        case SKI_ALPIN:
            return "SKI ALPIN";
            break;
        case SKI_DE_FOND:
            return "SKI DE FOND";
            break;
        case SNOW_SKITE:
            return "SNOWKITE";
            break;
        default:
            qDebug() << Q_FUNC_INFO << " [INFO] Type de site inconnu";
            return "";
            break;
    }
}
```

9.14.2.11 QString StationMeteo : :getNomVideo ( ) const

Références nomVideo.

```
{
    return this->nomVideo;
}
```

9.14.2.12 QString StationMeteo : :getPressionAir ( ) const

Renvoie

QString

Références pressionAir.

Référencé par getMesures().

```
{
    if(this->pressionAir == "500")
    {
        return "-- hPa";
    }
    else
    {
        return this->pressionAir + " hPa";
    }

}
```

9.14.2.13 QString StationMeteo : :getTemperatureAir ( )

Renvoie

QString

Références temperatureAir.

Référencé par getMesures().

```
{
    if(this->temperatureAir == "500")
    {
        return "-- °C";
    }
    else
    {
        return this->temperatureAir + "°C";
    }
}
```

9.14.2.14 QString StationMeteo : :getTemperatureNeige ( )

Renvoie

QString

Références temperatureNeige.

Référencé par getMesures().

```
{
    if(this->temperatureNeige == "500")
    {
        return "-- °C";
    }
    else
    {
        return this->temperatureNeige + "°C";
    }
}
```

9.14.2.15 int StationMeteo : :getTypeSite ( ) const

Renvoie

int id l'identifiant du site.

Références typeSite.

Référencé par WISMASProtocole : :decoderTrame().

```
{
    return typeSite;
}
```

9.14.2.16 QString StationMeteo : :getVitesseVent ( ) const

Renvoie

QString

Références vitesseVent.

Référencé par getMesures().

```
{
    if(this->vitesseVent == "500")
    {
        return "-- km/h";
    }
    else
    {
        return this->vitesseVent + "km/h";
    }
}
```

9.14.2.17 bool StationMeteo : :preparerDonnees (DONNEES_STATION & *donneesEnregistrement*) [private]

Renvoie

bool

Références DONNEES_STATION : :dateDonnes, directionVent, DONNEES_STATION : :directionVent, donneesStation, donneeStation, hauteurNeige, DONNEES_STATION : :hauteurNeige, heureCourante, heureDebut, DONNEES_STATION : :heureDonnes, heureFin, humidite, DONNEES_STATION : :humidite, pressionAir, DONNEES_STATION : :pressionAir, temperatureAir, DONNEES_STATION : :temperatureAir, temperatureNeige, DONNEES_STATION : :temperatureNeige, vitesseVent, et DONNEES_STATION : :vitesseVent.

Référencé par enregistrerDonnees().

```
{

    QTime heureCourante = QTime::currentTime();

    // une heure d'acquisition ?
    if(heureCourante > heureFin)
    {
        if(donneesStation.count() == 0)
            return false;

        // calcule la moyenne horaire des données stockées
        double vitesseVent      = 0.;
        double temperatureAir   = 0.;
        double temperatureNeige = 0.;

        int hauteurNeige = 0;
        int humidite     = 0;
        int pressionAir  = 0;

        QString directionVent = "";


        for(int i=0; i < donneesStation.count(); i++)
        {
            vitesseVent      += donneesStation.at(i).vitesseVent.toDouble();
            temperatureAir   += donneesStation.at(i).temperatureAir.toDouble();
            temperatureNeige += donneesStation.at(i).temperatureNeige.toDouble(
      );
```

```
            hauteurNeige += donneesStation.at(i).hauteurNeige.toInt();
            humidite     += donneesStation.at(i).humidite.toInt();
            pressionAir  += donneesStation.at(i).pressionAir.toInt();
        }

        donneesEnregistrement.dateDonnes = donneeStation.dateDonnes;
        donneesEnregistrement.heureDonnes = QString::number(heureDebut.hour());
       // juste l'heure H
        donneesEnregistrement.vitesseVent = QString("%1").arg(qRound(
      vitesseVent/donneesStation.count()), 0, 'f', 0);
        donneesEnregistrement.temperatureAir = QString("%1").arg(qRound(
      temperatureAir/donneesStation.count()), 0, 'f', 0);
        donneesEnregistrement.temperatureNeige = QString("%1").arg(qRound(
      temperatureNeige/donneesStation.count()), 0, 'f', 0);
        donneesEnregistrement.hauteurNeige = QString("%1").arg(qRound(
      hauteurNeige/donneesStation.count()), 0, 'f', 0);
        donneesEnregistrement.humidite = QString("%1").arg(qRound(humidite/
      donneesStation.count()), 0, 'f', 0);
        donneesEnregistrement.pressionAir = QString("%1").arg(qRound(
      pressionAir/donneesStation.count()), 0, 'f', 0);
        donneesEnregistrement.directionVent = directionVent;

        // initialise pour la prochaine acquisition
        donneesStation.clear();
        heureDebut = QTime::currentTime();
        //heureFin = heureDebut.addSecs(3600);
        heureFin = heureDebut.addSecs(60);

        return true;
    }
    else
        return false;
}
```

9.14.2.18 void StationMeteo : :setDirectionVent (QString *directionVent*)

Paramètres

direction-Vent	int

Références directionVent.

Référencé par setDonneesTrame().

```
{
    this->directionVent = directionVent;
}
```

9.14.2.19 void StationMeteo : :setDonneesTrame (DONNEES_STATION *donnees*)

Références DONNEES_STATION : :directionVent, donneesStation, donneeStation, -DONNEES_STATION : :hauteurNeige, DONNEES_STATION : :humidite, DONNEES_STATION : :pressionAir, setDirectionVent(), setHauteurNeige(), setHumidite(), setPressionAir(), setTemperatureAir(), setTemperatureNeige(), setVitesseVent(), DONNEES_STATION : :temperatureAir, DONNEES_STATION : :temperatureNeige, et DONNEES_STATION : :vitesseVent.

Référencé par WISMASProtocole : :decoderTrame().

```
{
    this->donneeStation = donnees;
```

```
    this->setDirectionVent(donneeStation.directionVent);
    this->setVitesseVent(donneeStation.vitesseVent);
    this->setTemperatureAir(donneeStation.temperatureAir);
    this->setTemperatureNeige(donneeStation.temperatureNeige);
    this->setHauteurNeige(donneeStation.hauteurNeige);
    this->setHumidite(donneeStation.humidite);
    this->setPressionAir(donneeStation.pressionAir);

    this->donneesStation.push_back(this->donneeStation);
}
```

9.14.2.20 void StationMeteo : :setHauteurNeige (QString *hauteur_neige*)

Paramètres

hauteur_neige	int

Références hauteurNeige.

Référencé par setDonneesTrame().

```
{
    this->hauteurNeige = hauteur_neige;
}
```

9.14.2.21 void StationMeteo : :setHumidite (QString *humidite*)

Paramètres

humidite	int

Références humidite.

Référencé par setDonneesTrame().

```
{
    this->humidite = humidite;
}
```

9.14.2.22 void StationMeteo : :setIdSite (int *id*)

Paramètres

id	int

Références idSite.

```
{
    this->idSite = id;
}
```

9.14.2.23 void StationMeteo : :setNomVideo (QString *nomVideo*)

Paramètres

nomVideo	QString

Références nomVideo.

```
{
    this->nomVideo = nomVideo;
}
```

9.14.2.24 void StationMeteo : :setPressionAir (QString *pression*)

Paramètres

pression	int

Références pressionAir.

Référencé par setDonneesTrame().

```
{
    this->pressionAir = pression;
}
```

9.14.2.25 void StationMeteo : :setTemperatureAir (QString *temperatureAir*)

Paramètres

temperature-Air	int

Renvoie

void

Références temperatureAir.

Référencé par setDonneesTrame().

```
{
    this->temperatureAir = temperatureAir;
}
```

9.14.2.26 void StationMeteo : :setTemperatureNeige (QString *temperatureNeige*)

Paramètres

temperature-Neige	int

Références temperatureNeige.

Référencé par setDonneesTrame().

```
{
```

```
    this->temperatureNeige = temperatureNeige;
}
```

9.14.2.27 void StationMeteo : :setTypeSite (int *typeSite*)

Paramètres

typeSite	int

Références typeSite.

```
{
    this->typeSite = typeSite;
}
```

9.14.2.28 void StationMeteo : :setVitesseVent (QString *vitesseVent*)

Paramètres

vitesseVent	QString vitesse du vent

Références vitesseVent.

Référencé par setDonneesTrame().

```
{
    this->vitesseVent = vitesseVent;
}
```

9.14.2.29 void StationMeteo : :traiterTrame (QString *trame*) [slot]

Paramètres

trame	QString trame reçue

Références WISMASProtocole : :decoderTrame(), enregistrerDonnees(), heureCourante, et protocole.

```
{
    QStringList trames = trame.split('$');

    for(int i = 1; i < trames.count(); i++)
    {
        if(this->protocole->decoderTrame(trames.at(i)))
        {
            int heure = QDateTime::currentDateTime().toString("HH").toInt();
            if(this->heureCourante + 1 == heure)
            {
               enregistrerDonnees();
               this->heureCourante = QDateTime::currentDateTime().toString("HH"
      ).toInt();
            }
        }
    }
}
```

9.14.3 Documentation des données membres

9.14.3.1 BaseDeDonnees* StationMeteo : :bdd [private]

Référencé par enregistrerDonnees(), getInformationComplementaire(), StationMeteo(), et ~StationMeteo().

9.14.3.2 QString StationMeteo : :cheminVideo [private]

Référencé par getCheminVideo().

9.14.3.3 QString StationMeteo : :couleurTextPanneau [private]

9.14.3.4 QString StationMeteo : :directionVent [private]

Référencé par getDirectionVent(), preparerDonnees(), et setDirectionVent().

9.14.3.5 QVector<DONNEES_STATION> StationMeteo : :donneesStation [private]

Référencé par getConseilsFartage(), preparerDonnees(), et setDonneesTrame().

9.14.3.6 DONNEES_STATION StationMeteo : :donneeStation [private]

Référencé par preparerDonnees(), et setDonneesTrame().

9.14.3.7 QString StationMeteo : :hauteurNeige [private]

Référencé par getHauteurNeige(), preparerDonnees(), et setHauteurNeige().

9.14.3.8 int StationMeteo : :heureCourante [private]

Référencé par preparerDonnees(), StationMeteo(), et traiterTrame().

9.14.3.9 QTime StationMeteo : :heureDebut [private]

Référencé par preparerDonnees(), et StationMeteo().

9.14.3.10 QTime StationMeteo : :heureFin [private]

Référencé par preparerDonnees(), et StationMeteo().

9.14.3.11 QString StationMeteo : :humidite [private]

Référencé par getHumidite(), preparerDonnees(), et setHumidite().

9.14.3.12 int StationMeteo : :idSite [private]

Référencé par getIdSite(), setIdSite(), et ~StationMeteo().

9.14.3.13 QString StationMeteo : :nomSite [private]

9.14.3.14 QString StationMeteo : :nomVideo [private]

Référencé par getNomVideo(), et setNomVideo().

9.14.3.15 QString StationMeteo : :pressionAir [private]

Référencé par getPressionAir(), preparerDonnees(), et setPressionAir().

9.14.3.16 WISMASProtocole* StationMeteo : :protocole [private]

Référencé par StationMeteo(), et traiterTrame().

9.14.3.17 QString StationMeteo : :temperatureAir [private]

Référencé par getTemperatureAir(), preparerDonnees(), et setTemperatureAir().

9.14.3.18 QString StationMeteo : :temperatureNeige [private]

Référencé par getConseilsFartage(), getTemperatureNeige(), preparerDonnees(), et setTemperatureNeige().

9.14.3.19 QTimer* StationMeteo : :timer_baseDeDonnees [private]

9.14.3.20 int StationMeteo : :typeSite [private]

Référencé par getNomTypeSite(), getTypeSite(), et setTypeSite().

9.14.3.21 QString StationMeteo : :vitesseVent [private]

Référencé par getVitesseVent(), preparerDonnees(), et setVitesseVent().

La documentation de cette classe a été générée à partir des fichiers suivants :
- stationmeteo.h
- stationmeteo.cpp

9.15 Référence de la classe Video

Traiter les vidéos capturer par le système vidéo.

`#include <Video.h>`

Connecteurs publics

- void arreter ()
  Arrêter la capture d'image vidéo.
- void progresser ()
  Actualiser la barre de progression de l'acquisition vidéo.
- void supprimerEnregistrements () const
  Supprimer les vidéos enregistrées.

Signaux

- void fini () const
- void progression (int chargement) const

Fonctions membres publiques

- Video ()
  Constructeur.
- ∼Video ()
  Destructeur.
- void enregistrerVideo (const int numeroCamera, const QString nomSite, const QString adresseIP, const QString numeroPort, const QString identifiant, const QString motDePasse, const int duree)
  Enregistrer la vidéo capturée par une caméra.

Fonctions membres publiques statiques

- static void arreterEnregistrements ()
  Arrêter tous les enregistrements en cours d'exécution.

Fonctions membres privées

- int getNumeroVideo ()
  Accesseur de nbVideos.
- void creerCheminVideo (int numeroCamera)
  Créer le chemin où enregistrer les vidéos.
- void demarrerTimers (int duree)
  Démarrer les timers.
- void arreterTimers ()
  Arrêter les timers.
- void verifierDossierEnregistrements (QDir &repertoire) const
  Vérifier si les répertoires des enregistrements pour chaque site sont p.
- void supprimerTousLesEnregistrements (QDir &repertoire) const
  Supprimer les vidéos enregistrées.
- void supprimerNbEnregistrements (QDir &repertoire) const
  Supprimer les vidéos enregistrées.

Attributs privés

- QProcess ∗ processus
- QTimer ∗ timerProgression
- QTimer ∗ timerEnregistrement
- bool enregistrement
- int chargement
- int numeroVideo
- int duree
- QString fichierINI
- QString cheminVideo

9.15.1 Description détaillée

Auteur

Pierre GRELET

Version

1.1

Date

15 mars 2018

9.15.2 Documentation des constructeurs et destructeur

9.15.2.1 Video : :Video ( )

Références chargement, enregistrement, fichierINI, numeroVideo, processus, timerEnregistrement, et timerProgression.

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    processus = new QProcess(this); // Crée l'objet processus
    timerProgression = new QTimer(this);
    timerEnregistrement = new QTimer(this);

    fichierINI = QApplication::applicationDirPath() + "/" + "
      configuration-cameras.ini";

    enregistrement = false;
    numeroVideo = 1;
    chargement = 0;
}
```

9.15.2.2 Video : :~Video ( )

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif
}
```

9.15.3 Documentation des fonctions membres

9.15.3.1 void Video : :arreter ( ) [slot]

Références arreterTimers(), chargement, enregistrement, processus, et progression().

Référencé par Camera : :arreter(), arreterTimers(), demarrerTimers(), enregistrerVideo(), et progresser().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    if(enregistrement == true)
    {
        arreterTimers();
        processus->terminate();
        this->enregistrement = false;

        chargement = 0;
        emit progression(chargement);
    }
}
```

9.15.3.2 void Video : :arreterEnregistrements ( ) [static]

Méthode statique

Références processus.

Référencé par IHMWismas : :~IHMWismas().

```
{
    QString programme = "killall"; // le nom du programme
    QStringList arguments; // arguments du programme

    arguments << "vlc";

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << programme << arguments;
    #endif

    QProcess *processus = new QProcess();
    processus->startDetached(programme, arguments);
    delete processus;
}
```

9.15.3.3 void Video : :arreterTimers ( ) [private]

Références arreter(), progresser(), timerEnregistrement, et timerProgression.

Référencé par arreter().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    if(timerProgression->isActive())
        timerProgression->stop();
    if(timerEnregistrement->isActive())
        timerEnregistrement->stop();

    disconnect(timerProgression, SIGNAL(timeout()), this, SLOT(progresser()));
    disconnect(timerEnregistrement, SIGNAL(timeout()), this, SLOT(arreter()));
}
```

9.15.3.4 void Video : :creerCheminVideo (int *numeroCamera*) [private]

Références cheminVideo, et fichierINI.

Référencé par enregistrerVideo().

```
{
    QSettings configuration(fichierINI, QSettings::IniFormat);
    QString identifiantCamera = "Camera" + QString::number(numeroCamera);

    cheminVideo = configuration.value(identifiantCamera + "/chemin_video", "
      ./videos/").toString();

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << cheminVideo;
    #endif
}
```

9.15.3.5 void Video : :demarrerTimers (int *duree*) [private]

Paramètres

duree	int duree d'enregistrement

Références arreter(), progresser(), timerEnregistrement, et timerProgression.

Référencé par enregistrerVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    if(duree != 0)
    {
        timerProgression->start((duree*1000)/100);
        timerEnregistrement->start(duree*1000);
        connect(timerProgression, SIGNAL(timeout()), this, SLOT(progresser()));
        connect(timerEnregistrement, SIGNAL(timeout()), this, SLOT(arreter()));

        #ifdef DEBUG
        qDebug() << Q_FUNC_INFO << "timer progression" << ((duree*1000)/100);
        qDebug() << Q_FUNC_INFO << "timer enregistrement" << (duree*1000);
        #endif
    }
}
```

9.15.3.6 void Video : :enregistrerVideo (const int *numeroCamera,* const QString *nomSite,* const QString *adresseIP,* const QString *numeroPort,* const QString *identifiant,* const QString *motDePasse,* const int *duree*)

Paramètres

numero-Camera	int le numéro de la caméra
nomSite	QString le nom du site
adresseIP	QString l'adresse IP
numeroPort	QString le numéro de port
identifiant	QString le nom de l'administrateur
motDePasse	QString le mot de passe administrateur
duree	int le temps d'enregistrement

Références arreter(), chargement, cheminVideo, creerCheminVideo(), demarrer-Timers(), duree, enregistrement, getNumeroVideo(), numeroVideo, processus, et verifierDossierEnregistrements().

Référencé par Camera : :enregistrer().

```
{
    QDir repertoire(cheminVideo);

    QString programme; // Le nom du programme
    QStringList arguments; // Arguments du programme

    QString nomFichier;
    QString extension;

    /*
        Commandes d'enregistrement video avec cvlc :

        Le flux video de la camera est en MJPEG
```

```
        - en mjpg : cvlc "http://192.168.52.221:99/
      videostream.cgi?user=admin&pwd=&resolution=32&rate=0" --sout file/ts:./video-test.mjpg;

        - en mp4 : cvlc "http://192.168.52.221:99/
      videostream.cgi?user=admin&pwd=&resolution=32&rate=0" --sout '#transcode{vcodec=h264}
       :file{mux=ts,dst=test.mpg}'
    */

    if(duree != 0)
    {
        // Enregistrement en cours ?
        if(enregistrement == true)
        {
            #ifdef DEBUG
            qDebug() << Q_FUNC_INFO << "Fin de l'enregistrement";
            #endif

            arreter();
        }

        creerCheminVideo(numeroCamera);

        extension = ".mp4";
        nomFichier = cheminVideo + QString::number(getNumeroVideo()) + "_" +
      nomSite + extension;

        verifierDossierEnregistrements(repertoire);

        // Format mp4
        programme = "./cvlc.sh";
        if(motDePasse.length() == 0)
        {
            arguments   << adresseIP
                        << numeroPort
                        << identifiant
                        << nomFichier
                        << QString::number(duree);
        }
        else
        {
            arguments   << adresseIP
                        << numeroPort
                        << identifiant
                        << motDePasse
                        << nomFichier
                        << QString::number(duree);
        }

        #ifdef DEBUG
        //qDebug() << Q_FUNC_INFO << programme << arguments;
        qDebug() << Q_FUNC_INFO << "FICHIER : " + nomFichier;
        #endif

        processus->start(programme, arguments);

        enregistrement = true;
        chargement = 0;
        numeroVideo++;

        this->duree = duree;
        demarrerTimers(duree);
    }
}
```

9.15.3.7 void Video : :fini ( ) const [signal]

9.15.3.8 int Video : :getNumeroVideo ( ) [private]

Renvoie

int le nombre de vidéos

Références fichierINI, et numeroVideo.

Référencé par enregistrerVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << numeroVideo;
    #endif

    QSettings configuration(fichierINI, QSettings::IniFormat);
    int totalEnregistrement = configuration.value("nb_videos", "1").toInt();

    if(numeroVideo > totalEnregistrement)
        numeroVideo = 1;

    return numeroVideo;
}
```

9.15.3.9 void Video : :progresser ( ) [slot]

Références arreter(), chargement, duree, et progression().

Référencé par arreterTimers(), et demarrerTimers().

```
{
    if(chargement == duree*1000)
    {
        #ifdef DEBUG
        qDebug() << Q_FUNC_INFO << "Fin de l'enregistrement";
        #endif

        arreter();
    }
    else
    {
        chargement += ((duree*1000)/100); // en pourcentage
        emit progression(((double)chargement/((double)duree*1000.))*100.);

        #ifdef DEBUG
        //qDebug() << Q_FUNC_INFO <<
       (((double)chargement/((double)duree*1000.))*100.) << chargement << (duree*1000);
        #endif
    }
}
```

9.15.3.10 void Video : :progression (int *chargement*) const [signal]

Référencé par arreter(), et progresser().

9.15.3.11 void Video : :supprimerEnregistrements ( ) const [slot]

Les enregistrements sont supprimés selon le niveau de supression indiqué dans le fichier de configuration

Paramètres

nomSite	QString le nom du site

Références cheminVideo, fichierINI, supprimerNbEnregistrements(), et supprimerTous-LesEnregistrements().

Référencé par Camera : :supprimer().

```
{
    QSettings configuration(fichierINI, QSettings::IniFormat);

    QDir repertoire(cheminVideo);

    QString typeSuppression = configuration.value("niveau_suppression", "Aucun"
      ).toString();

    /*
        Un nom de fichier vidéo enregistrée :

        - en mp4 : 1_Alpin.mp4
    */

    if(repertoire.exists())
    {
        if(typeSuppression == "Tous")   // Tous les fichiers seront supprimés
        {
            supprimerTousLesEnregistrements(repertoire);
        }
        else if(typeSuppression == "Nb_enregistrements")    // Un fichier (le
       getNombreVideos() le plus récent) de chaque caméra sauvegardé
        {
            supprimerNbEnregistrements(repertoire);
        }
        else if(typeSuppression == "Aucun") // Aucun fichiers supprimés
        {
            // Ne rien faire
        }
        else    // Valeur niveau_suppression invalide
        {
            // Ne rien faire
        }
    }
}
```

9.15.3.12 void Video : :supprimerNbEnregistrements (QDir & *repertoire*) const [private]

Les enregistrements les plus anciens sont supprimés

Paramètres

repertoire	QDir le répertoire où se trouve les enregistrements
nb-Enregistrements-AConserver	int le nombre d'enregistrements à conserver
nomSite	QString le nom du site de la caméra

Références enregistrement.

Référencé par supprimerEnregistrements().

```
{
    QStringList filtre;
    QStringList fichier;
    QString extension = ".mp4";
    filtre << "*" + extension;
```

```
    repertoire.setSorting(QDir::Time);

    fichier += repertoire.entryList(filtre);

    foreach(QString enregistrement, repertoire.entryList(QDir::NoDotAndDotDot |
       QDir::AllEntries))
    {
        if(enregistrement != fichier[0])
            repertoire.remove(enregistrement);

    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO << fichier[0];
    #endif
    }
}
```

9.15.3.13 void Video : :supprimerTousLesEnregistrements (QDir & *repertoire*) const [private]

Les enregistrements sont tous supprimés

Paramètres

repertoire	QDir le répertoire où se trouve les enregistrements

Références enregistrement.

Référencé par supprimerEnregistrements().

```
{
    QStringList filtre;
    QString extension = ".mp4";
    filtre << "*" + extension;

    foreach(QString enregistrement, repertoire.entryList(filtre))
    {
        repertoire.remove(enregistrement);

        #ifdef DEBUG
        for(int compteur = 0; compteur < enregistrement.length(); compteur++)
            qDebug() << Q_FUNC_INFO << repertoire.count() << enregistrement.at(
      compteur);
        #endif
    }
}
```

9.15.3.14 void Video : :verifierDossierEnregistrements (QDir & *repertoire*) const [private]

Paramètres

repertoire	QDir le répertoire où se trouve les enregistrements

Références cheminVideo.

Référencé par enregistrerVideo().

```
{
    #ifdef DEBUG
    qDebug() << Q_FUNC_INFO;
    #endif

    if(!QDir(cheminVideo).exists())
```

```
    {
        repertoire.mkdir(cheminVideo);
    }
}
```

9.15.4 Documentation des données membres

9.15.4.1 int Video : :chargement [private]

Référencé par arreter(), enregistrerVideo(), progresser(), et Video().

9.15.4.2 QString Video : :cheminVideo [private]

Référencé par creerCheminVideo(), enregistrerVideo(), supprimerEnregistrements(), et verifierDossierEnregistrements().

9.15.4.3 int Video : :duree [private]

Référencé par enregistrerVideo(), et progresser().

9.15.4.4 bool Video : :enregistrement [private]

Référencé par arreter(), enregistrerVideo(), supprimerNbEnregistrements(), supprimer-TousLesEnregistrements(), et Video().

9.15.4.5 QString Video : :fichierINI [private]

Référencé par creerCheminVideo(), getNumeroVideo(), supprimerEnregistrements(), et Video().

9.15.4.6 int Video : :numeroVideo [private]

Référencé par enregistrerVideo(), getNumeroVideo(), et Video().

9.15.4.7 QProcess* Video : :processus [private]

Référencé par arreter(), arreterEnregistrements(), enregistrerVideo(), et Video().

9.15.4.8 QTimer* Video : :timerEnregistrement [private]

Référencé par arreterTimers(), demarrerTimers(), et Video().

9.15.4.9 QTimer* Video : :timerProgression [private]

Référencé par arreterTimers(), demarrerTimers(), et Video().

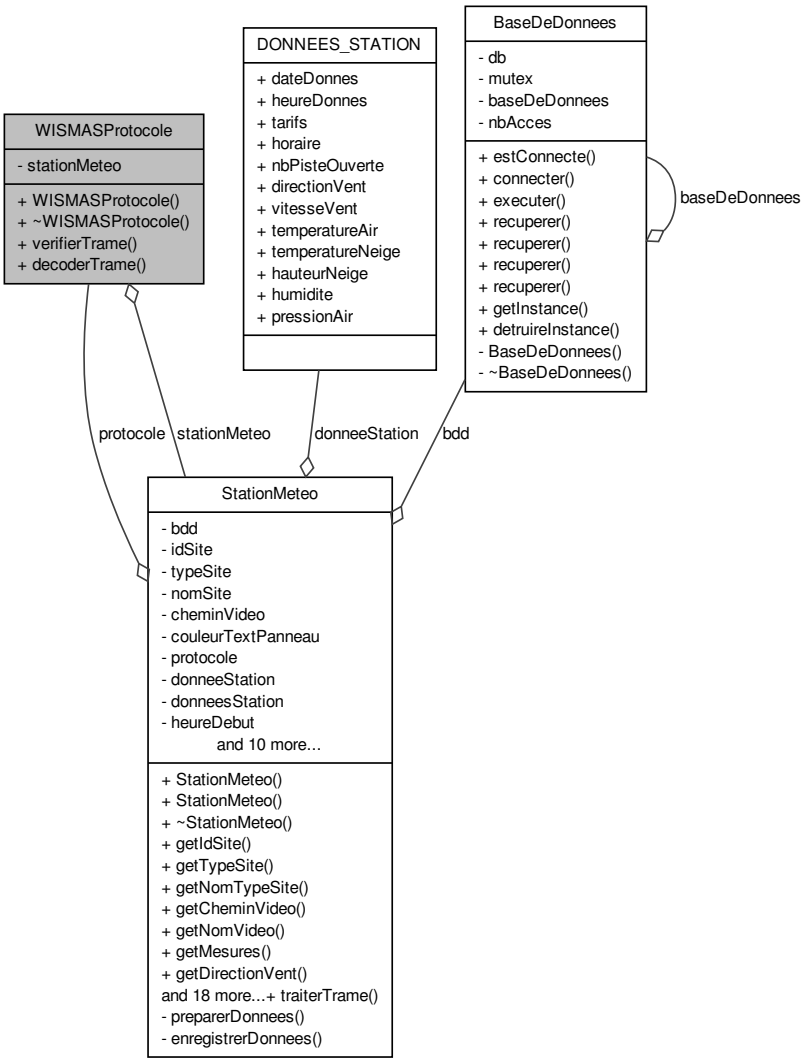
La documentation de cette classe a été générée à partir des fichiers suivants :
– Video.h
– Video.cpp

9.16 Référence de la classe WISMASProtocole

Gère la transmission entre les sites.

`#include <wismasprotocole.h>`

Graphe de collaboration de WISMASProtocole :



Fonctions membres publiques

– WISMASProtocole (StationMeteo *stationMeteo=0)

WISMASProtocole() constructeur par défaut.
- ~WISMASProtocole ()
  WISMASProtocole() destructeur.
- bool verifierTrame (QString trame)
  verifie la validite de la trame
- bool decoderTrame (QString trame)
  decode la trame et extrait les donnees pour la station météo

Attributs privés

- StationMeteo * stationMeteo
  Association vers la classe StationMeteo.

9.16.1 Description détaillée

Auteur

PETRELLA Olivier

Version

1.1

Date

18 février 2018

9.16.2 Documentation des constructeurs et destructeur

9.16.2.1 WISMASProtocole : :WISMASProtocole (StationMeteo * *stationMeteo* = 0)

```
                                                    : stationMeteo(
    stationMeteo)
{

}
```

9.16.2.2 WISMASProtocole : :~WISMASProtocole ( )

```
{

}
```

9.16.3 Documentation des fonctions membres

9.16.3.1 bool WISMASProtocole : :decoderTrame (QString *trame*)

Paramètres

trame	QString une trame

Renvoie

bool true si la trame est decodee sinon false

Références DONNEES_STATION : :dateDonnes, DONNEES_STATION : :directionVent, StationMeteo : :getIdSite(), StationMeteo : :getTypeSite(), DONNEES_STATION : :hauteurNeige, DONNEES_STATION : :heureDonnes, DONNEES_STATION : :humidite, DONNEES_STATION : :pressionAir, StationMeteo : :setDonneesTrame(), stationMeteo, DONNEES_STATION : :temperatureAir, DONNEES_STATION : :temperatureNeige, verifierTrame(), et DONNEES_STATION : :vitesseVent.

Référencé par StationMeteo : :traiterTrame().

```
{
    if(!verifierTrame(trame))
        return false;

    //qDebug() << Q_FUNC_INFO << trame;

    // Extraire les champs de la trame
    QStringList champs = trame.split(',');

    // Extraire les valeurs ID et TYPE des champs de la trame
    QStringList id = champs.at(0).split('.');
    QStringList type = champs.at(1).split('.');

    // Bonne station ?
    if(id.at(1).toInt() == stationMeteo->getIdSite())
    {
        // Bon type ?
        if(type.at(1).toInt() == stationMeteo->getTypeSite())
        {
            // Extraire les données des champs de la trame
            QString direction_vent = champs.at(2).split('.')[1];
            QString vitesse_vent = champs.at(3).split('.')[1];
            QString temperature_air = champs.at(4).split('.')[1];
            QString temperature_neige = champs.at(5).split('.')[1];
            QString hauteur_neige = champs.at(6).split('.')[1];
            QString humidite = champs.at(7).split('.')[1];
            QString pression_air = champs.at(8).split('.')[1].split('/')[0];
            pression_air.chop(0);

            DONNEES_STATION donneeStation;
            QDate date = QDate::currentDate();
            QTime heure = QTime::currentTime();

            donneeStation.dateDonnes = date.toString("dd/MM/yyyy");
            donneeStation.heureDonnes = heure.toString("hh:mm:ss");
            donneeStation.directionVent = direction_vent;
            donneeStation.vitesseVent = vitesse_vent;
            donneeStation.temperatureAir = temperature_air;
            donneeStation.temperatureNeige = temperature_neige;
            donneeStation.hauteurNeige = hauteur_neige;
            donneeStation.humidite = humidite;
            donneeStation.pressionAir = pression_air;

            stationMeteo->setDonneesTrame(donneeStation);

            return true;
        }
    }

    return false;
}
```

9.16.3.2 bool WISMASProtocole : :verifierTrame (QString *trame*)

Paramètres

trame	QString une trame

Renvoie

bool true si la trame est valide sinon false

Référencé par decoderTrame().

```
{
    if(trame.length() == 0)
        return false;

    if(!trame.endsWith("/r"))
        return false;

    return true;
}
```

9.16.4 Documentation des données membres

9.16.4.1 StationMeteo* WISMASProtocole : :stationMeteo [private]

Référencé par decoderTrame().

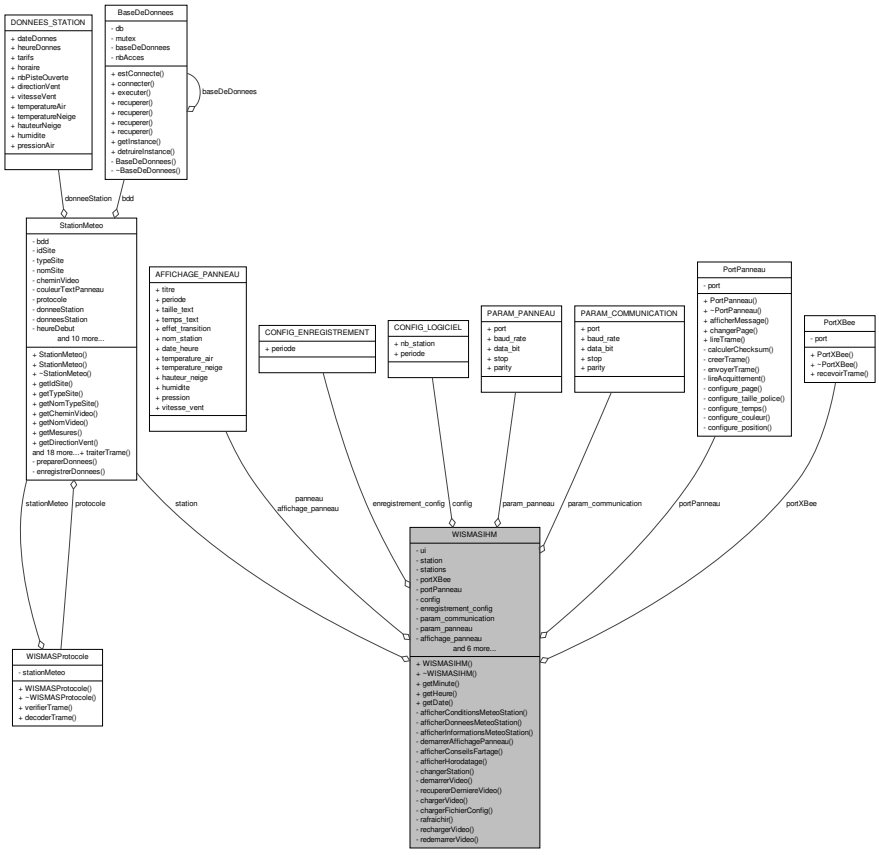
La documentation de cette classe a été générée à partir des fichiers suivants :
– wismasprotocole.h
– wismasprotocole.cpp

9.17 Référence de la classe WISMASIHM

Gère l'interface des sites de haute montagne.

`#include <wismasihm.h>`

Graphe de collaboration de WISMASIHM :



Fonctions membres publiques

- WISMASIHM (QWidget ∗parent=0)
  WISMASIHM : :WISMASIHM.
- ∼WISMASIHM ()
  WISMASIHM : :∼WISMASIHM.
- QString getMinute ()
  WISMASIHM : :getMinute.
- QString getHeure ()
  WISMASIHM : :getHeure.
- QString getDate ()

Connecteurs privés

- void rafraichir ()
  SLOT WISMASIHM : :rafraichir rafraichir l'IHM.
- void rechargerVideo ()
  SLOT permettant de recharger la vidéo pour une lecture en boucle.

– void redemarrerVideo ()
  SLOT permettant de relire la vidéo.

Fonctions membres privées

– void afficherConditionsMeteoStation ()
  WISMASIHM : :afficherConditionsMeteoStation affiche sur l'écran de diffusion (écran TV) l'ensemble des données météos d'un site et les conseils de fartage.
– void afficherDonneesMeteoStation ()
  WISMASIHM : :afficherDonneesMeteoStation assure l'affichage des mesures de la station météo d'un site.
– void afficherInformationsMeteoStation ()
  SLOT WISMASIHM : :afficherInformationsMeteoStation.
– void demarrerAffichagePanneau ()
  WISMASIHM : :demarrerAffichagePanneau affiche sur le panneau lumineux la synthèse des données météos d'un site.
– void afficherConseilsFartage ()
  WISMASIHM : :afficherConseilsFartage affiche les conseils de fartage.
– void afficherHorodatage ()
  WISMASIHM : :afficherHorodatage met à jour l'affichage de la date et l'heure.
– void changerStation ()
  WISMASIHM : :changerStation change de station.
– void demarrerVideo ()
  démarre une vidéo
– void recupererDerniereVideo ()
– void chargerVideo ()
  charge la vidéo enregistrée sur le serveur
– bool chargerFichierConfig ()
  WISMASIHM : :chargerFichierConfig.

Attributs privés

– Ui : :WISMASIHM ∗ ui
– StationMeteo ∗ station
  Association vers la classe StationMeteo.
– QVector< StationMeteo ∗ > stations
– PortXBee ∗ portXBee
  Association vers la classe PortXBee.
– PortPanneau ∗ portPanneau
  Association vers la classe PortPanneau.
– CONFIG_LOGICIEL config
– CONFIG_ENREGISTREMENT enregistrement_config
– PARAM_COMMUNICATION param_communication
– PARAM_PANNEAU param_panneau
– AFFICHAGE_PANNEAU affichage_panneau
– AFFICHAGE_PANNEAU panneau
– Phonon : :MediaObject ∗ media
– QTimer ∗ m_timer
  Timer pour le rafraichissement de la fenetre principale.
– int idStationCourante
– long m_valeur
– long pagePanneau_compteur
– int nb_page

9.17.1 Description détaillée

Auteur

PETRELLA Olivier

Version

1.1

Date

18 février 2018

9.17.2 Documentation des constructeurs et destructeur

9.17.2.1 WISMASIHM : :WISMASIHM (QWidget * *parent* = 0) [explicit]

Paramètres

QWidget	parent, Ui ui, QTimer m_timer, int idStationCourante, int m_valeur, int pagePanneau_compteur(0), int nb_page.

Références PortPanneau : :afficherMessage(), chargerFichierConfig(), demarrerVideo(), StationMeteo : :getInformationComplementaire(), idStationCourante, m_timer, media, portPanneau, portXBee, rafraichir(), rechargerVideo(), redemarrerVideo(), station, stations, et ui.

```
                                    : QWidget(parent), ui(new Ui::WISMASIHM),
      m_timer(NULL), idStationCourante(0), m_valeur(0), pagePanneau_compteur(0),
      nb_page(0)
{
    ui->setupUi(this);

    QAction *actionQuitter = new QAction("&Quitter", this);
    actionQuitter->setShortcut(QKeySequence(QKeySequence::Quit)); // Ctrl+Q
    addAction(actionQuitter);
    connect(actionQuitter, SIGNAL(triggered()), this, SLOT(close()));

    // Lecture de la configuration
    if(!this->chargerFichierConfig())
    {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << "Erreur : Lecture du fichier a échoué !"
      ;
        #endif

    }

    // Timer pour le rafraichissement de la fenetre principale
    m_timer = new QTimer(this);
    m_timer->start(1000);
    connect(m_timer, SIGNAL(timeout()), this, SLOT(rafraichir()));

    // Gestion des vidéos
    media = new Phonon::MediaObject(this);
    Phonon::createPath(media, ui->vwidget);

    // gestion des stations installées sur site
    if(stations.count() != 0)
    {
        // on prend le premier site pour l'affichage
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << "Site : " << idStationCourante <<
```

```
        stations.at((idStationCourante%stations.count()))->getIdSite() << stations.at((
        idStationCourante%stations.count()))->getNomTypeSite();
          #endif

          ui->nom_site->setText(stations.at(idStationCourante)->getNomTypeSite())
        ;

          connect(media, SIGNAL(aboutToFinish()), this, SLOT(rechargerVideo()));
          connect(media, SIGNAL(finished()), this, SLOT(redemarrerVideo()));
          demarrerVideo();

          portXBee = new PortXBee(this);
          for(int i = 0; i < stations.count(); i++)
          {
              connect(portXBee, SIGNAL(nouvelleTrame(QString)), stations.at(i),
        SLOT(traiterTrame(QString)));
          }

          portPanneau = new PortPanneau(this);

          portPanneau->afficherMessage(stations.at((idStationCourante%stations.
        count()))->getNomTypeSite() + ";" + QDateTime::currentDateTime().toString("
        dd/MM/yy") + ";" + QDateTime::currentDateTime().toString("HH:mm") + ";" + station->
        getInformationComplementaire(stations.at((idStationCourante%stations.count()))
        ->getIdSite())[0] + "; Horaire " + station->getInformationComplementaire(stations
        .at((idStationCourante%stations.count()))->getIdSite())[1] + "; Piste " + station
        ->getInformationComplementaire(stations.at((idStationCourante%stations.count()))
        ->getIdSite())[2] + ";" + stations.at((idStationCourante%stations.count()))->
        getTemperatureAir() + ";" + "Vent " + stations.at((idStationCourante%stations.count
        ()))->getVitesseVent() + ";" + "Neige " + stations.at((idStationCourante%stations
        .count()))->getHauteurNeige());
    }
    else
    {
        #ifdef DEBUG

            qDebug() << Q_FUNC_INFO << "Erreur Aucune station !";
        #endif

    }

}
```

9.17.2.2 WISMASIHM : :~WISMASIHM ( )

Références ui.

```
{
    qDebug() << Q_FUNC_INFO;
    delete ui;
}
```

9.17.3 Documentation des fonctions membres

9.17.3.1 void WISMASIHM : :afficherConditionsMeteoStation ( ) [private]

Références afficherConseilsFartage(), et afficherDonneesMeteoStation().

Référencé par rafraichir().

```
{
    afficherDonneesMeteoStation();

    afficherConseilsFartage();
}
```

9.17.3.2 void WISMASIHM : :afficherConseilsFartage ( ) [private]

Renvoie

void

Références idStationCourante, stations, et ui.

Référencé par afficherConditionsMeteoStation().

```
{
    ui->label_SWIX->setText(stations.at((idStationCourante))->
      getConseilsFartage(1));
    ui->label_TOKO->setText(stations.at((idStationCourante))->
      getConseilsFartage(2));
    ui->label_VOLA->setText(stations.at((idStationCourante))->
      getConseilsFartage(3));
    ui->label_BRIK->setText(stations.at((idStationCourante))->
      getConseilsFartage(4));
    ui->label_DRAG->setText(stations.at((idStationCourante))->
      getConseilsFartage(5));
    ui->label_XCSO->setText(stations.at((idStationCourante))->
      getConseilsFartage(6));
}
```

9.17.3.3 void WISMASIHM : :afficherDonneesMeteoStation ( ) [private]

Renvoie

void

Références DIRECTION_VENT, HAUTEUR_NEIGE, HUMIDITE, idStationCourante, PRESSION_AIR, stations, TEMPERATURE_AIR, TEMPERATURE_NEIGE, ui, et VITESSE_VENT.

Référencé par afficherConditionsMeteoStation().

```
{
    ui->nom_site->setText(stations.at((idStationCourante))->getNomTypeSite());

    ui->label_directionVent->setText(stations.at((idStationCourante))->
      getMesures(DIRECTION_VENT));
    ui->label_vitesseVent->setText(stations.at((idStationCourante))->getMesures
      (VITESSE_VENT));
    ui->label_temperature->setText(stations.at((idStationCourante))->getMesures
      (TEMPERATURE_AIR));
    ui->label_temperatureNeige->setText(stations.at((idStationCourante))->
      getMesures(TEMPERATURE_NEIGE));
    ui->label_hauteurNeige->setText(stations.at((idStationCourante))->
      getMesures(HAUTEUR_NEIGE));
    ui->label_humidite->setText(stations.at((idStationCourante))->getMesures(
      HUMIDITE));
    ui->label_pression->setText(stations.at((idStationCourante))->getMesures(
      PRESSION_AIR));
}
```

9.17.3.4 void WISMASIHM : :afficherHorodatage ( ) [private]

Références ui.

Référencé par rafraichir().

```
{
    ui->date_heure->setText("    " + QDateTime::currentDateTime().toString("
      dd/MM/yy HH:mm"));
}
```

9.17.3.5 void WISMASIHM : :afficherInformationsMeteoStation ( ) [private]

Références PortPanneau : :changerPage(), nb_page, pagePanneau_compteur, PANNEAU_NB_PAGES_MAX, et portPanneau.

Référencé par rafraichir().

```
{
    pagePanneau_compteur++;

    if((this->pagePanneau_compteur % 10) == 0)
    {
        portPanneau->changerPage(nb_page);
        nb_page++;

        if(nb_page == PANNEAU_NB_PAGES_MAX)
        {
            nb_page = 0;
        }
    }
}
```

9.17.3.6 void WISMASIHM : :changerStation ( ) [private]

Références idStationCourante, et stations.

Référencé par rafraichir().

```
{

    ++idStationCourante;

    #ifdef DEBUG

        qDebug() << Q_FUNC_INFO << "Site : " << idStationCourante << stations.
      at((idStationCourante%stations.count()))->getIdSite() << stations.at((
      idStationCourante%stations.count()))->getNomTypeSite();
    #endif


    idStationCourante = (idStationCourante%stations.count());

}
```

9.17.3.7 bool WISMASIHM : :chargerFichierConfig ( ) [private]

Renvoie

bool

Références affichage_panneau, PARAM_COMMUNICATION : :baud_rate, PARAM_PANNEAU : :baud_rate, config, PARAM_COMMUNICATION : :data_bit, PARAM_PANNEAU : :data_bit, AFFICHAGE_PANNEAU : :date_heure, AFFICHAGE_PANNEAU : :effet_transition, enregistrement_config, AFFICHAGE_PANNEAU : :hauteur_neige,

AFFICHAGE_PANNEAU : :humidite, CONFIG_LOGICIEL : :nb_station, AFFICHAGE_-PANNEAU : :nom_station, param_communication, param_panneau, PARAM_COMM-UNICATION : :parity, PARAM_PANNEAU : :parity, CONFIG_LOGICIEL : :periode, CO-NFIG_ENREGISTREMENT : :periode, AFFICHAGE_PANNEAU : :periode, PARAM_C-OMMUNICATION : :port, PARAM_PANNEAU : :port, AFFICHAGE_PANNEAU : :pres-sion, station, stations, PARAM_COMMUNICATION : :stop, PARAM_PANNEAU : :stop, AFFICHAGE_PANNEAU : :taille_text, AFFICHAGE_PANNEAU : :temperature_air, A-FFICHAGE_PANNEAU : :temperature_neige, AFFICHAGE_PANNEAU : :temps_text, AFFICHAGE_PANNEAU : :titre, et AFFICHAGE_PANNEAU : :vitesse_vent.

Référencé par WISMASIHM().

```
{
    // Le nom du fichier INI : chemin-executable/nom-executable.ini
    QString fichierINI =  qApp->applicationDirPath() + "/" + qApp->
      applicationName() + ".ini";

    #ifdef DEBUG

        qDebug() << Q_FUNC_INFO << "Nom fichier INI : " + fichierINI;
    #endif


    if(QFile::exists(fichierINI))
    {
        QSettings parametres(fichierINI, QSettings::IniFormat);

        // Chargement des informations logiciel
        this->config.nb_station                     = parametres.value("
      Configuration/nb_station", "0").toInt();
        this->config.periode                        = parametres.value("
      Configuration/periode", "90").toInt();

        // Chargement des informations d'enregistrement en base de données
        this->enregistrement_config.periode         = parametres.value("
      Enregistrement/periode", "60").toInt();

        // Chargement de la configuration du recepteur station
        this->param_communication.port              = parametres.value("
      Station_configuration/port", "0").toString();
        this->param_communication.baud_rate         = parametres.value("
      Station_configuration/baud_rate", "0").toInt();
        this->param_communication.data_bit          = parametres.value("
      Station_configuration/data_bit", "0").toInt();
        this->param_communication.stop              = parametres.value("
      Station_configuration/stop", "0").toInt();
        this->param_communication.parity            = parametres.value("
      Station_configuration/parity", "0").toInt();

        // Chargement de la configuration du panneau lumineux
        this->param_panneau.port                    = parametres.value("
      Panneau_configuration/port", "0").toString();
        this->param_panneau.baud_rate               = parametres.value("
      Panneau_configuration/baud_rate", "0").toInt();
        this->param_panneau.data_bit                = parametres.value("
      Panneau_configuration/data_bit", "0").toInt();
        this->param_panneau.stop                    = parametres.value("
      Panneau_configuration/stop", "0").toInt();
        this->param_panneau.parity                  = parametres.value("
      Panneau_configuration/parity", "0").toInt();

        // Chargement des paramètres d'affichage du panneau lumineux
        this->affichage_panneau.titre               = parametres.value("
      Panneau_affichage/titre", "0").toString();
        this->affichage_panneau.periode             = parametres.value("
      Panneau_affichage/periode", "0").toInt();
        this->affichage_panneau.taille_text         = parametres.value("
      Panneau_affichage/taille_text", "0").toString();
        this->affichage_panneau.temps_text          = parametres.value("
```

```
    Panneau_affichage/temps_text", "0").toInt();
      this->affichage_panneau.effet_transition      = parametres.value("
    Panneau_affichage/effet_transition", "0").toString();
      this->affichage_panneau.nom_station           = parametres.value("
    Panneau_affichage/nom_station", "0").toBool();
      this->affichage_panneau.date_heure            = parametres.value("
    Panneau_affichage/date_heure", "0").toBool();
      this->affichage_panneau.temperature_air       = parametres.value("
    Panneau_affichage/temperature_air", "0").toBool();
      this->affichage_panneau.temperature_neige     = parametres.value("
    Panneau_affichage/temperature_neige", "0").toBool();
      this->affichage_panneau.hauteur_neige         = parametres.value("
    Panneau_affichage/hauteur_neige", "0").toBool();
      this->affichage_panneau.humidite              = parametres.value("
    Panneau_affichage/humidite", "0").toBool();
      this->affichage_panneau.pression              = parametres.value("
    Panneau_affichage/pression", "0").toBool();
      this->affichage_panneau.vitesse_vent          = parametres.value("
    Panneau_affichage/vitesse_vent", "0").toBool();

      // Chargement des paramètres des stations

      for(int i=0; i < this->config.nb_station; i++)
      {
          int idSite                  = parametres.value(QString("
    Station%1/id").arg(i+1), "").toInt();
          int typeSite                = parametres.value(QString("
    Station%1/type").arg(i+1), "").toInt();
          QString nomSite             = parametres.value(QString("
    Station%1/nom").arg(i+1), "").toString();
          QString cheminVideo         = parametres.value(QString("
    Station%1/chemin_video").arg(i+1), "").toString();
          QString couleurTextPanneau  = parametres.value(QString("
    Station%1/couleur_text_panneau").arg(i+1), "").toString();

          if(!nomSite.isEmpty())
          {
              station = new StationMeteo(idSite, typeSite, nomSite,
    cheminVideo, couleurTextPanneau, this);
              stations.push_back(station);
          }
      }
  }
  else
  {
      #ifdef DEBUG

          qDebug() << "Erreur Fichier " + fichierINI + " absent !";
      #endif

      return false;
  }
  return true;
}
```

9.17.3.8 void WISMASIHM : :chargerVideo ( ) [private]

A faire charger la dernière video.

Références idStationCourante, media, et stations.

Référencé par demarrerVideo().

```
{
    if(QFile(qApp->applicationDirPath() + stations.at(idStationCourante)->
      getCheminVideo() + stations.at(idStationCourante)->getNomVideo()).exists())
    {
        media->setCurrentSource(qApp->applicationDirPath() + stations.at(
      idStationCourante)->getCheminVideo() + stations.at(idStationCourante)->
      getNomVideo());
```

```
    }
    else
    {
        media->setCurrentSource(qApp->applicationDirPath() + "
      /videos/defaut.mp4");
    }
}
```

9.17.3.9 void WISMASIHM : :demarrerAffichagePanneau ( ) [private]

Références PortPanneau : :afficherMessage(), StationMeteo : :getInformation-Complementaire(), idStationCourante, portPanneau, station, et stations.

Référencé par rafraichir().

```
{
        portPanneau->afficherMessage(stations.at((idStationCourante%stations.
      count()))->getNomTypeSite() + ";" + QDateTime::currentDateTime().toString("
      dd/MM/yy") + ";" + QDateTime::currentDateTime().toString("HH:mm") + ";" + station->
      getInformationComplementaire(stations.at((idStationCourante%stations.count()))
      ->getIdSite())[0] + "; Horaire " + station->getInformationComplementaire(stations
      .at((idStationCourante%stations.count()))->getIdSite())[1] + "; Piste " + station
      ->getInformationComplementaire(stations.at((idStationCourante%stations.count()))
      ->getIdSite())[2] + ";" + stations.at((idStationCourante%stations.count()))->
      getTemperatureAir() + ";" + "Vent " + stations.at((idStationCourante%stations.count
      ()))->getVitesseVent() + ";" + "Neige " + stations.at((idStationCourante%stations
      .count()))->getHauteurNeige());
}
```

9.17.3.10 void WISMASIHM : :demarrerVideo ( ) [private]

Références chargerVideo(), media, et recupererDerniereVideo().

Référencé par rafraichir(), et WISMASIHM().

```
{
    recupererDerniereVideo();
    media->stop();
    media->clear();
    chargerVideo();
    media->play();
}
```

9.17.3.11 QString WISMASIHM : :getDate ( )

9.17.3.12 QString WISMASIHM : :getHeure ( )

Renvoie

QString de l'heure

```
{
    return QDateTime::currentDateTime().toString("HH");
}
```

9.17.3.13 QString WISMASIHM : :getMinute ( )

Renvoie

QString des minutes

```
{
    return QDateTime::currentDateTime().toString("mm");
}
```

9.17.3.14 void WISMASIHM : :rafraichir ( ) [private, slot]

Références afficherConditionsMeteoStation(), afficherHorodatage(), afficherInformations-MeteoStation(), changerStation(), config, demarrerAffichagePanneau(), demarrer-Video(), m_valeur, et CONFIG_LOGICIEL : :periode.

Référencé par WISMASIHM().

```
{
    m_valeur++;
    afficherHorodatage();

    if((m_valeur % config.periode) == 0)
    {
        changerStation();
        demarrerAffichagePanneau();
        demarrerVideo();
    }

    // écran TV
    afficherConditionsMeteoStation();


    // panneau lumineux
    afficherInformationsMeteoStation();
}
```

9.17.3.15 void WISMASIHM : :rechargerVideo ( ) [private, slot]

Références idStationCourante, media, et stations.

Référencé par WISMASIHM().

```
{
    if(QFile(qApp->applicationDirPath() + stations.at(idStationCourante)->
      getCheminVideo() + stations.at(idStationCourante)->getNomVideo()).exists())
    {
        media->enqueue(qApp->applicationDirPath() + stations.at(
      idStationCourante)->getCheminVideo() + stations.at(idStationCourante)->
      getNomVideo());
    }
    else
    {
        media->setCurrentSource(qApp->applicationDirPath() + "
      /videos/defaut.mp4");
    }
}
```

9.17.3.16 void WISMASIHM : :recupererDerniereVideo ( ) [private]

Références idStationCourante, et stations.

Référencé par demarrerVideo().

```
{
    //reperer les repertoires des stations
    QString chemin = QUrl::fromLocalFile(QCoreApplication::applicationDirPath()
       + stations.at(idStationCourante)->getCheminVideo()).toString();
    QStringList fichier;
    QStringList filtre;
    filtre << ".mp4";

    QDirIterator it(chemin.split("file://")[1], QDirIterator::Subdirectories);
    while (it.hasNext())
    {
        QDir dir(it.next());

        dir.setFilter(QDir::Files);
        dir.setSorting(QDir::Time);

        fichier += dir.entryList();
    }

    if(!fichier.isEmpty())
    {
        stations.at(idStationCourante)->setNomVideo(fichier[0]);
    }
}
```

9.17.3.17 void WISMASIHM : :redemarrerVideo ( ) [private, slot]

Références media.

Référencé par WISMASIHM().

```
{
    media->play();
}
```

9.17.4 Documentation des données membres

9.17.4.1 AFFICHAGE_PANNEAU WISMASIHM : :affichage_panneau [private]

Référencé par chargerFichierConfig().

9.17.4.2 CONFIG_LOGICIEL WISMASIHM : :config [private]

Référencé par chargerFichierConfig(), et rafraichir().

9.17.4.3 CONFIG_ENREGISTREMENT WISMASIHM : :enregistrement_config [private]

Référencé par chargerFichierConfig().

9.17.4.4 int WISMASIHM : :idStationCourante [private]

Référencé par afficherConseilsFartage(), afficherDonneesMeteoStation(), changerStation(), chargerVideo(), demarrerAffichagePanneau(), rechargerVideo(), recupererDerniereVideo(), et WISMASIHM().

9.17.4.5 QTimer* WISMASIHM : :m_timer [private]

Référencé par WISMASIHM().

9.17.4.6 long WISMASIHM : :m_valeur [private]

Référencé par rafraichir().

9.17.4.7 Phonon : :MediaObject* WISMASIHM : :media [private]

Référencé par chargerVideo(), demarrerVideo(), rechargerVideo(), redemarrerVideo(), et WISMASIHM().

9.17.4.8 int WISMASIHM : :nb_page [private]

Référencé par afficherInformationsMeteoStation().

9.17.4.9 long WISMASIHM : :pagePanneau_compteur [private]

Référencé par afficherInformationsMeteoStation().

9.17.4.10 AFFICHAGE_PANNEAU WISMASIHM : :panneau [private]

9.17.4.11 PARAM_COMMUNICATION WISMASIHM : :param_communication [private]

Référencé par chargerFichierConfig().

9.17.4.12 PARAM_PANNEAU WISMASIHM : :param_panneau [private]

Référencé par chargerFichierConfig().

9.17.4.13 PortPanneau* WISMASIHM : :portPanneau [private]

Référencé par afficherInformationsMeteoStation(), demarrerAffichagePanneau(), et WISMASIHM().

9.17.4.14 PortXBee* WISMASIHM : :portXBee [private]

Référencé par WISMASIHM().

9.17.4.15 StationMeteo* WISMASIHM : :station [private]

Référencé par chargerFichierConfig(), demarrerAffichagePanneau(), et WISMASIHM().

9.17.4.16 QVector<StationMeteo*> WISMASIHM : :stations [private]

Référencé par afficherConseilsFartage(), afficherDonneesMeteoStation(), changerStation(), chargerFichierConfig(), chargerVideo(), demarrerAffichagePanneau(), rechargerVideo(), recupererDerniereVideo(), et WISMASIHM().

9.17.4.17 Ui : :WISMASIHM∗ WISMASIHM : :ui [private]

Référencé par afficherConseilsFartage(), afficherDonneesMeteoStation(), afficherHorodatage(), WISMASIHM(), et ∼WISMASIHM().

La documentation de cette classe a été générée à partir des fichiers suivants :
– wismasihm.h
– wismasihm.cpp

10 Documentation des fichiers

10.1 Référence du fichier basededonnees.cpp

`#include "basededonnees.h" #include <QDebug> #include <QMessageBox>`

10.2 Référence du fichier basededonnees.h

`#include <QObject> #include <QtSql/QtSql> #include <QSqlDatabase> #include <QMutex>`

Classes

– class BaseDeDonnees

Macros

– #define HOSTNAME "localhost"
– #define USERNAME "root"
– #define PASSWORD "password"
– #define DATABASENAME "WISMAS_2018"

10.2.1 Documentation des macros

10.2.1.1 #define DATABASENAME "WISMAS_2018"

10.2.1.2 #define HOSTNAME "localhost"

Référencé par BaseDeDonnees : :connecter().

10.2.1.3 #define PASSWORD "password"

Référencé par BaseDeDonnees : :connecter().

10.2.1.4 #define USERNAME "root"

Référencé par BaseDeDonnees : :connecter().

10.3 Référence du fichier Camera.cpp

```
#include "Camera.h" #include "Structures.h"
```

10.3.1 Description détaillée

10.4 Référence du fichier Camera.h

```
#include <QString> #include <QUrl> #include "Parametrage.-
h" #include "Video.h"
```

Classes

– class Camera
Acquérir le système vidéo.

Macros

– #define DEBUG

10.4.1 Description détaillée

10.4.2 Documentation des macros

10.4.2.1 #define DEBUG

10.5 Référence du fichier Changelog.dox

10.6 Référence du fichier IHMWismas.cpp

```
#include "IHMWismas.h" #include "ui_IHMWismas.h"
```

10.6.1 Description détaillée

10.7 Référence du fichier IHMWismas.h

```
#include <QtGui> #include <QWidget> #include <QTimer> ×
#include <QLabel> #include <QDebug> #include <QVector>
#include <QNetworkRequest> #include <QNetworkAccess-
Manager> #include "Camera.h"
```

Classes

– class IHMWismas
Interface administrateur pour gérer le système vidéo.

Macros

– #define DEBUG

10.7.1 Description détaillée

10.7.2 Documentation des macros

10.7.2.1 #define DEBUG

10.8 Référence du fichier main.cpp

Programme principal WISMAS_station.

```
#include <QApplication> #include <QTextCodec> #include <-
QDebug> #include "wismasihm.h"
```

Fonctions

– int main (int argc, char ∗argv[])
Programme principal : Crée et affiche la fenêtre principale de l'application WISMAS_-station.

10.8.1 Description détaillée

Crée et affiche la fenêtre principale de l'application WISMAS_station

Version

1.1

10.8.2 Documentation des fonctions

10.8.2.1 main (int *argc*, char ∗ *argv[]*)

Main du système d'acquisition vidéo.

Paramètres

argc	
argv[]	

Renvoie

int

```
{
    QApplication a(argc, argv);
    QTextCodec::setCodecForCStrings(QTextCodec::codecForName("utf8"));
    a.setApplicationName("WISMAS_station");

    WISMASIHM w;
```

```
    //w.show();
    w.showFullScreen();

    return a.exec();
}
```

10.9 Référence du fichier main.cpp

Programme principal WISMAS_video.

`#include <QApplication> #include "IHMWismas.h" #include "-Camera.h"`

Fonctions

– int main (int argc, char ∗argv[])

10.9.1 Description détaillée

10.9.2 Documentation des fonctions

10.9.2.1 int main (int *argc,* char ∗ *argv[]*)

```
{
    QTextCodec::setCodecForCStrings(QTextCodec::codecForName("utf8"));
    QApplication a(argc, argv);
    IHMWismas w;

    w.showFullScreen();

    return a.exec();
}
```

10.10 Référence du fichier Parametrage.cpp

`#include "Parametrage.h"`

10.10.1 Description détaillée

10.11 Référence du fichier Parametrage.h

`#include <QDebug> #include <QSettings> #include <QApplication> × #include <QString> #include <QVector> #include "Structures.-h"`

Classes

– class Parametrage
Paramétrer le système vidéo.

Macros

– #define DEBUG

10.11.1 Description détaillée

10.11.2 Documentation des macros

10.11.2.1 #define DEBUG

10.12 Référence du fichier portpanneau.cpp

Définition de la classe PortPanneau.

```
#include "portpanneau.h" #include <unistd.h> #include <Q-
StringList> #include <QDebug>
```

10.12.1 Description détaillée

Auteur

Petrella Olivier

Version

1.1

10.13 Référence du fichier portpanneau.h

Déclaration de la classe PortPanneau.

```
#include <QObject>  #include "qextserialport.h"  #include
<QString> #include <QStringList>
```

Classes

– class PortPanneau
Communique avec les stations météo des sites via un port série virtuel.

Macros

– #define PORT_PANNEAU "/dev/ttyUSB1"
Fichier périphérique par défaut.
– #define PANNEAU_LG_MAX_TRAME 128
– #define PANNEAU_LG_MAX 16
– #define PANNEAU_LG_REPONSE 4
– #define PANNEAU_NUL 0x00
– #define PANNEAU_ACK "ACK"
– #define PANNEAU_NACK "NACK"
– #define PANNEAU_PAGE_A 'A'
– #define PANNEAU_PAGE_B 'B'

- #define PANNEAU_PAGE_C 'C'
- #define PANNEAU_PAGE_D 'D'
- #define PANNEAU_PAGE_E 'E'
- #define PANNEAU_PAGE_F 'F'
- #define PANNEAU_PAGE_G 'G'
- #define PANNEAU_PAGE_H 'H'
- #define PANNEAU_PAGE_I 'I'
- #define PANNEAU_IMMEDIATE 'A'
- #define PANNEAU_XOPEN 'B'
- #define PANNEAU_CURTAIN_UP 'C'
- #define PANNEAU_CURTAIN_DOWN 'D'
- #define PANNEAU_SCROLL_LEFT 'E'
- #define PANNEAU_SCROLL_RIGHT 'F'
- #define PANNEAU_VOPEN 'G'
- #define PANNEAU_VCLOSE 'H'
- #define PANNEAU_SCROLL_UP 'I'
- #define PANNEAU_SCROLL_DOWN 'J'
- #define PANNEAU_HOLD 'K'
- #define PANNEAU_SNOW 'L'
- #define PANNEAU_TWINKLE 'M'
- #define PANNEAU_BLOCK_MOVE 'N'
- #define PANNEAU_RANDOM 'P'
- #define PANNEAU_NORMAL 'A'
- #define PANNEAU_BLINK 'B'
- #define PANNEAU_SIZE_5X7 'A'
- #define PANNEAU_SIZE_6X7 'B'
- #define PANNEAU_BOLD SIZE_6X7
- #define PANNEAU_POSITION_CENTER 'C'
- #define PANNEAU_NB_PAGES_MAX 8
- #define DEMO

Définitions de type

- typedef unsigned char uint8_t

10.13.1 Description détaillée

Auteur

Version

1.1

10.13.2 Documentation des macros

10.13.2.1 #define DEMO

10.13.2.2 #define PANNEAU_ACK "ACK"

Référencé par PortPanneau : :lireAcquittement().

10.13.2.3 #define PANNEAU_BLINK 'B'

10.13.2.4 #define PANNEAU_BLOCK_MOVE 'N'

10.13.2.5 #define PANNEAU_BOLD SIZE_6X7

10.13.2.6 #define PANNEAU_CURTAIN_DOWN 'D'

10.13.2.7 #define PANNEAU_CURTAIN_UP 'C'

10.13.2.8 #define PANNEAU_HOLD 'K'

10.13.2.9 #define PANNEAU_IMMEDIATE 'A'

10.13.2.10 #define PANNEAU_LG_MAX 16

10.13.2.11 #define PANNEAU_LG_MAX_TRAME 128

10.13.2.12 #define PANNEAU_LG_REPONSE 4

10.13.2.13 #define PANNEAU_NACK "NACK"

Référencé par PortPanneau : :lireAcquittement().

10.13.2.14 #define PANNEAU_NB_PAGES_MAX 8

Référencé par WISMASIHM : :afficherInformationsMeteoStation().

10.13.2.15 #define PANNEAU_NORMAL 'A'

10.13.2.16 #define PANNEAU_NUL 0x00

10.13.2.17 #define PANNEAU_PAGE_A 'A'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.18 #define PANNEAU_PAGE_B 'B'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.19 #define PANNEAU_PAGE_C 'C'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.20 #define PANNEAU_PAGE_D 'D'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.21 #define PANNEAU_PAGE_E 'E'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.22 #define PANNEAU_PAGE_F 'F'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.23 #define PANNEAU_PAGE_G 'G'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.24 #define PANNEAU_PAGE_H 'H'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.25 #define PANNEAU_PAGE_I 'I'

Référencé par PortPanneau : :changerPage(), et PortPanneau : :creerTrame().

10.13.2.26 #define PANNEAU_POSITION_CENTER 'C'

10.13.2.27 #define PANNEAU_RANDOM 'P'

10.13.2.28 #define PANNEAU_SCROLL_DOWN 'J'

10.13.2.29 #define PANNEAU_SCROLL_LEFT 'E'

10.13.2.30 #define PANNEAU_SCROLL_RIGHT 'F'

10.13.2.31 #define PANNEAU_SCROLL_UP 'I'

10.13.2.32 #define PANNEAU_SIZE_5X7 'A'

Référencé par PortPanneau : :creerTrame().

10.13.2.33 #define PANNEAU_SIZE_6X7 'B'

10.13.2.34 #define PANNEAU_SNOW 'L'

10.13.2.35 #define PANNEAU_TWINKLE 'M'

10.13.2.36 #define PANNEAU_VCLOSE 'H'

10.13.2.37 #define PANNEAU_VOPEN 'G'

10.13.2.38 #define PANNEAU_XOPEN 'B'

10.13.2.39 #define PORT_PANNEAU "/dev/ttyUSB1"

Référencé par PortPanneau : :PortPanneau().

10.13.3 Documentation des définitions de type

10.13.3.1 typedef unsigned char uint8_t

10.14 Référence du fichier portxbee.cpp

Définition de la classe PortXBee.

```
#include "portxbee.h"  #include <unistd.h>  #include <Q-
Debug>
```

10.14.1 Description détaillée

Auteur

Petrella Olivier

Version

1.1

10.15 Référence du fichier portxbee.h

Déclaration de la classe PortXBee.

```
#include <QObject>  #include "qextserialport.h"  #include
<QString>
```

Classes

– class PortXBee
Communique avec les stations météo des sites via un port série virtuel.

Macros

– #define PORT_XBEE "/dev/ttyUSB0"
Fichier périphérique par défaut.

10.15.1 Description détaillée

Auteur

Version

1.1

10.15.2 Documentation des macros

10.15.2.1 #define PORT_XBEE "/dev/ttyUSB0"

Référencé par PortXBee : :PortXBee().

10.16 Référence du fichier qrc_ressources.cpp

```
#include <QtCore/qglobal.h>
```

Fonctions

- QT_BEGIN_NAMESPACE Q_CORE_EXPORT bool qRegisterResourceData (int, const unsigned char *, const unsigned char *, const unsigned char *)
- Q_CORE_EXPORT bool qUnregisterResourceData (int, const unsigned char *, const unsigned char *, const unsigned char *)
- QT_END_NAMESPACE int QT_MANGLE_NAMESPACE() qInitResources_ressources ()
- int QT_MANGLE_NAMESPACE() qCleanupResources_ressources ()

Variables

- static const unsigned char qt_resource_data []
- static const unsigned char qt_resource_name []
- static const unsigned char qt_resource_struct []

10.16.1 Documentation des fonctions

10.16.1.1 int QT_MANGLE_NAMESPACE() qCleanupResources_ressources ( )

Références qt_resource_data, qt_resource_name, qt_resource_struct, et qUnregisterResourceData().

```
{
    QT_PREPEND_NAMESPACE(qUnregisterResourceData)
       (0x01, qt_resource_struct, qt_resource_name, qt_resource_data);
    return 1;
}
```

10.16.1.2 QT_END_NAMESPACE int QT_MANGLE_NAMESPACE() qInitResources_ressources ( )

Références qRegisterResourceData(), qt_resource_data, qt_resource_name, et qt_resource_struct.

```
{
    QT_PREPEND_NAMESPACE(qRegisterResourceData)
       (0x01, qt_resource_struct, qt_resource_name, qt_resource_data);
    return 1;
}
```

10.16.1.3 QT_BEGIN_NAMESPACE Q_CORE_EXPORT bool qRegisterResourceData (int , const unsigned char * , const unsigned char * , const unsigned char *)

Référencé par qInitResources_ressources().

10.16.1.4 Q_CORE_EXPORT bool qUnregisterResourceData (int , const unsigned char * , const unsigned char * , const unsigned char *)

Référencé par qCleanupResources_ressources().

10.16.2 Documentation des variables

10.16.2.1 const unsigned char qt_resource_data[] [static]

Référencé par qCleanupResources_ressources(), et qInitResources_ressources().

10.16.2.2 const unsigned char qt_resource_name[] [static]

Référencé par qCleanupResources_ressources(), et qInitResources_ressources().

10.16.2.3 const unsigned char qt_resource_struct[] [static]

Référencé par qCleanupResources_ressources(), et qInitResources_ressources().

10.17 Référence du fichier qrc_ressources.cpp

```
#include <QtCore/qglobal.h>
```

Fonctions

- QT_BEGIN_NAMESPACE Q_CORE_EXPORT bool qRegisterResourceData (int, const unsigned char *, const unsigned char *, const unsigned char *)
- Q_CORE_EXPORT bool qUnregisterResourceData (int, const unsigned char *, const unsigned char *, const unsigned char *)
- QT_END_NAMESPACE int QT_MANGLE_NAMESPACE() qInitResources_ressources ()
- int QT_MANGLE_NAMESPACE() qCleanupResources_ressources ()

Variables

- static const unsigned char qt_resource_data []
- static const unsigned char qt_resource_name []
- static const unsigned char qt_resource_struct []

10.17.1 Documentation des fonctions

10.17.1.1 int QT_MANGLE_NAMESPACE() qCleanupResources_ressources ( )

Références qt_resource_data, qt_resource_name, qt_resource_struct, et qUnregisterResourceData().

```
{
    QT_PREPEND_NAMESPACE(qUnregisterResourceData)
       (0x01, qt_resource_struct, qt_resource_name, qt_resource_data);
    return 1;
}
```

10.17.1.2 QT_END_NAMESPACE int QT_MANGLE_NAMESPACE() qInitResources_ressources ( )

Références qRegisterResourceData(), qt_resource_data, qt_resource_name, et qt_resource_struct.

```
{
    QT_PREPEND_NAMESPACE(qRegisterResourceData)
        (0x01, qt_resource_struct, qt_resource_name, qt_resource_data);
    return 1;
}
```

10.17.1.3 QT_BEGIN_NAMESPACE Q_CORE_EXPORT bool qRegisterResourceData (int , const unsigned char * , const unsigned char * , const unsigned char *)

10.17.1.4 Q_CORE_EXPORT bool qUnregisterResourceData (int , const unsigned char * , const unsigned char * , const unsigned char *)

10.17.2 Documentation des variables

10.17.2.1 const unsigned char qt_resource_data[] [static]

Référencé par qCleanupResources_ressources(), et qInitResources_ressources().

10.17.2.2 const unsigned char qt_resource_name[] [static]

Valeur initiale :

```
{

  0x0,0x6,
  0x6,0xfa,0x64,0xc3,
  0x0,0x69,
  0x0,0x63,0x0,0x6f,0x0,0x6e,0x0,0x65,0x0,0x73,

  0x0,0x7,
  0x7,0x9a,0xbb,0x3c,
  0x0,0x61,
  0x0,0x63,0x0,0x63,0x0,0x75,0x0,0x65,0x0,0x69,0x0,0x6c,

  0x0,0x5,
  0x0,0x73,0xba,0xf1,
  0x0,0x6d,
  0x0,0x65,0x0,0x64,0x0,0x69,0x0,0x61,

  0x0,0xf,
  0x6,0x53,0x70,0x27,
  0x0,0x69,
  0x0,0x63,0x0,0x6f,0x0,0x6e,0x0,0x65,0x0,0x2d,0x0,0x76,0x0,0x69,0x0,0x64,0x0,
      0x65,0x0,0x6f,0x0,0x2e,0x0,0x70,0x0,0x6e,0x0,0x67,

  0x0,0x18,
  0xb,0x46,0x37,0xa7,
  0x0,0x69,
  0x0,0x63,0x0,0x6f,0x0,0x6e,0x0,0x65,0x0,0x2d,0x0,0x73,0x0,0x74,0x0,0x61,0x0,
      0x74,0x0,0x69,0x0,0x6f,0x0,0x6e,0x0,0x2d,0x0,0x64,0x0,0x65,0x0,0x2d,
  0x0,0x73,0x0,0x6b,0x0,0x69,0x0,0x2e,0x0,0x70,0x0,0x6e,0x0,0x67,

}
```

Référencé par qCleanupResources_ressources(), et qInitResources_ressources().

10.17.2.3 const unsigned char qt_resource_struct[] [static]

Valeur initiale :

```
{
  0x0,0x0,0x0,0x0,0x0,0x2,0x0,0x0,0x0,0x1,0x0,0x0,0x0,0x1,
  0x0,0x0,0x0,0x0,0x0,0x2,0x0,0x0,0x0,0x1,0x0,0x0,0x0,0x2,
  0x0,0x0,0x0,0x0,0x0,0x2,0x0,0x0,0x0,0x2,0x0,0x0,0x0,0x3,
  0x0,0x0,0x0,0x26,0x0,0x2,0x0,0x0,0x0,0x1,0x0,0x0,0x0,0x6,
  0x0,0x0,0x0,0x12,0x0,0x2,0x0,0x0,0x0,0x1,0x0,0x0,0x0,0x5,
  0x0,0x0,0x0,0x5a,0x0,0x0,0x0,0x0,0x0,0x1,0x0,0x0,0x1,0x76,
  0x0,0x0,0x0,0x36,0x0,0x0,0x0,0x0,0x0,0x1,0x0,0x0,0x0,0x0,
}
```

Référencé par qCleanupResources_ressources(), et qInitResources_ressources().

10.18 Référence du fichier README.dox

10.19 Référence du fichier stationmeteo.cpp

Définition de la classe StationMeteo.

```
#include "stationmeteo.h"     #include "basededonnees.h" ×
#include <qmath.h>
```

10.19.1 Description détaillée

Auteur

Petrella Olivier

Version

1.1

10.20 Référence du fichier stationmeteo.h

Déclaration de la classe StationMeteo.

```
#include <QVector>  #include <QString>  #include <QTime>
#include <QTimer> #include <QDateTime> #include <QDebug> ×
#include "structures.h" #include "wismasprotocole.h"
```

Classes

– class StationMeteo
Gére les données météos d'une station d'un site.

Macros

- #define SKI_ALPIN 1
- #define SKI_DE_FOND 2
- #define SNOW_SKITE 3
- #define DIRECTION_VENT 1
- #define VITESSE_VENT 2
- #define TEMPERATURE_AIR 3
- #define TEMPERATURE_NEIGE 4
- #define HAUTEUR_NEIGE 5
- #define HUMIDITE 6
- #define PRESSION_AIR 7

10.20.1 Description détaillée

Auteur

Petrella Olivier

Version

1.1

10.20.2 Documentation des macros

10.20.2.1 #define DIRECTION_VENT 1

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.20.2.2 #define HAUTEUR_NEIGE 5

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.20.2.3 #define HUMIDITE 6

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.20.2.4 #define PRESSION_AIR 7

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.20.2.5 #define SKI_ALPIN 1

Référencé par StationMeteo : :getNomTypeSite().

10.20.2.6 #define SKI_DE_FOND 2

Référencé par StationMeteo : :getNomTypeSite().

10.20.2.7 #define SNOW_SKITE 3

Référencé par StationMeteo : :getNomTypeSite().

10.20.2.8 #define TEMPERATURE_AIR 3

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.20.2.9 #define TEMPERATURE_NEIGE 4

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.20.2.10 #define VITESSE_VENT 2

Référencé par WISMASIHM : :afficherDonneesMeteoStation(), et StationMeteo : :getMesures().

10.21 Référence du fichier Structures.h

Regroupement de structures de données.

Classes

– struct Parametres
paramètres d'une caméra

10.21.1 Description détaillée

La structure Parametres.

Auteur

PETRELLA Olivier

Version

1.1

Date

21 mars 2018

Auteur

Pierre GRELET

Version

1.1

Date

15 mars 2018

10.22 Référence du fichier structures.h

```
#include <QString>
```

Classes

- struct CONFIG_LOGICIEL
  fixe la configuration de départ de l'application.
- struct CONFIG_ENREGISTREMENT
- struct PARAM_COMMUNICATION
- struct PARAM_PANNEAU
  fixe les paramètres du périphérique panneau lumineux. port QString, int baud_rate, int data_bit, int stop, int parity
- struct AFFICHAGE_PANNEAU
  fixe la configuration de l'affichage des informations sur le panneau lumineux. titre QString, int periode, bool temperature_air, bool temperature_neige, bool hauteur_neige, bool humidite, bool pression, bool vitesse_vent,
- struct DONNEES_STATION
  fixe les données d'un station

10.23 Référence du fichier Video.cpp

```
#include "Video.h"
```

10.23.1 Description détaillée

10.24 Référence du fichier Video.h

```
#include <QtCore>  #include <QApplication>  #include <Q-
Debug>
```

Classes

- class Video
  Traiter les vidéos capturer par le système vidéo.

Macros

- #define DEBUG

10.24.1 Description détaillée

10.24.2 Documentation des macros

10.24.2.1 #define DEBUG

10.25 Référence du fichier wismasihm.cpp

```
#include "wismasihm.h" #include <QDateTime> #include <-
QSettings> #include <QFile> #include <QUrl> #include <-
QDir> #include <QDirIterator> #include <QStringList> ×
#include <QList> #include <QDebug> #include "portxbee.h"
#include "portpanneau.h"
```

10.26 Référence du fichier wismasihm.h

Déclaration de la classe WISMASIHM.

```
#include <QWidget> #include <QTimer> #include <QString>
#include "ui_wismasihm.h" #include "structures.h" #include
"stationmeteo.h" #include <Phonon/MediaObject> #include
<Phonon/VideoWidget> #include <Phonon/VideoPlayer>
```

Classes

– class WISMASIHM
Gère l'interface des sites de haute montagne.

10.26.1 Description détaillée

Auteur

PETRELLA Olivier

Version

0.9

Date

18 février 2018

10.27 Référence du fichier wismasprotocole.cpp

Définition de la classe WISMASProtocole.

```
#include "wismasprotocole.h" #include "stationmeteo.h" ×
#include <QStringList> #include <QDate> #include <Q-
Time> #include <QDebug>
```

10.27.1 Description détaillée

Auteur

Petrella Olivier

Version

1.1

10.28 Référence du fichier wismasprotocole.h

Déclaration de la classe WISMASProtocole.

```
#include <QString>
```

Classes

– class WISMASProtocole
Gère la transmission entre les sites.

10.28.1 Description détaillée

Auteur

PETRELLA Olivier

Version

0.9

Date

18 février 2018

Index