WISMAS

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

ZANCA Kilian (IR)
FOULARD Jimmy (IR)
BLAIRON Jeanlin (EC)
OPPERMANN Balthazar (EC)

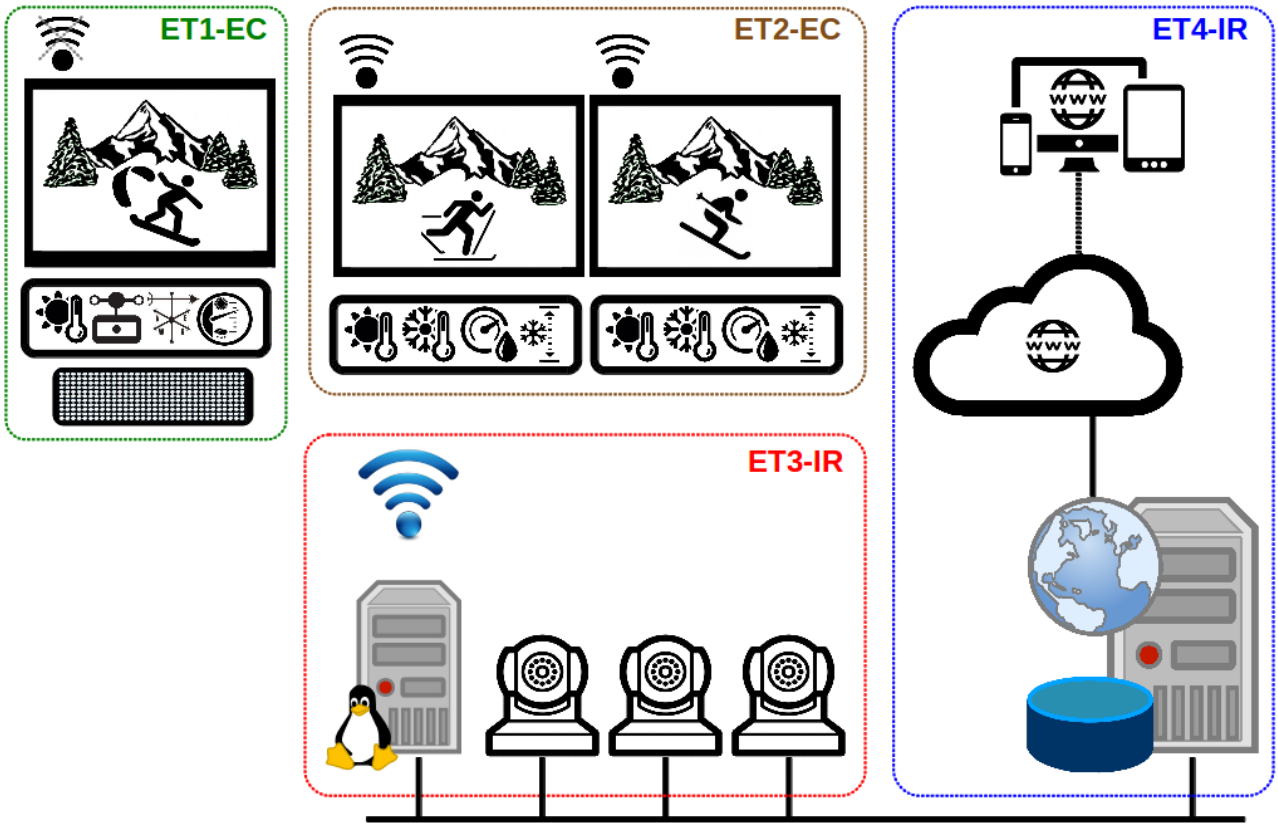
Table des matières

Présentation

Le système d'information météo pour station multi activités **WISMAS** (*Weather Information System Multi Activity Station*) devra :

- Faire des mesures météorologiques sur plusieurs sites.
- **Prendre des séquences vidéo sur plusieurs camera IP**.
- Afficher l'ensemble de ces renseignements sur les sites d'achat des forfaits.
- Renseigner une base de données.
- Gérer un site internet avec accès en consultation et abonné.

La station sur laquelle s'appuie l'étude est une station de moyenne montagne multi-activités dans laquelle on peut faire du ski de fond, du ski de piste et également du Snow-Kite sur 3 sites voisins.



L'objectif est de moderniser l'ensemble en y ajoutant des fonctionnalités.

Il s'agit :

- d'une part de développer sur carte ATMEL SAM4S-EK les applications nécessaires à la mise en œuvre des différents capteurs et à la transmission sans fil des données vers le PC de gestion.
- d'autre part la gestion des panneaux lumineux et des écrans de diffusion d'images.
- et enfin la réception et traitement des données, la gestion de la base de données et la gestion du site (accès et mise à jour des données).

Le système sera construit à partir de carte(s) micro-contrôleur qui seront mises en place sur les sites de mesures et d'un poste PC de contrôle.

L'application logicielle devra assurer :

- **D'acquérir les images vidéos des caméras IP installées sur les différents sites**
- **D'enregistrer les vidéos sur le serveur pour une diffusion ultérieure**
- **De commander les déplacements d'une caméra pour une acquisition panoramique**
- De relever les mesures des stations météorologiques installées sur les différents sites
- D'enregistrer les mesures dans une base données installée sur le serveur

L'administrateur pourra paramétrer :

- **Les adresses IP et numéros de port des différentes caméras**
- **La position d'origine de la prise de vue en pouvant déplacer la caméra manuellement**
- **Le type (fixe ou panoramique) et la durée d'acquisition vidéo**

Enfin, le site web sera hébergé sur le serveur et permettra aux clients d'accéder aux pages suivantes :

- Informations sur la station (accès, horaires, tarifs, …)
- Conditions météorologiques suivant le site (Alpin, Fond ou Snow kite) avec la possibilité de visualiser une vidéo du lieu
- Conseils de fartage en fonction des conditions météorologiques

Analyse de l'existant

Le système existant sur le site est un ensemble de « petits » systèmes non reliés :

- Un panneau lumineux affichant température et date sur les sites Alpin et Fond (non inclus dans le futur système Wismas).
- Des caméras sur les sites Fond et Alpin.
- Un site web permettant de connaitre les pistes ouvertes, la météo du site (prévision) et de visualiser en temps réel l'image des caméras.

Énoncé des tâches à réaliser par les étudiants

Étudiant 1 (EC)

Fonctions :

- Mesurer la température
- Mesurer l'hygrométrie
- Mesurer la vitesse et le sens du vent
- Mettre en forme les mesures
- Gérer l'affichage sur panneau lumineux

Étudiant 2 (EC)

Fonctions :

- Mesurer la température de l'air et de la neige
- Mesurer l'hygrométrie
- Mesurer la hauteur de neige
- Mettre en forme les mesures
- Transmettre ces mesures via une liaison sans fil

Étudiant 3 (IR) : ZANCA Killian

Cas d'utilisation :

- Acquérir vidéo
- Enregistrer (vidéo et mesures)
- Déplacer caméra
- Paramétrer le système
- Relever les mesures des sites

Installation :

- Les caméras

Mise en oeuvre :

- Les caméras, la liaison sans fil
- L'environnement de développement

Configuration :

- Les caméras, la liaison sans fil

Réalisation :

- Les diagrammes SysML (diagramme de définition de blocs et diagramme interne de bloc de son module) et diagrammes UML
- L'IHM et les classes du module

Documentation :

- Le dossier technique et les documents relatifs au module,
- Un guide de mise en route et d'utilisation du module

Étudiant 4 (IR)

Cas d'utilisation :

- Consulter les informations sur la station
- Consulter les conditions météorologiques d'un site
- Visualiser les vidéos

- Obtenir des conseils de fartage

Installation :

- Le serveur (OS, MySQL, Apache, …)

Mise en oeuvre :

- Le serveur (OS, MySQL, Apache, …),
- L'environnement de développement

Configuration :

- Le serveur (OS, MySQL, Apache, …)

Réalisation :

- Les diagrammes SysML (diagramme de définition de blocs et diagramme interne de bloc de son module) et diagrammes UML
- L'IHM et les classes du module

Documentation :

- Le dossier technique et les documents relatifs au module,
- Un guide de mise en route et d'utilisation du module

Les ressources matérielles

Capteurs de température, d'hygrométrie, de vitesse et direction du vent

Capteur à ultrasons

Caméras IP PTZ

Journal lumineux CONRAD

Carte ATMEL SAM4S-EK

Les ressources logicielles

Ressource	Version
OS	GNU Linux (Ubuntu 12.04.5 LTS)
SGBDR	MySQL
Serveur web	Apache
EDI	Qt Creator 2.4.1
Compilateur	GNU g++/gcc version 4.6.3
Débugueur	GNU gdb 7.4
Fabrication	QMake 2.01a et GNU make 3.81
API GUI	Qt 4.8.1
Langage (Web)	HTML5/CSS3/PHP5
API Web	Bootstrap, jQuery Mobile, Smarty et TinyMVC
UML	bouml 4.23
Tests	CppUnit 1.12.1, PHPUnit
Versions	subversion (client svn 1.6.17)
Documentation	Doxygen version 1.7.6.1 et pandoc 1.9.1.1
Gantt	Planner (version 0.14.5) ou gantter

Partie Personnelle – ZANCA Killian (IR1)

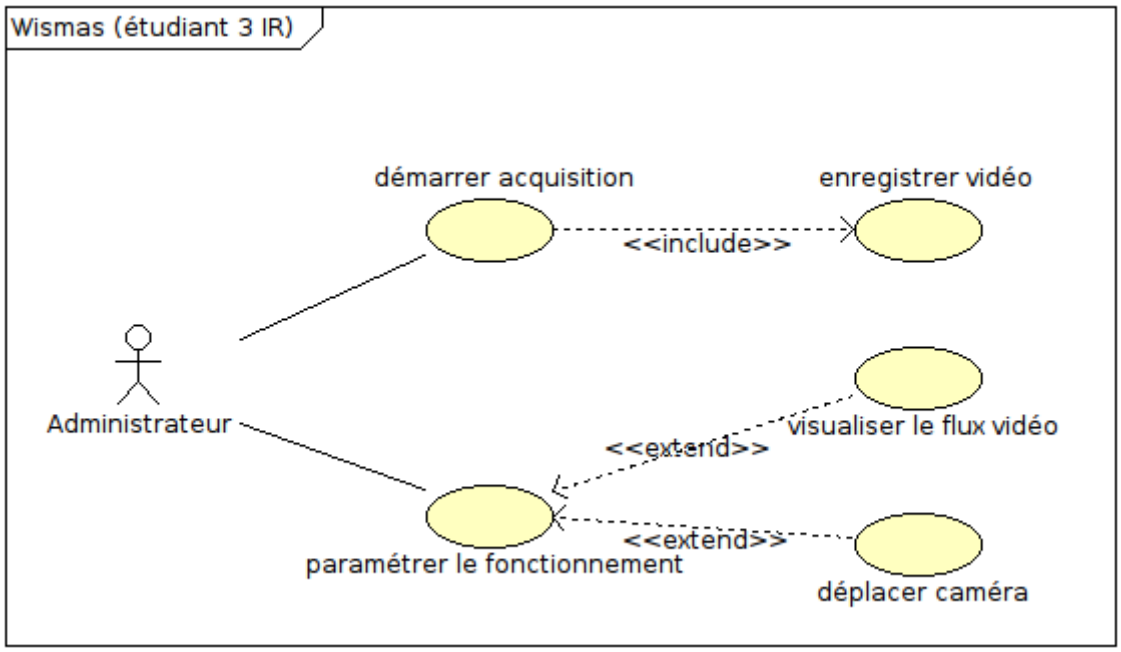
Je suis l'étudiant 3 (IR1) dans le projet WISMAS.

J'ai réalisé les cas d'utilisation :

- Démarrer l'acquisition vidéo
- Enregistrer vidéo
- Paramétrer le fonctionnement
- Déplacer caméra
- Visualiser le flux vidéo

Diagramme de cas d'utilisation

Le diagramme des cas d'utilisation de l'application « Acquisition » est le suivant :



L'administrateur pourra paramétrer le fonctionnement du système d'Acquisition, notamment la durée et la période en secondes, la résolution et le nombre d'images/seconde de chaque caméra (ces paramètres seront stockés dans un fichier .ini).

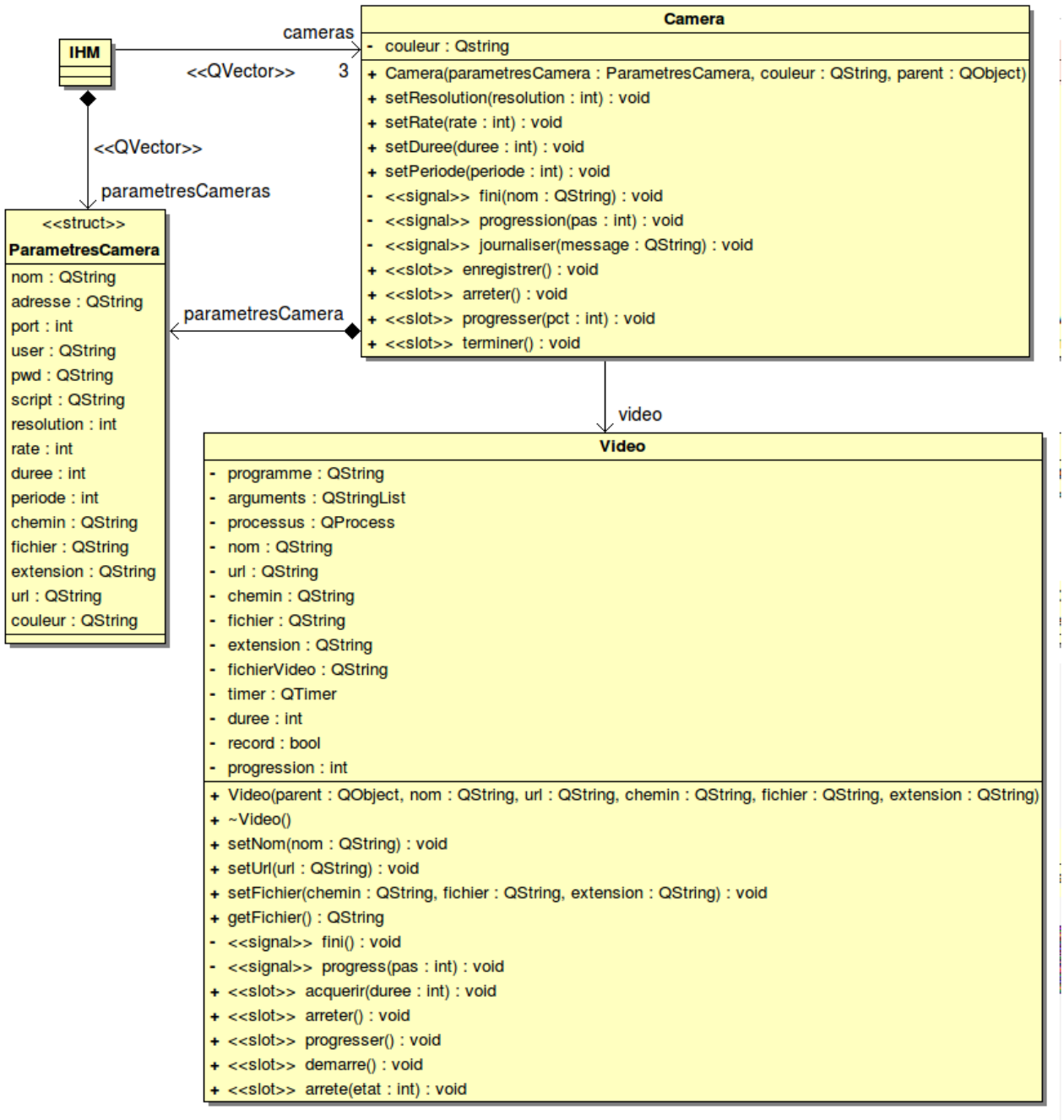
Il pourra modifier l'orientation des caméras avec les boutons de l'IHM qui vont commander les déplacements d'une caméra pour une acquisition panoramique.

Il démarrera l'acquisition des vidéos ce qui provoquera leurs enregistrements en local dans un dossier qui sera partagé sur le réseau via le protocole NFS dont son emplacement sera préalablement connu de l'application (fichier .ini) et enregistré dans la base de données qui se situe sur le serveur du site web.

L'application "Acquisition" pourra également relever les mesures des sites envoyé à partir des différents capteurs via une carte Xbee en wi-fi qui seront enregistré dans la base de données.

La base de données contiendra donc la valeur des mesures des différents capteurs ainsi que l'emplacement des vidéos afin d'être récupérées par le serveur web et ainsi afficher les données sur la page du site concerné qui affichera la vidéo enregistrée et les données des capteurs.

Diagramme de classes



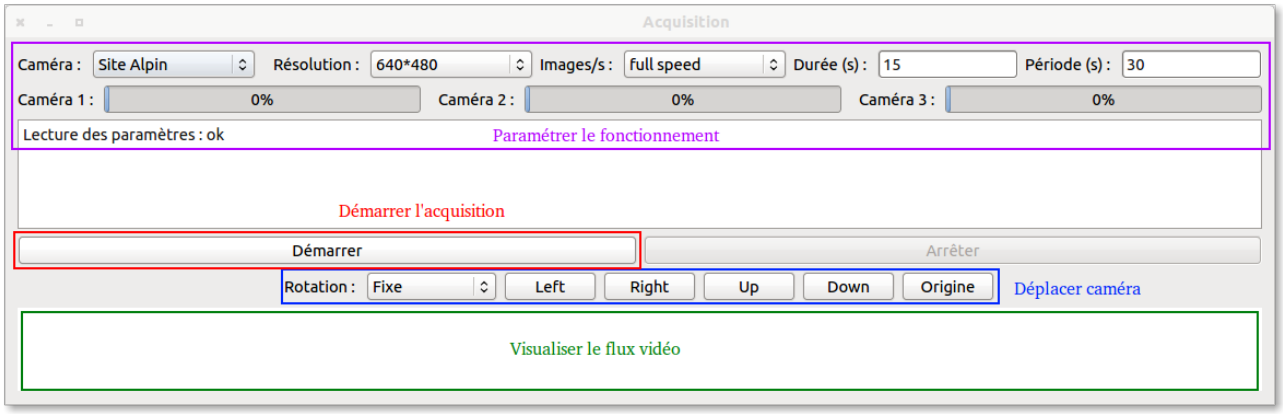
La classe **IHM** gérera 3 objets Camera stockés dans un QVector ainsi que leurs paramètres dont les principaux seront la résolution, le nombre d'images/seconde, la durée et la période en seconde.

La classe **Camera** gère l'acquisition vidéo.

La classe **Video** assure l'acquisition du flux vidéo et son enregistrement dans un fichier.

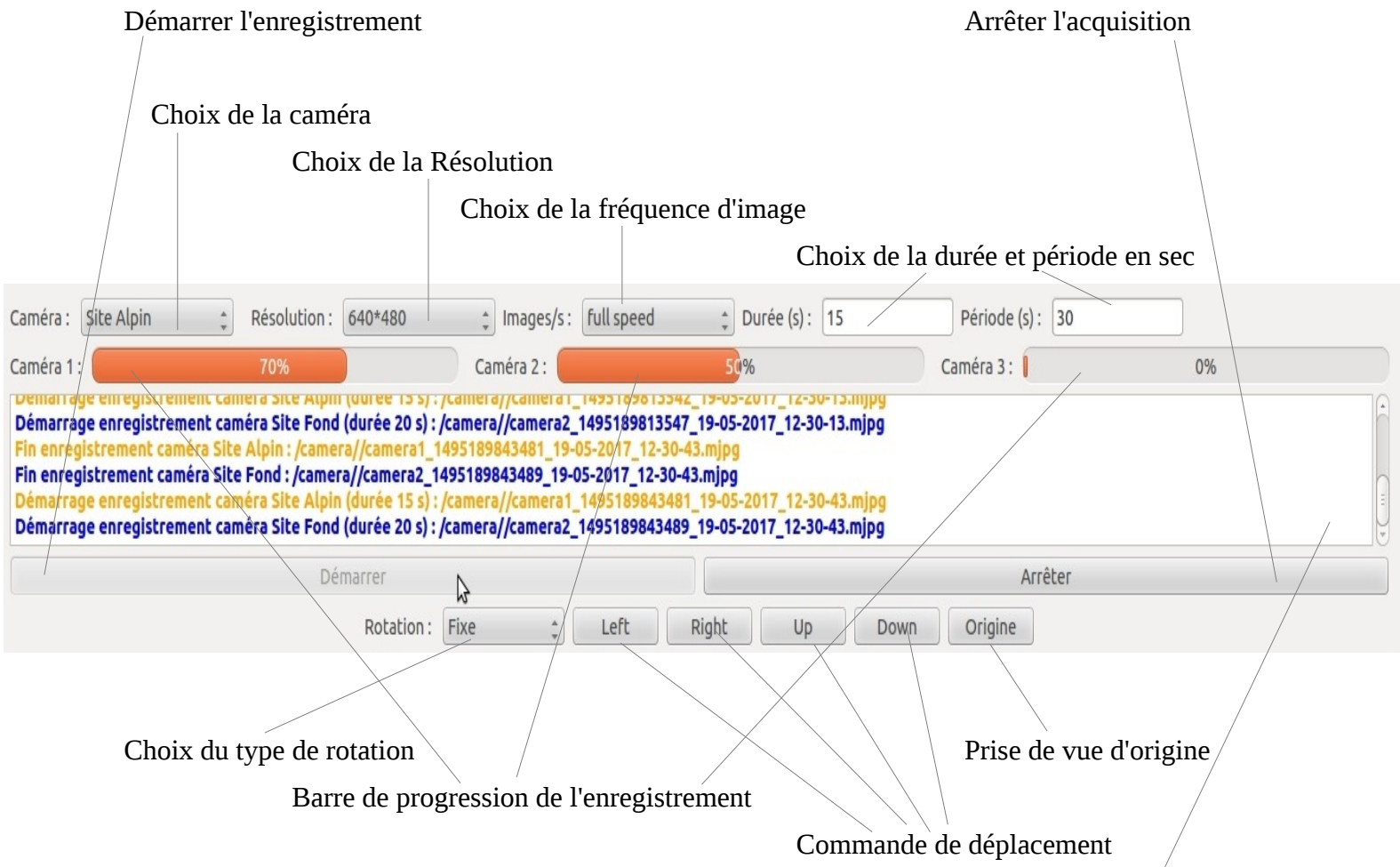
La structure **ParametresCamera** définira les paramètres des caméras renseignés en lien avec un fichier .ini.

La classe IHM définira l'interface utilisateur graphique (homme-machine) qui sera utilisée par l'administrateur.



On pourra choisir la caméra via une liste déroulante en fonction du site (site alpin, fond ou Snow-Kite) et paramétrer la résolution (640x480, 320x240), le nombre d'images/seconde (full speed = 25 fps, 20 fps, 15 fps...), la durée et la période en seconde puis démarrer ou arrêter l'acquisition. Ensuite on pourra manuellement (rotation fixe) commander les déplacements de la caméra sélectionné ou automatiquement via les positions pré-fixé (rotation panoramique).

Caméra :
Site Alpin
Résolution :
640*480
Images/s :
full speed
Durée (s) :
15
Période (s) :
30
Caméra 1 :
70%
Caméra 2 :
50%
Caméra 3 :
0%
Démarrage enregistrement caméra Site Alpin (durée 15 s) : /camera//camera1_1495189843481_19-05-2017_12-30-13.mjpg
Démarrage enregistrement caméra Site Fond (durée 20 s) : /camera//camera2_1495189813547_19-05-2017_12-30-13.mjpg
Fin enregistrement caméra Site Alpin : /camera//camera1_1495189843481_19-05-2017_12-30-43.mjpg
Fin enregistrement caméra Site Fond : /camera//camera2_1495189843489_19-05-2017_12-30-43.mjpg
Démarrage enregistrement caméra Site Alpin (durée 15 s) : /camera//camera1_1495189843481_19-05-2017_12-30-43.mjpg
Démarrage enregistrement caméra Site Fond (durée 20 s) : /camera//camera2_1495189843489_19-05-2017_12-30-43.mjpg
Démarrer
Arrêter
Rotation :
Fixe
Left
Right
Up
Down
Origine



Message concernant l'acquisition (le démarrage/arrêt de l'acquisition et de l'enregistrement avec les chemins la durée et la période d'acquisition) renvoie également ok lorsque la commande de déplacement de la caméra a été correctement transmise.

Choix des caméras

Comparaison caméra intérieure (utilisé actuellement pour les tests) et extérieur (cahier des charges) :

idCamera	Marque	Extérieur	Motorisé	Angle	Réseaux	Wi-fi	Étanche
1	Wanscam JW0008	non	oui	355°/90°	RJ45/Wi-Fi	802.11 b/g/n	non
2	Sécurité Mania	oui	oui	355°/90°	RJ45/Wi-Fi	802.11 b/g/n	oui
3	Sricam SP015	oui	oui	355°/90°	RJ45/Wi-Fi	802.11 b/g/n	oui
idCamera	Infrarouge	Résolution	Compression	fréquence img max	Prix Unitaire	Quantité	Prix TTC
1	oui	480p	MJPEG	25	~45€	x1	~ 45,00 €
2	oui	720p	H.264	25	129,95 €	x2	259,90 €
3	oui	720p	H.264	30	89,99 €	x1	89,99 €

La caméra n°3 est la plus avantageuse car elle permet une fréquence d'images plus grande et est moins onéreuse, c'est la moins chère des caméras trouvé sur internet ayant les mêmes

caractéristiques communes avec celle d'intérieur mais en extérieur.

Partage des vidéos (NFS)

Les vidéos enregistrées doivent être partagées avec le site web pour permettre leur visonnage. Le protocole **NFS** (*Network File System*) est un système de fichier réseau permettant de partager le dossier sur le serveur à un client définit dans le fichier **`/etc/export`** ci-dessous et monté sur le client.

Contenu du fichier **`/etc/exports`** :

```
 # /etc/exports: the access control list for filesystems which may be exported
 #        to NFS clients.  See exports(5).
 #
 # Example for NFSv2 and NFSv3:
 # /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
 #
 # Example for NFSv4:
 # /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
 # /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
 #
 # Dossier Partagé
 /camera 192.168.52.134(rw)
```

Cas d'utilisation « Paramétrer le système »

L'administrateur pourra paramétrer :

- Les adresses IP et numéros de port des différentes caméras
- La position d’origine de la prise de vue
- Le type (fixe ou panoramique) et la durée d’acquisition vidéo en seconde

Les paramètres de configuration seront stockés dans un fichier .INI (exemple pour la caméra 1) :

```
 [camera1]
 nom=Site Alpin → Le nom afin d'identifier facilement le lieu
 adresse=192.168.52.93 → Son adresse IP
 port=99 → Son port
 user=admin → Son identifiant associé pour s'authentifier
 pwd= → Son mot de passe, vide par défaut
 script=mobile.htm → Le script associé au type d'appareil utilisé
```

```
pour y accéder
  resolution=32 → Sa résolution (32=640x480)
  rate=0 → Le nombre d'images/seconde (0=full speed=25 ips)
  position=30 → La position d'origine paramétré à la position 1 (setPosition 1)
  type=0 → Le type de rotation (0=fixe;1=panoramique)
  duree=15000 → La durée de la vidéo en millisecondes (converti automatiquement dans l'IHM en seconde)
  periode=30 → La période d'attente entre 2 enregistrements en seconde
  fichier=camera1 → Le préfixe du nom du fichier d'enregistrement
  url="http://192.168.52.93:99/videostream.cgi?user=admin&pwd=" → adresse du flux vidéo
  chemin=/camera/ → chemin du fichier enregistré
  extension=mjpg → L'extension utilisé en fonction du format de fichier pris en charge
  Le Motion JPEG ou M-JPEG est un codec vidéo qui compresse les images une à une en JPEG qui est un format d'image (image en continu)
  couleur=orange → La couleur permet de différencier la caméra qui sera enregistrer via les messages diffusé dans le journal.
```

Cas d'utilisation « Démarrer acquisition »

Acquérir vidéo → acquérir les images vidéos des caméras IP installées sur les différents sites et donc permettre leur affichage dans l'IHM

Cas d'utilisation « Enregistrer vidéo »

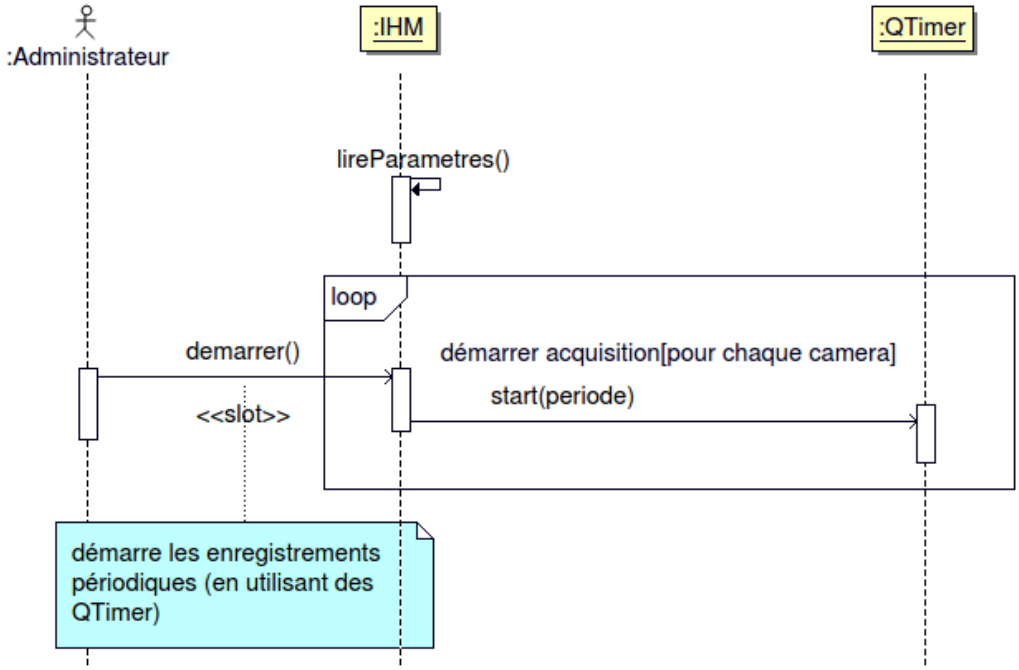
Enregistrer (vidéo et mesure) → Enregistrer les vidéos sur le serveur (local) pour une diffusion ultérieure sur le site Permettre l'enregistrement en local de la vidéo (emplacement réseau) et de relever les mesures des stations météorologiques installées sur les différents sites

L'enregistrement des caméras se fait via le programme **CVLC** (*Command VideoLAN Client*), un client VLC sans interface (se lance en tâche de fond), qui enregistre les vidéos dans un répertoire en local qui sera partagé en utilisant le protocole NFS (*Network File System*).

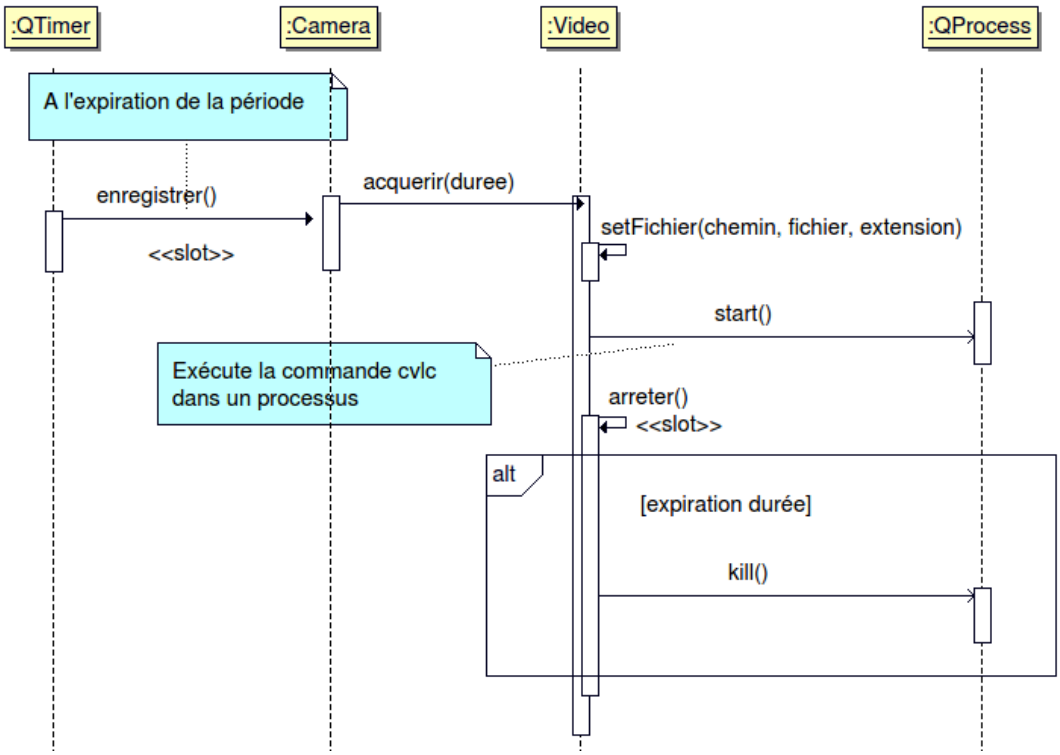
Cas d'utilisation « Déplacer caméra »

Déplacer caméra → Commander les déplacements d'une caméra depuis l'IHM pour une acquisition panoramique ou fixe.

Diagramme de séquences

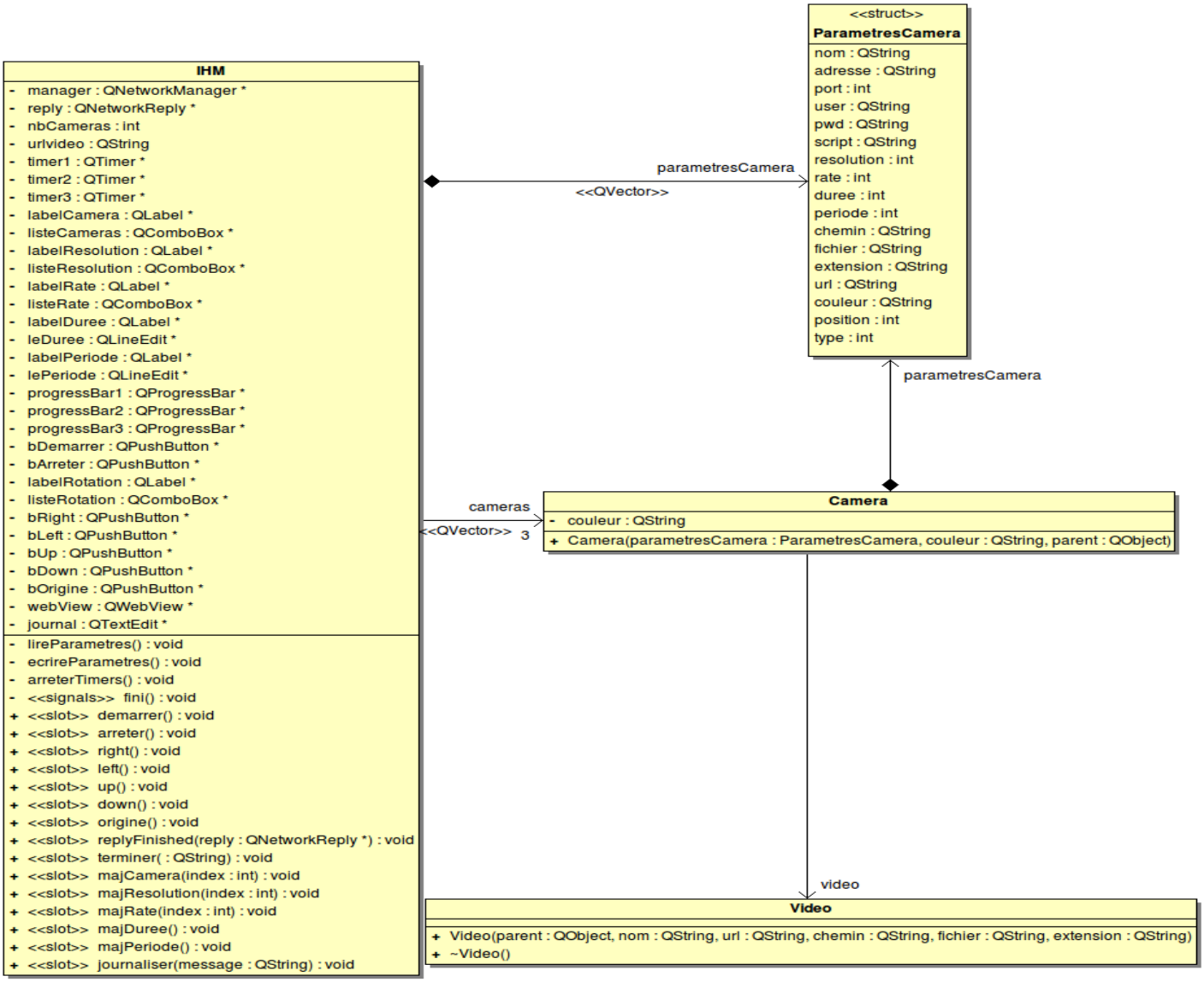


Ce diagramme décrit le cas d'utilisation « démarrer acquisition », la classe IHM lit les paramètres des caméras dans le fichier .ini au lancement puis l'Administrateur va cliquer sur le bouton démarrer ensuite l'IHM va démarrer la période des Qtimer pour chaque caméra.



Ce diagramme décrit le cas d’utilisation « enregistrer video », à l ’expiration des périodes des 3 Qtimers, la classe Camera va enregistrer les flux vidéos, la classe Video va acquérir en fonction de la durée les vidéos, elle va créer les fichiers vidéo et va exécuter la commande cvlc dans un processus et ferme le programme cvlc à l’expiration de la durée.

Annexe 1 : diagramme de classes



Annexe 2 : déclaration de la classe IHM : Fichier ihm.h

```
#ifndef _IHM_H
#define _IHM_H

#include <QtGui>
#include <QWebView>
#include <QWebFrame>
#include <QNetworkAccessManager>
#include <QNetworkReply>
#include "camera.h"

class IHM : public QWidget
{
    Q_OBJECT

public:
    IHM(QWidget *parent = 0);
    ~IHM();

private:
    QNetworkAccessManager *manager;
    QNetworkReply         *reply;

    QVector<Camera*> cameras;
    QVector<ParametresCamera> parametresCameras;
    int              nbCameras;
    QString          urlvideo;
    QTimer           *timer1;
    QTimer           *timer2;
    QTimer           *timer3;

    // Widgets
    QLabel           *labelCamera;
    QComboBox        *listeCameras;
    QLabel           *labelResolution;
    QComboBox        *listeResolution;
    QLabel           *labelRate;
    QComboBox        *listeRate;
    //QLabel           *labelPosition;
    //QComboBox        *listePosition;
    QLabel           *labelDuree;
    QLineEdit        *leDuree;
    QLabel           *labelPeriode;
    QLineEdit        *lePeriode;
    QProgressBar     *progressBar1;
    QProgressBar     *progressBar2;
    QProgressBar     *progressBar3;
```

```
    QPushButton     *bDemarrer;
    QPushButton     *bArreter;
    QLabel          *labelRotation;
    QComboBox       *listeRotation;
    QPushButton     *bRight;
    QPushButton     *bLeft;
    QPushButton     *bUp;
    QPushButton     *bDown;
    QPushButton     *bOrigine;
    QWebView        *webView;
    QTextEdit       *journal;

    void            lireParametres();
    void            ecrireParametres();
    void            arreterTimers();

signals:
    void            fini();

public slots:
    void            demarrer();
    void            arreter();
    void            right();
    void            left();
    void            up();
    void            down();
    void            origine();
    void            replyFinished(QNetworkReply *reply);
    void            terminer(QString);
    void            majCamera(int index);
    void            majResolution(int index);
    void            majRate(int index);
    void            majDuree();
    void            majPeriode();
    void            journaliser(QString message);
};

#endif // IHM_H
```

Annexe 3 : définition de la classe IHM : Fichier ihm.cpp

```
#include "video.h"

#include "ihm.h"
#include "debug.h"

IHM::IHM( QWidget *parent ) : QWidget( parent )
{
    /* les timers pour les enregistrements périodiques */
    timer1 = new QTimer(this);
    timer2 = new QTimer(this);
    timer3 = new QTimer(this);

    /* les widgets */
    labelCamera = new QLabel(this); /* crée dynamiquement le label */
    labelCamera->setText(QString::fromUtf8("Caméra : ")); /* Défini le texte du
label */
    listeCameras = new QComboBox(this); /* crée dynamiquement la liste
déroulante */
    listeCameras->setFixedWidth(150); /* Fixe la largeur */

    labelResolution = new QLabel(this);
    labelResolution->setText(QString::fromUtf8("Résolution : "));
    listeResolution = new QComboBox(this);
    listeResolution->setFixedWidth(150);
    listeResolution->addItem(QString::fromUtf8("640*480")); /* 32 */
    listeResolution->addItem(QString::fromUtf8("320*240")); /* 8 */

    /** Fréquence d'image */
    labelRate = new QLabel(this);
    labelRate->setText(QString::fromUtf8("Images/s : "));
    listeRate = new QComboBox(this);
    listeRate->setFixedWidth(150);
    listeRate->addItem(QString::fromUtf8("full speed")); /* 0 */
    listeRate->addItem(QString::fromUtf8("20 fps")); /* 1 */
    listeRate->addItem(QString::fromUtf8("15 fps")); /* 3 */
    listeRate->addItem(QString::fromUtf8("10 fps")); /* 6 */
    listeRate->addItem(QString::fromUtf8("5 fps")); /* 11 */
    listeRate->addItem(QString::fromUtf8("4 fps")); /* 12 */
    listeRate->addItem(QString::fromUtf8("3 fps")); /* 13 */
    listeRate->addItem(QString::fromUtf8("2 fps")); /* 14 */
    listeRate->addItem(QString::fromUtf8("1 fps")); /* 15 */

    labelDuree = new QLabel(this);
    labelDuree->setText(QString::fromUtf8("Durée (s) : "));
    leDuree = new QLineEdit(this); /* crée dynamiquement un éditeur de texte */
```

```
leDuree->setEnabled(true); /* Active par défaut le champ */
leDuree->setFixedWidth(leDuree->sizeHint().width());

labelPeriode = new QLabel(this);
labelPeriode->setText(QString::fromUtf8("Période (s) : "));
lePeriode = new QLineEdit(this);
lePeriode->setEnabled(true);
lePeriode->setFixedWidth(lePeriode->sizeHint().width());

/* Crée les labels et barre de progression de l'enregistrement */
QLabel *labelCamera1 = new QLabel(QString::fromUtf8("Caméra 1 : "), this);
progressBar1 = new QProgressBar(this);
progressBar1->setValue(0); /* Indique la valeur par défaut */
QLabel *labelCamera2 = new QLabel(QString::fromUtf8("Caméra 2 : "), this);
progressBar2 = new QProgressBar(this);
progressBar2->setValue(0);
QLabel *labelCamera3 = new QLabel(QString::fromUtf8("Caméra 3 : "), this);
progressBar3 = new QProgressBar(this);
progressBar3->setValue(0);

/* Crée les bouton Démarrer et Arrêter l'enregistrement */
bDemarrer = new QPushButton(QString::fromUtf8("Démarrer"), this);
bDemarrer->setEnabled(true);
bArreter = new QPushButton(QString::fromUtf8("Arrêter"), this);
bArreter->setEnabled(false);

journal = new QTextEdit(this);
journal->setReadOnly(true); /* On ne modifiera pas le contenu du journal */
journal->setFixedHeight(100); /* Fixe la hauteur */

/* liste pour le type de rotation fixe ou panoramique */
labelRotation = new QLabel(this);
labelRotation->setText(QString::fromUtf8("Rotation : "));
listeRotation = new QComboBox(this);
//listeRotation->setFixedWidth(150);
listeRotation->addItem(QString::fromUtf8("Fixe"));
listeRotation->addItem(QString::fromUtf8("Panoramique"));

/* crée les bouttons de déplacement et les désactive par défaut */
bRight = new QPushButton(QString::fromUtf8("Right"), this);
bRight->setEnabled(false);
bLeft = new QPushButton(QString::fromUtf8("Left"), this);
bLeft->setEnabled(false);
bUp = new QPushButton(QString::fromUtf8("Up"), this);
bUp->setEnabled(false);
bDown = new QPushButton(QString::fromUtf8("Down"), this);
bDown->setEnabled(false);
```

```
    bOrigine = new QPushButton(QString::fromUtf8("Origine"), this);
    bOrigine->setEnabled(false);

    /* Crée l'affichage du flux video */
    webView = new QWebView(this);
    webView->setGeometry(0, 0, 320, 200);
    /* Retrait des barres de défilement */
    webView->page()->mainFrame()->setScrollBarPolicy(Qt::Horizontal,
Qt::ScrollBarAlwaysOff);
    webView->page()->mainFrame()->setScrollBarPolicy(Qt::Vertical,
Qt::ScrollBarAlwaysOff);

    /* positionnement des widgets (mise en place des layouts) */
    QHBoxLayout *hLayout0 = new QHBoxLayout;
    QHBoxLayout *hLayout1 = new QHBoxLayout;
    QHBoxLayout *hLayout2 = new QHBoxLayout;
    QHBoxLayout *hLayout3 = new QHBoxLayout;
    QHBoxLayout *hLayout4 = new QHBoxLayout;
    QHBoxLayout *hLayout5 = new QHBoxLayout;
    QVBoxLayout *mainLayout = new QVBoxLayout;

    hLayout0->addWidget(labelCamera);
    hLayout0->addWidget(listeCameras);
    hLayout0->addSpacing(10);
    hLayout0->addWidget(labelResolution);
    hLayout0->addWidget(listeResolution);
    hLayout0->addWidget(labelRate);
    hLayout0->addWidget(listeRate);
    hLayout0->addWidget(labelDuree);
    hLayout0->addWidget(leDuree);
    hLayout0->addWidget(labelPeriode);
    hLayout0->addWidget(lePeriode);
    hLayout0->addStretch(1);
    hLayout1->addWidget(labelCamera1);
    progressBar1->setAlignment(Qt::AlignHCenter);
    hLayout1->addWidget(progressBar1);
    hLayout1->addSpacing(10);
    hLayout1->addWidget(labelCamera2);
    progressBar2->setAlignment(Qt::AlignHCenter);
    hLayout1->addWidget(progressBar2);
    hLayout1->addSpacing(10);
    hLayout1->addWidget(labelCamera3);
    progressBar3->setAlignment(Qt::AlignHCenter);
    hLayout1->addWidget(progressBar3);
    hLayout1->setAlignment(Qt::AlignHCenter);
    hLayout2->addWidget(bDemarrer);
    hLayout2->addWidget(bArreter);
```

```
    hLayout3->addWidget(journal);
    hLayout4->setAlignment(Qt::AlignHCenter);
    hLayout4->addWidget(labelRotation);
    hLayout4->addWidget(listeRotation);
    hLayout4->addWidget(bLeft);
    hLayout4->addWidget(bRight);
    hLayout4->addWidget(bUp);
    hLayout4->addWidget(bDown);
    hLayout4->addWidget(bOrigine);
    hLayout5->addWidget(webView);
    mainLayout->addLayout(hLayout0);
    mainLayout->addLayout(hLayout1);
    mainLayout->addLayout(hLayout3);
    mainLayout->addLayout(hLayout2);
    mainLayout->addLayout(hLayout4);
    mainLayout->addLayout(hLayout5);
    mainLayout->addStretch(1);
    setLayout(mainLayout);

    /* initialisation */
    nbCameras = 3;
    lireParametres();

    /* les signaux/slots */
    connect(listeCameras, SIGNAL(currentIndexChanged(int)), this,
SLOT(majCamera(int)));
    connect(leDuree, SIGNAL(editingFinished()), this, SLOT(majDuree()));
    connect(lePeriode, SIGNAL(editingFinished()), this, SLOT(majPeriode()));
    connect(listeResolution, SIGNAL(currentIndexChanged(int)), this,
SLOT(majResolution(int)));
    connect(listeRate, SIGNAL(currentIndexChanged(int)), this,
SLOT(majRate(int)));
    connect(bDemarrer, SIGNAL(clicked()), this, SLOT(demarrer()));
    connect(bArreter, SIGNAL(clicked()), this, SLOT(arreter()));
    connect(bRight, SIGNAL(clicked()), this, SLOT(right()));
    connect(bLeft, SIGNAL(clicked()), this, SLOT(left()));
    connect(bUp, SIGNAL(clicked()), this, SLOT(up()));
    connect(bDown, SIGNAL(clicked()), this, SLOT(down()));
    connect(bOrigine, SIGNAL(clicked()), this, SLOT(origine()));
    connect(webView, SIGNAL(urlChanged(QUrl)), this, SLOT(webview_changed(const
QUrl &)));
    majCamera(listeCameras->currentIndex());

    /* Envoyer des requêtes de commande déplacement des caméras */
    manager = new QNetworkAccessManager(this);
    connect(manager, SIGNAL(finished(QNetworkReply*)), this,
SLOT(replyFinished(QNetworkReply*)));
```

```
    bRight->setEnabled(true);
    bLeft->setEnabled(true);
    bUp->setEnabled(true);
    bDown->setEnabled(true);
    bOrigine->setEnabled(true);

    /* affichage */
    QDesktopWidget *widget = QApplication::desktop();
    resize(widget->width(), widget->height());
}

IHM::~IHM()
{
    arreter();
    ecrireParametres();
}

void IHM::demarrer() /** Bouton Démarrer */
{
    progressBar1->setValue(0);
    progressBar2->setValue(0);
    progressBar3->setValue(0);
    bDemarrer->setEnabled(false);
    bArreter->setEnabled(true);

    arreterTimers();

    /* démarre les enregistrements périodiques */
    journaliser(QString::fromUtf8("Programmation enregistrement caméra %1 :
toutes les %2
s").arg(parametresCameras.at(0).nom).arg(parametresCameras.at(0).periode));
    timer1->start(parametresCameras.at(0).periode*1000);
    journaliser(QString::fromUtf8("Programmation enregistrement caméra %1 :
toutes les %2
s").arg(parametresCameras.at(1).nom).arg(parametresCameras.at(1).periode));
    timer2->start(parametresCameras.at(1).periode*1000);
    journaliser(QString::fromUtf8("Programmation enregistrement caméra %1 :
toutes les %2
s").arg(parametresCameras.at(2).nom).arg(parametresCameras.at(2).periode));
    timer3->start(parametresCameras.at(2).periode*1000);
}

void IHM::arreter() /** Bouton Arrêter */
{
    arreterTimers();

    bDemarrer->setEnabled(true);
```

```
    bArreter->setEnabled(false);

    emit fini();
    journaliser(QString::fromUtf8("Fin de la programmation des enregistrements
caméra %1").arg(parametresCameras.at(0).nom));
    journaliser(QString::fromUtf8("Fin de la programmation des enregistrements
caméra %1").arg(parametresCameras.at(1).nom));
    journaliser(QString::fromUtf8("Fin de la programmation des enregistrements
caméra %1").arg(parametresCameras.at(2).nom));
}

void IHM::arreterTimers()
{
    if(timer1->isActive())
        timer1->stop();
    if(timer2->isActive())
        timer2->stop();
    if(timer3->isActive())
        timer3->stop();
}

void IHM::terminer(QString nomCamera)
{
    Q_UNUSED(nomCamera)
    qDebug() << Q_FUNC_INFO << nomCamera;
}

void IHM::lireParametres() /** Paramètres de configuration des 3 caméras dans le
fichier .ini */
{
    QString fichierINI = qApp->applicationDirPath() + "/" + qApp-
>applicationName() + ".ini";
    QSettings parametres(fichierINI, QSettings::IniFormat);
    QString id;
    int choixResolution[2] = { 32, 8 };
    int choixRate[9] = { 0, 1, 3, 6, 11, 12, 13, 14, 15 };

    /* lecture des paramètres pour les 3 caméras */
    for(int i = 0; i < nbCameras; i++)
    {
        ParametresCamera parametresCamera;

        id = "camera" + QString::number(i+1) + "/";
        parametresCamera.nom = parametres.value(id + "nom", "").toString();
        parametresCamera.adresse = parametres.value(id + "adresse",
"").toString();
        parametresCamera.port = parametres.value(id + "port", "0").toInt();
```

```
        parametresCamera.user = parametres.value(id + "user",
"admin").toString();
        parametresCamera.pwd = parametres.value(id + "pwd", "").toString();
        parametresCamera.script = parametres.value(id + "script",
"mobile.htm").toString();
        parametresCamera.resolution = parametres.value(id + "resolution",
"32").toInt();
        parametresCamera.rate = parametres.value(id + "rate", "0").toInt();
        parametresCamera.duree = parametres.value(id + "duree",
"10000").toInt();
        parametresCamera.periode = parametres.value(id + "periode",
"300").toInt();
        parametresCamera.chemin = parametres.value(id + "chemin",
"").toString();
        parametresCamera.fichier = parametres.value(id + "fichier",
"enregistrement").toString();
        parametresCamera.extension = parametres.value(id + "extension",
"mjpg").toString();
        parametresCamera.url = parametres.value(id + "url", "").toString();
        parametresCamera.couleur = parametres.value(id + "couleur",
"").toString();
        parametresCameras.push_back(parametresCamera);

        /* on crée la caméra */
        Camera *camera = new Camera(parametresCamera, parametresCamera.couleur,
this);
        connect(this, SIGNAL(fini()), camera, SLOT(arreter()));
        connect(camera, SIGNAL(fini(QString)), this, SLOT(terminer(QString)));
        connect(camera, SIGNAL(journaliser(QString)), this,
SLOT(journaliser(QString)));
        switch(i)
        {
        case 0:
            connect(timer1, SIGNAL(timeout()), camera, SLOT(enregistrer()));
            connect(camera, SIGNAL(progression(int)), progressBar1,
SLOT(setValue(int)));
            break;
        case 1:
            connect(timer2, SIGNAL(timeout()), camera, SLOT(enregistrer()));
            connect(camera, SIGNAL(progression(int)), progressBar2,
SLOT(setValue(int)));
            break;
        case 2:
            connect(timer3, SIGNAL(timeout()), camera, SLOT(enregistrer()));
            connect(camera, SIGNAL(progression(int)), progressBar3,
SLOT(setValue(int)));
            break;
```

```
        }
        cameras.push_back(camera);
    }

    /* ajoute les caméras dans la liste */
    for(int i = 0; i < nbCameras; i++)
    {
        listeCameras->addItem(parametresCameras.at(i).nom);
    }

    /* Par défaut la caméra n°1 */
    listeCameras->setCurrentIndex(0);
    for(int i = 0; i < 2; i++)
    {
        if(choixResolution[i] == parametresCameras.at(0).resolution)
            listeResolution->setCurrentIndex(i);
    }
    for(int i = 0; i < 9; i++)
    {
        if(choixRate[i] == parametresCameras.at(0).rate)
            listeRate->setCurrentIndex(i);
    }
    leDuree->setText(QString::number(parametresCameras.at(0).duree/1000));
    lePeriode->setText(QString::number(parametresCameras.at(0).periode));
    journaliser(QString::fromUtf8("Lecture des paramètres : ok"));
}

void IHM::ecrireParametres() /** Ecriture des paramètres des 3 caméras dans le
fichier .ini */
{
    QString fichierINI = qApp->applicationDirPath() + "/" + qApp-
>applicationName() + ".ini";
    QSettings parametres(fichierINI, QSettings::IniFormat);
    QString id;

    /* écriture des paramètres pour les 3 caméras */
    for(int i = 0; i < nbCameras; i++)
    {
        ParametresCamera parametresCamera = parametresCameras.at(i);

        id = "camera" + QString::number(i+1);
        parametres.beginGroup(id);
        parametres.setValue("nom", parametresCamera.nom);
        parametres.setValue("adresse", parametresCamera.adresse);
        parametres.setValue("port", parametresCamera.port);
        parametres.setValue("user", parametresCamera.user);
        parametres.setValue("pwd", parametresCamera.pwd);
```

```
        parametres.setValue("script", parametresCamera.script);
        parametres.setValue("resolution", parametresCamera.resolution);
        parametres.setValue("rate", parametresCamera.rate);
        //parametres.setValue("position", parametresCamera.position);
        parametres.setValue("duree", parametresCamera.duree);
        parametres.setValue("periode", parametresCamera.periode);
        parametres.setValue("chemin", parametresCamera.chemin);
        parametres.setValue("fichier", parametresCamera.fichier);
        parametres.setValue("extension", parametresCamera.extension);
        parametres.setValue("url", parametresCamera.url);
        parametres.endGroup();
    }
    journaliser(QString::fromUtf8("Écriture des paramètres : ok"));
}

void IHM::majCamera(int index) /** Séléction des paramètres en fonction de la
caméra choisi et de ses paramètres */
{
    Q_UNUSED(index)
    int choixResolution[2] = { 32, 8 };
    int choixRate[9] = { 0, 1, 3, 6, 11, 12, 13, 14, 15 };

    for(int i = 0; i < 2; i++)
    {
        if(choixResolution[i] == parametresCameras.at(listeCameras-
>currentIndex()).resolution)
            listeResolution->setCurrentIndex(i);
    }
    for(int i = 0; i < 9; i++)
    {
        if(choixRate[i] == parametresCameras.at(listeCameras-
>currentIndex()).rate)
            listeRate->setCurrentIndex(i);
    }
    leDuree->setText(QString::number(parametresCameras.at(listeCameras-
>currentIndex()).duree/1000));
    lePeriode->setText(QString::number(parametresCameras.at(listeCameras-
>currentIndex()).periode));

    QString urlvideo = "http://" + parametresCameras.at(listeCameras-
>currentIndex()).adresse + ":" +
QString::number(parametresCameras.at(listeCameras->currentIndex()).port) + "/" +
parametresCameras.at(listeCameras->currentIndex()).script;
    QUrl URLVIDEO = urlvideo;
    URLVIDEO.setUserName(parametresCameras.at(listeCameras-
>currentIndex()).user);
    URLVIDEO.setPassword(parametresCameras.at(listeCameras-
```

```
>currentIndex()).pwd);
    webView->load(URLVIDEO);
}

void IHM::majResolution(int index)
{
    Q_UNUSED(index)
    int choixResolution[2] = { 32, 8 };

    parametresCameras[listeCameras->currentIndex()].resolution =
choixResolution[listeResolution->currentIndex()];
    cameras[listeCameras->currentIndex()]-
>setResolution(parametresCameras.at(listeCameras->currentIndex()).resolution);
    ParametresCamera parametresCamera = parametresCameras.at(listeCameras-
>currentIndex());
    QString URL = "http://" + parametresCamera.adresse + ":" +
QString::number(parametresCamera.port) + "/camera_control.cgi?param=0&value=" +
parametresCamera.resolution + "&user=" + parametresCamera.user + "&pwd=" +
parametresCamera.pwd;
    manager->get(QNetworkRequest(QUrl(URL)));
}

void IHM::majRate(int index)
{
    Q_UNUSED(index)
    int choixRate[9] = { 0, 1, 3, 6, 11, 12, 13, 14, 15 };

    parametresCameras[listeCameras->currentIndex()].rate = choixRate[listeRate-
>currentIndex()];
    cameras.at(listeCameras->currentIndex())-
>setRate(parametresCameras.at(listeCameras->currentIndex()).rate);
}

/*void IHM::majPosition(int index)
{
    Q_UNUSED(index)
    int choixPosition[9] = { 30, 32, 34, 36, 38, 40, 42, 44, 46 };

    parametresCameras[listeCameras->currentIndex()].position =
choixPosition[listePosition->currentIndex()];
    cameras.at(listeCameras->currentIndex())-
>setPosition(parametresCameras.at(listeCameras->currentIndex()).position);
}*/

void IHM::majDuree()
{
    parametresCameras[listeCameras->currentIndex()].duree = leDuree-
```

```
>text().toInt() * 1000;
    cameras.at(listeCameras->currentIndex())-
>setDuree(parametresCameras.at(listeCameras->currentIndex()).duree);
}

void IHM::majPeriode()
{
    parametresCameras[listeCameras->currentIndex()].periode = lePeriode-
>text().toInt();
    cameras.at(listeCameras->currentIndex())-
>setPeriode(parametresCameras.at(listeCameras->currentIndex()).periode);
}

/** commande de déplacement des caméras */

void IHM::right()
{
    ParametresCamera parametresCamera = parametresCameras.at(listeCameras-
>currentIndex());
    QString URL = "http://" + parametresCamera.adresse + ":" +
QString::number(parametresCamera.port) + "/decoder_control.cgi?
command=6&onestep=2" + "&user=" + parametresCamera.user + "&pwd=" +
parametresCamera.pwd;
    manager->get(QNetworkRequest(QUrl(URL)));
}

void IHM::left()
{
    ParametresCamera parametresCamera = parametresCameras.at(listeCameras-
>currentIndex());
    QString URL = "http://" + parametresCamera.adresse + ":" +
QString::number(parametresCamera.port) + "/decoder_control.cgi?
command=4&onestep=2" + "&user=" + parametresCamera.user + "&pwd=" +
parametresCamera.pwd;
    manager->get(QNetworkRequest(QUrl(URL)));
}

void IHM::up()
{
    ParametresCamera parametresCamera = parametresCameras.at(listeCameras-
>currentIndex());
    QString URL = "http://" + parametresCamera.adresse + ":" +
QString::number(parametresCamera.port) + "/decoder_control.cgi?
command=0&onestep=2" + "&user=" + parametresCamera.user + "&pwd=" +
parametresCamera.pwd;
    manager->get(QNetworkRequest(QUrl(URL)));
}
```

```
void IHM::down()
{
    ParametresCamera parametresCamera = parametresCameras.at(listeCameras-
>currentIndex());
    QString URL = "http://" + parametresCamera.adresse + ":" +
QString::number(parametresCamera.port) + "/decoder_control.cgi?
command=2&onestep=2" + "&user=" + parametresCamera.user + "&pwd=" +
parametresCamera.pwd;
    manager->get(QNetworkRequest(QUrl(URL)));
}

void IHM::origine()
{
    ParametresCamera parametresCamera = parametresCameras.at(listeCameras-
>currentIndex());
    QString URL = "http://" + parametresCamera.adresse + ":" +
QString::number(parametresCamera.port) + "/decoder_control.cgi?
command=31&onestep=0" + "&user=" + parametresCamera.user + "&pwd=" +
parametresCamera.pwd;
    manager->get(QNetworkRequest(QUrl(URL)));
}

void IHM::journaliser(QString message)
{
    journal->append(message);
}

void IHM::replyFinished(QNetworkReply *reply)
{
    //reply->read();
    QByteArray datas = reply->readAll();
    #ifdef DEBUG_IHM
    qDebug() << QString::fromUtf8("<IHM::replyFinished()> reply : ") << datas;
    #endif
    QString infos(datas);
    journal->append(infos);
}
```

Annexe 4 : déclaration de la classe Camera : Fichier camera.h

```
#ifndef CAMERA_H
#define CAMERA_H

#include <QObject>

/** La structure de données pour les parametres d'une caméra */
typedef struct
{
    QString nom; // par exemple "Site Alpin"
    QString adresse; // par exemple "192.168.52.95"
    int port; // par exemple 99
    QString user; // par exemple "admin"
    QString pwd; // par exemple ""
    QString script; // par exemple "mobile.htm"
    int resolution; // par exemple 32
    int rate; // par exemple 0
    //int position; //par exemple 30
    int duree; // par exemple 10000 ms
    int periode; // par exemple 300 s (5 mn)
    QString chemin; // emplacement de stockage des fichiers videos
    QString fichier; // le préfixe du nom de fichier par exemple "camera1"
    QString extension; // par exemple "mjpg"
    QString url; // par exemple "http://192.168.52.93:99/videostream.cgi?
user=admin&pwd="
    QString couleur; // pour les messages
} ParametresCamera;

class Video;

/** @class Camera
    @brief Déclaration des paramètres des caméras */
class Camera : public QObject
{
    Q_OBJECT
public:
    explicit Camera(ParametresCamera parametresCamera, QString couleur, QObject
*parent = 0);

    void setResolution(int resolution); /** @brief Permet de définir la
résolution */
    void setRate(int rate); /** @brief Permet de définir la fréquence
d'images/seconde */
    void setDuree(int duree); /** @brief Permet de définir la durée de
l'enregistrement en secondes */
    void setPeriode(int periode); /** @brief Permet de définir la période entre
deux enregistrements en secondes */
```

```
private:
    ParametresCamera parametresCamera;
    Video *video;
    QString couleur;

signals:
    void fini(QString); /** @brief envoie le signal au slot terminer */
    void progression(int pas); /** @brief envoie le signal au slot
progresser(int) qui va envoyer la valeur de la barre de progression calculé par
la méthode video::progresser(int) à la progressBar */
    void journaliser(QString); /** @brief Diffuse des messages dans le journal
*/

public slots:
    void enregistrer(); /** @brief Démarre l'enregistrement de la vidéo lorsque
le timer arrive à 0 */
    void arreter();
    void progresser(int pct);
    void terminer(); /** @brief Met fin à l'enregistrement de la video à 100% */
};

#endif // CAMERA_H
```

Annexe 5 : définition de la classe Camera : Fichier camera.cpp

```
#include "camera.h"
#include "video.h"
#include <QDebug>
#include "debug.h"

Camera::Camera(ParametresCamera parametresCamera, QString couleur, QObject
*parent) : QObject(parent)
{
    this->parametresCamera = parametresCamera;
    this->couleur = couleur;

    qDebug() << Q_FUNC_INFO << this->parametresCamera.nom <<
QString::fromUtf8("url=%1 chemin=%2 prefixe=%3 extension=%4").arg(this-
>parametresCamera.url).arg(this->parametresCamera.chemin).arg(this-
>parametresCamera.fichier).arg(this->parametresCamera.extension);

    video = new Video(this, this->parametresCamera.nom, this-
>parametresCamera.url, this->parametresCamera.chemin, this-
>parametresCamera.fichier, this->parametresCamera.extension);
    connect(video, SIGNAL(progress(int)), this, SLOT(progresser(int)));
    connect(video, SIGNAL(fini()), this, SLOT(terminer()));
}

void Camera::setResolution(int resolution) /** défini la résolution */
{
    this->parametresCamera.resolution = resolution;
}

void Camera::setRate(int rate) /** défini la fréquence d'images */
{
    this->parametresCamera.rate = rate;
}

void Camera::setDuree(int duree) /** défini la durée de l'enregistrement */
{
    this->parametresCamera.duree = duree;
}

void Camera::setPeriode(int periode) /** défini la période entre chaque
enregistrement */
{
    this->parametresCamera.periode = periode;
}

void Camera::enregistrer()
{
```

```
    QString fichier = video->getFichier();
    fichier.remove(0, 8); // retire le préfixe "file/ts:"
    emit journaliser(QString::fromUtf8("<b><font color=%1>Démarrage
enregistrement caméra %2 (durée %3 s) :
%4</font></b>").arg(couleur).arg(parametresCamera.nom).arg(parametresCamera.dure
e/1000.).arg(fichier));
    video->acquerir(parametresCamera.duree);
}

void Camera::arreter()
{
    video->arreter();
    emit journaliser(QString::fromUtf8("<b><font color=%1>Arrêt caméra
%2</font></b>").arg(couleur).arg(parametresCamera.nom));
}

void Camera::progresser(int pct)
{
    emit progression(pct);
}

void Camera::terminer()
{
    QString fichier = video->getFichier();
    fichier.remove(0, 8); // retire le préfixe "file/ts:"
    emit journaliser(QString::fromUtf8("<b><font color=%1>Fin enregistrement
caméra %2 :
%3</font></b>").arg(couleur).arg(parametresCamera.nom).arg(fichier));
    emit fini(parametresCamera.nom);
    emit progression(0);
}
```

Annexe 6 : déclaration de la classe Video : Fichier video.h

```
#ifndef _VIDEO_H
#define _VIDEO_H

#include <QObject>
#include <QString>
#include <QStringList>
#include <QProcess>
#include <QTimer>

typedef QObject typedef2;
class Video : public typedef2 {
  Q_OBJECT
  public:
    Video(QObject * parent = 0, QString url = "", QString chemin = "", QString
fichier = "enregistrement", QString extension = "mjpg");

public:
    Video(QObject *parent=0, QString nom="", QString url="", QString chemin="",
QString fichier="enregistrement", QString extension="mjpg");
    ~Video();

    void        setNom(QString nom);
    void        setUrl(QString url);
    QString     getFichier() const;

    void        setFichier(QString chemin, QString fichier, QString extension);

  private:
    QString programme;

    QStringList arguments;
    QProcess    *processus;
    QString     nom;
    QString     url;
    QString     chemin;
    QString     fichier;
    QString     extension;
    QString     fichierVideo;
    QTimer      *timer;
    int         duree;
    bool        record;
    int         progression;

signals:
    void        fini();
    void        progress(int);
```

```
public slots:
    void        acquerir(int duree=0);
    void        arreter();
    void        progresser();
    void        demarre();
    void        arrete(int);
};
#endif
```

Annexe 7 : définition de la classe Video : Fichier video.cpp

```
#include <QApplication>
#include <QDateTime>
#include "video.h"
#include "debug.h"

Video::Video(QObject *parent, QString nom, QString url, QString chemin, QString
fichier, QString extension) : QObject(parent), programme("cvlc"), arguments(""),
processus(NULL), nom(nom), url(url), chemin(chemin), fichier(fichier),
extension(extension), fichierVideo(""), timer(NULL), duree(0), record(false),
progression(0)
{
    setFichier(chemin, fichier, extension);

    processus = new QProcess(this); /* Crée le processus */
    timer = new QTimer(this); /* crée le timer */

    /* signal/slots */
    connect(timer, SIGNAL(timeout()), this, SLOT(progresser()));
    connect(processus, SIGNAL(started()), this, SLOT(demarre()));
    connect(processus, SIGNAL(finished(int)), this, SLOT(arrete(int)));
}

Video::~Video(){
}

void Video::setUrl(QString url) { /** Défini l'Url */
    this->url = url; /* Déclare la variable url */
    arguments.clear(); /* Efface les arguments */
    arguments << this->url << "--sout" << fichierVideo;
}

void Video::setFichier(QString chemin, QString fichier, QString extension) { /**
Défini le fichier vidéo */
    if(chemin.length() == 0)
        this->chemin = qApp->applicationDirPath();
    else
        this->chemin = chemin;
    if(fichier.length() == 0)
        this->fichier = "enregistrement";
    else
        this->fichier = fichier;
    if(extension.length() == 0)
        this->extension = "mjpg";
    else
        this->extension = extension;
```

```
    QDateTime maintenant = QDateTime::currentDateTime();
    QString format="dd-MM-yyyy_hh-mm-ss"; /* Horodate */
    fichierVideo = "file/ts:" + this->chemin + "/" + this->fichier + "_" +
QString::number(maintenant.toMSecsSinceEpoch()) + "_" +
maintenant.toString(format) + "." + this->extension;

    qDebug() << Q_FUNC_INFO << nom << QString::fromUtf8("%1").arg(fichierVideo);

    arguments.clear();
    arguments << this->url << "--sout" << fichierVideo;
}

QString Video::getFichier() const /** Obtenir fichier vidéo */
{
    return fichierVideo;
}

void Video::acquerir(int duree) /** Acquisition vidéo */
{
    if(duree != 0)
    {
        qDebug() << Q_FUNC_INFO << nom << programme << arguments;
        setFichier(chemin, fichier, extension);
        this->duree = duree;
        this->progression = 0;
        this->record = true;
        timer->start(duree/100);
        processus->start(programme, arguments); /* Démarre le processus avec le
programme cvlc en arrière plan */
    }
}

void Video::arreter() { /** Arrête l'enregistrement */
    if(record == true)
    {
        if(timer->isActive())
            timer->stop();
        processus->kill();
        record = false;
#ifdef DEBUG_VIDEO
        qDebug() << Q_FUNC_INFO;
#endif
    }
}

void Video::progresser() { /** Calcule l'avancement de la progression en
fonction de la durée, arrêt de la progression si égal à la durée */
```

```
    if(progression == duree)
    {
        arreter();
    }
    else
    {
        progression += (duree/100);
        emit progress(((double)progression/(double)duree)*100.);
    }
}

void Video::demarre()
{
    qDebug() << Q_FUNC_INFO << nom;
}

void Video::arrete(int etat) {
    Q_UNUSED(etat)
    emit fini();
}
```

Annexe 8 : déclaration du debuggage : Fichier debug.h

```
#ifndef DEBUG_H
#define DEBUG_H

#include <QDebug>

#define DEBUG_IHM
#define DEBUG_VIDEO

#endif // DEBUG_H
```

Annexe 9 : Programme principal : Fichier main.cpp

```
#include <QApplication>
#include <QTextCodec>
#include "ihm.h"

int main(int argc, char **argv)
{
    QApplication a(argc, argv);
    QTextCodec::setCodecForCStrings(QTextCodec::codecForName("UTF-8"));

    a.setApplicationName("Acquisition");

    IHM ihm;
    ihm.adjustSize();
    ihm.show();

    return a.exec();
}
```

Annexe 10 : Configuration des caméras : Fichier Acquisition.ini

```
[camera1]

nom=Site Alpin
adresse=192.168.52.93
port=99
user=admin
pwd=
script=mobile.htm
resolution=32
rate=0
position=30
type=0
duree=15000
periode=30
fichier=camera1
url="http://192.168.52.93:99/videostream.cgi?user=admin&pwd="
chemin=/camera/
extension=mjpg
couleur=orange

[camera2]
nom=Site Fond
adresse=192.168.52.95
port=99
user=admin
pwd=
script=mobile.htm
resolution=32
rate=1
position=30
type=0
duree=20000
periode=600
fichier=camera2
url="http://192.168.52.95:99/videostream.cgi?user=admin&pwd="
chemin=/camera/
extension=mjpg
couleur=blue

[camera3]
nom=Site Snow-Kite
adresse=192.168.52.94
port=99
```

```
user=admin
pwd=
duree=10000
chemin=/camera/
origine=
type=0
fichier=camera3
extension=mjpg
resolution=32
rate=0
position=34
periode=900
url="http://192.168.52.94:99/videostream.cgi?user=admin&pwd="
couleur=green
script=mobile.htm
```

Académie Aix-Marseille

BTS SN 2016-2017

E6 – Projet Informatique

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

-

ZANCA Killian
FOULARD Jimmy

LT La Salle Avignon

SOMMAIRE

Groupement académique n°1 : Aix-Marseille, Montpellier, Nice, Corse	**Session 2015**
Lycée : St Jean Baptiste de La Salle	
Ville : Avignon	
Nom du projet : Projet Wismas (Weather Information System Multi Activity Station)	

1. Cahier des charges

1.1. Présentation et situation du projet dans son environnement

1.1.1. Contexte de réalisation

Statut des étudiants	Candidats scolarisés en temps plein
Equipe de développement	4 étudiants
	Etudiant EC 1 : Balthazar Opperman Etudiant EC 2 : Jeanlin Blairon Etudiant IR 1 : Killian Zanca Etudiant IR 2 : Jimmy Foulard
Projet développé et suivi	Entreprise Partenaire : oui □ non√ Origine du projet : - Idée : Lycée √ Entreprise □ - Suivi : Lycée √ Entreprise □
Budget	

1.1.2. Situation du projet

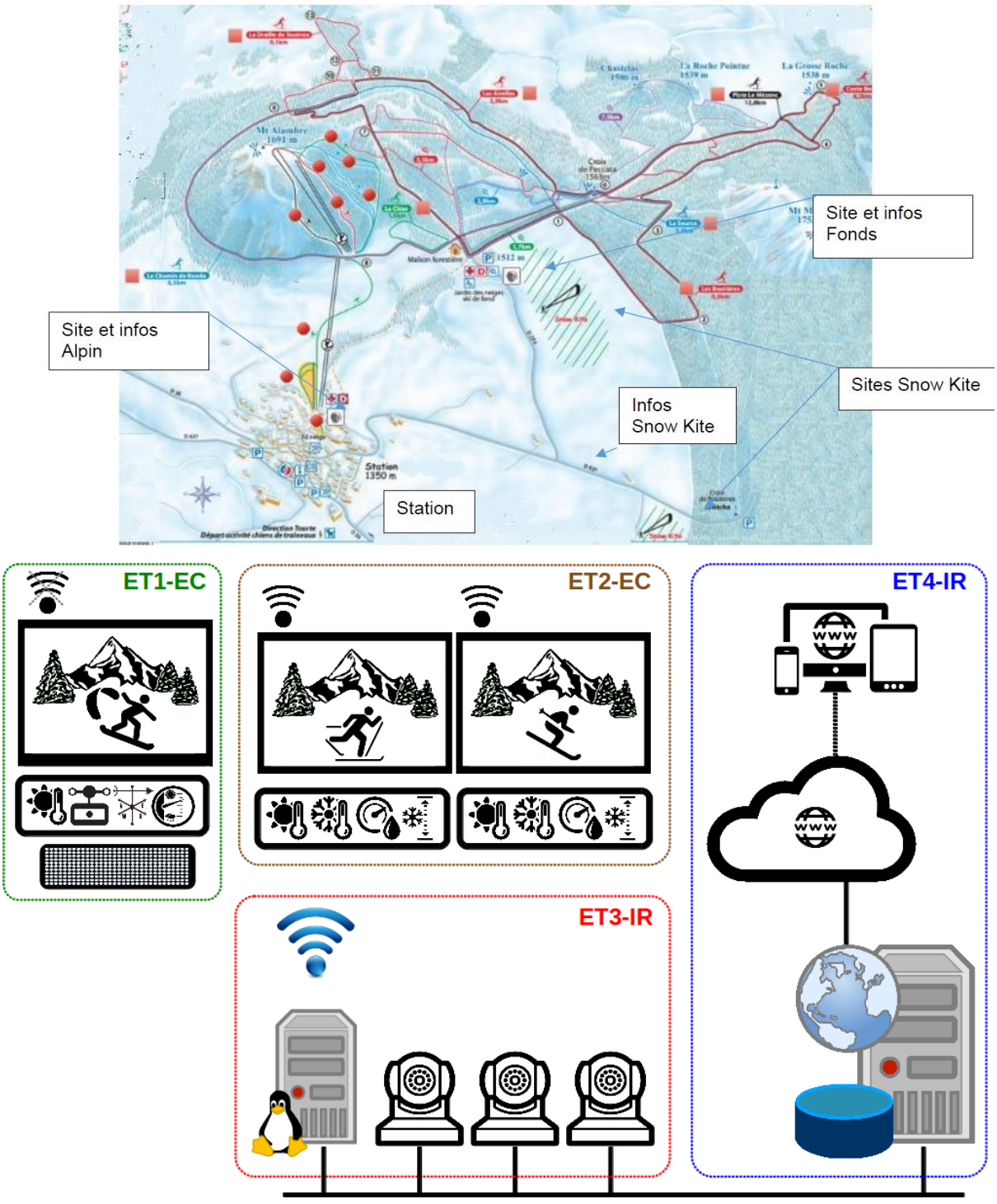
Catégorie de système du projet	
Moyens de production Exploitation par des professionnels en milieu industriel (contrôle/commande de production)	
Service techniques	
Biens d'équipement	X

1.1.3. Objectifs professionnels du projet

Domaines d'activités professionnelles abordés et développés avec le projet	
Analyser et spécifier le système informatique à développer	X
Réaliser la conception générale et détaillée	X
Coder et réaliser	X
Tester, mettre au point et valider	X
Intégrer et interconnecter des systèmes	X
Installer, explorer, optimiser et maintenir	X
Assurer l'évolution locale ou la rénovation d'un système informatique	X
Coopérer et communiquer	X

1.2. Présentation du projet

La station sur laquelle s'appuie l'étude est une station de moyenne montagne multi-activités dans laquelle on peut faire du ski de fond, du ski de piste et également du Snow-kite sur 3 sites voisins



1.2.1. Analyse de l'existant

- Le système existant sur le site est un ensemble de « petits » systèmes non reliés :
- Un panneau lumineux affichant température et date sur les sites Alpin et Fond (non inclus dans le futur système Wismas).
- Des caméras sur les sites Fond et Alpin.

- Un site web permettant de connaitre les pistes ouvertes, la météo du site (prévision) et de visualiser en temps réel l'image des caméras.

1.2.2. Expression du besoin

L'objectif est de moderniser l'ensemble en y ajoutant des fonctionnalités.
Il s'agit d'une part de développer sur carte ATMEL SAM4S-EK les applications nécessaires à la mise en œuvre des différents capteurs et à la transmission sans fil des données vers le PC de gestion.
D'autre part la gestion des panneaux lumineux et des écrans de diffusion d'images.
Et enfin la réception et traitement des données, la gestion de la base de données et la gestion du site (accès et mise à jour des données).
Le système sera construit à partir de carte(s) microcontrôleur qui seront mises en place sur les sites de mesures et d'un poste PC de contrôle.
L'ensemble du système sera équipé de :

- Site Fond :
  - Capteur de température de l'air
  - Capteur d'humidité de l'air
  - Capteur de pression atmosphérique
  - Capteur de hauteur de neige
  - Capteur de température de la neige
  - Caméra IP
  - Ecran de diffusion d'images
- Site Alpin :
  - Capteur de température de l'air
  - Capteur d'humidité de l'air
  - Capteurs de hauteur de neige
  - Caméra IP
  - Ecran de diffusion d'images.
- Site Snow-Kite :
  - Capteur de température de l'air
  - Capteur de vitesse de vent
  - Capteur de direction de vent
  - Panneau lumineux pour les conditions d'accès au site Snow Kite

D'autre part, l'application logicielle « Acquisition » doit permettre :

- D'acquérir les images vidéo des caméras IP installées sur les différents sites
- D'enregistrer les vidéos sur le serveur pour une diffusion ultérieure
- De commander les déplacements d'une caméra pour une acquisition panoramique
- De relever les mesures des stations météorologiques installées sur les différents sites
- D'enregistrer les mesures dans une base données installée sur le serveur

L'administrateur pourra paramétrer :

- Les adresses IP et numéros de port des différentes caméras
- La position d'origine de la prise de vue en pouvant déplacer la caméra manuellement
- Le type (fixe ou panoramique) et la durée d'acquisition vidéo

Les paramètres de configuration seront stockés dans un fichier .INI.
Enfin, le site web sera hébergé sur le serveur et permettra aux clients d'accéder aux pages suivantes :

- Informations sur la station (accès, horaires, tarifs, …)

- Conditions météorologiques suivant le site (Alpin, Fond ou Snow kite) avec la possibilité de visualiser une vidéo du lieu
- Conseils de fartage en fonction des conditions météorologiques

1.2.3. Enonce des taches à réaliser par les étudiants

Étudiant 1 (EC)

Fonctions :

- Mesurer la température
- Mesurer l'hygrométrie
- Mesurer la vitesse et le sens du vent
- Mettre en forme les mesures
- Gérer l'affichage sur panneau lumineux

Installation :

- Du panneau lumineux

Mise en œuvre :

- Des différents capteurs
- Mettre en forme les mesures
- Gérer l'affichage sur panneau lumineux

Configuration :

- Du panneau lumineux

Réalisation :

- Les diagrammes SYSML, le code source de l'application

Documentation :

- Le dossier technique et les documents relatifs au module

Étudiant 2 (EC)

Fonctions :

- Mesurer la température de l'air et de la neige
- Mesurer l'hygrométrie
- Mesurer la hauteur de neige
- Mettre en forme les mesures
- Transmettre ces mesures via une liaison sans fil

Installation :

- Système de mesure de hauteur de neige

Mise en œuvre :
- Des différents capteurs
- Liaison sans fil

Configuration :
- Liaison sans fil

Réalisation :
- Les diagrammes SYSML, le code source de l'application

Documentation :
- Le dossier technique et les documents relatifs au module

Étudiant 3 (IR)
Cas d'utilisation :
- Acquérir vidéo
- Enregistrer (vidéo et mesures)
- Déplacer caméra
- Paramétrer le système
- Relever les mesures des sites

Installation :
- Les caméras

Mise en œuvre :
- Les caméras, la liaison sans fil
- L'environnement de développement

Configuration :
- Les caméras, la liaison sans fil

Réalisation :
- Les diagrammes SysML (diagramme de définition de blocs et diagramme interne de bloc de son module) et diagrammes UML,
- L'IHM et les classes du module

Documentation :
- Le dossier technique et les documents relatifs au module,
- Un guide de mise en route et d'utilisation du module

Étudiant 4 (IR)
Cas d'utilisation :

- Consulter les informations sur la station
- Consulter les conditions météorologiques d'un site
- Visualiser les vidéos
- Obtenir des conseils de fartage

Installation :

- Le serveur (OS, MySQL, Apache, …)

Mise en œuvre :

- Le serveur (OS, MySQL, Apache, …),
- L'environnement de développement

Configuration :

- Le serveur (OS, MySQL, Apache, …)

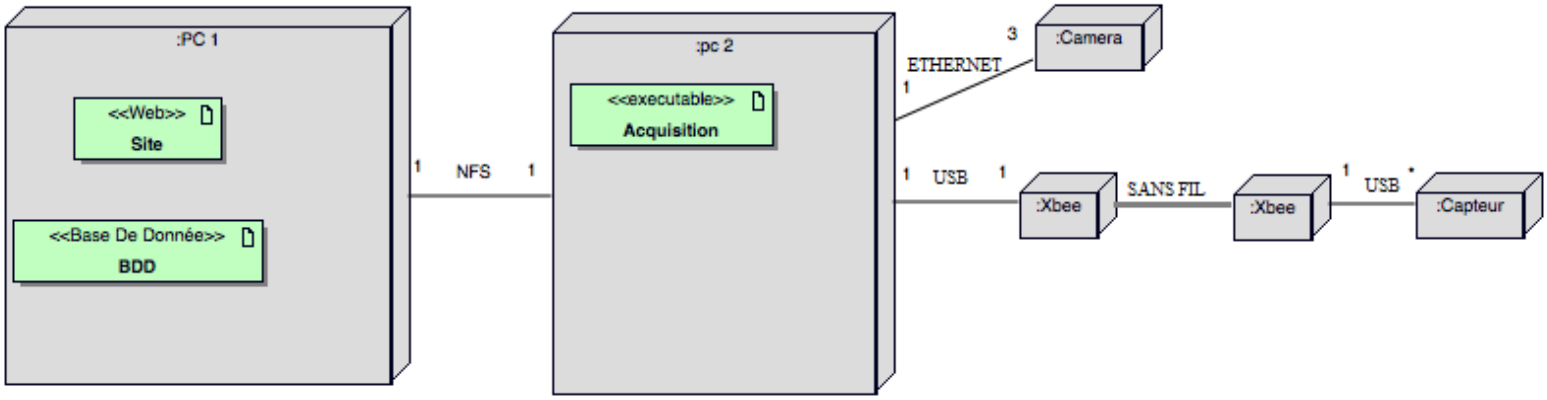
Réalisation :

- Les diagrammes SysML (diagramme de définition de blocs et diagramme interne de bloc de son module) et diagrammes UML,
- L'IHM et les classes du module

Documentation :

- Le dossier technique et les documents relatifs au module,
- Un guide de mise en route et d'utilisation du module

Diagramme de deploiement



Partie personnelle – IR2

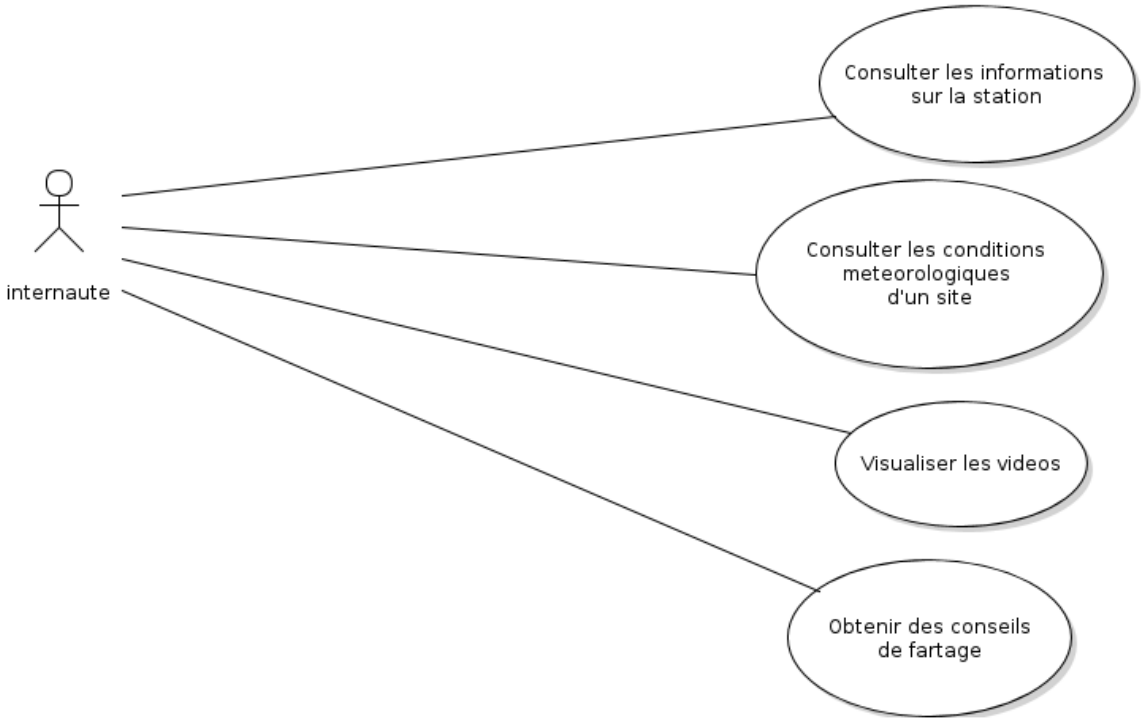
Etudiant 4

Jimmy FOULARD

3.1) Taches de l'étudiant :

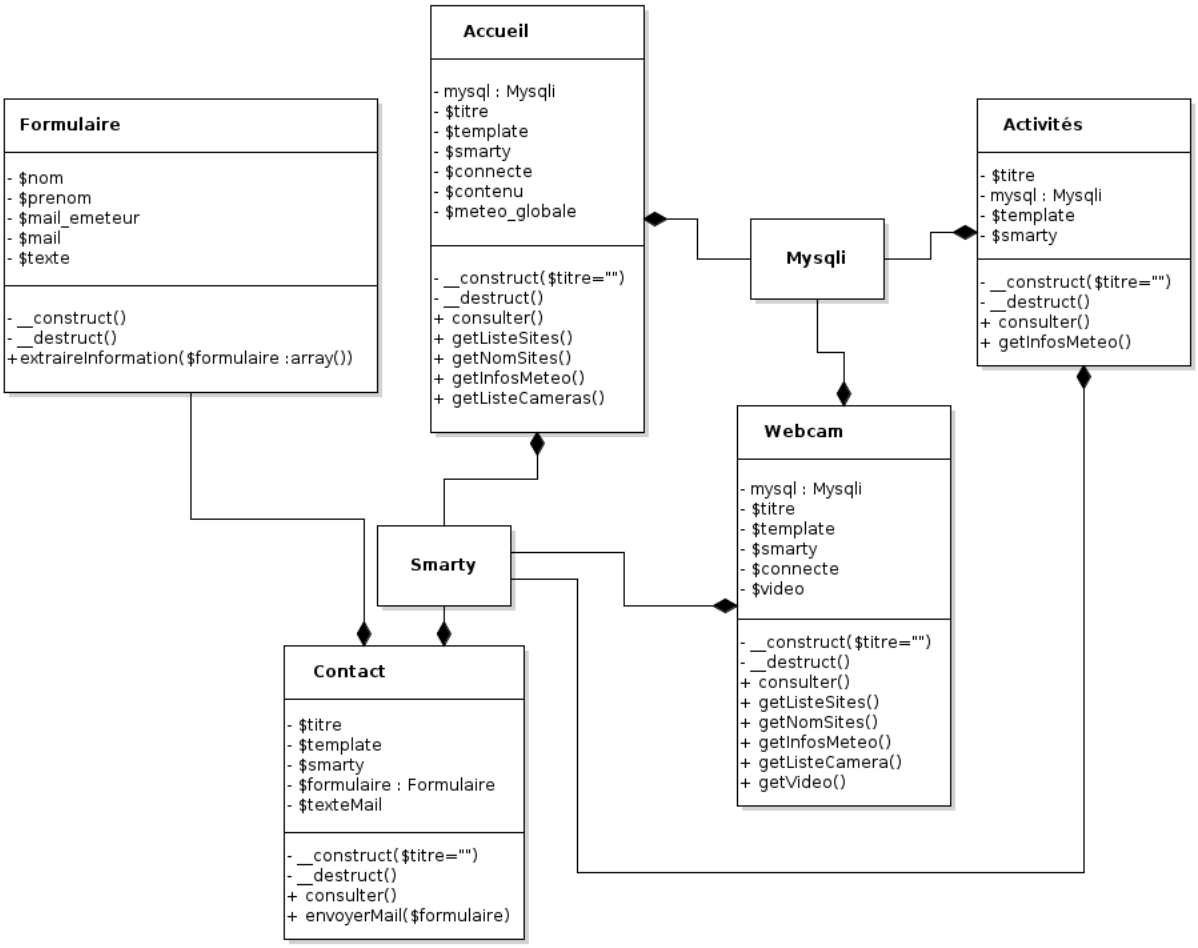
- Créer un site Web PHP.
- Créer une Base de données.
- Rendre consultable les informations générale et météorologiques de la station.
- Permettre la visualisation des vidéos enregistrées par les caméras.
- Diffuser sur le site internet des conseils de fartage en fonction des conditions météorologiques.

3.2) Diagramme de Cas d'utilisation



Un internaute doit être capable de consulter le site web afin d'obtenir des informations sur la station, sur les conditions météorologiques.
Un internaute peut aussi obtenir des conseils de fartage (conseils sur le revêtement des skis en fonction des conditions de glisse) via le site web.

3.3) Diagramme de Classe



La classe Accueil correspond à la page Index, la première page que voit un internaute qui arrive sur le site, elle donne une vision d'ensemble de la station :

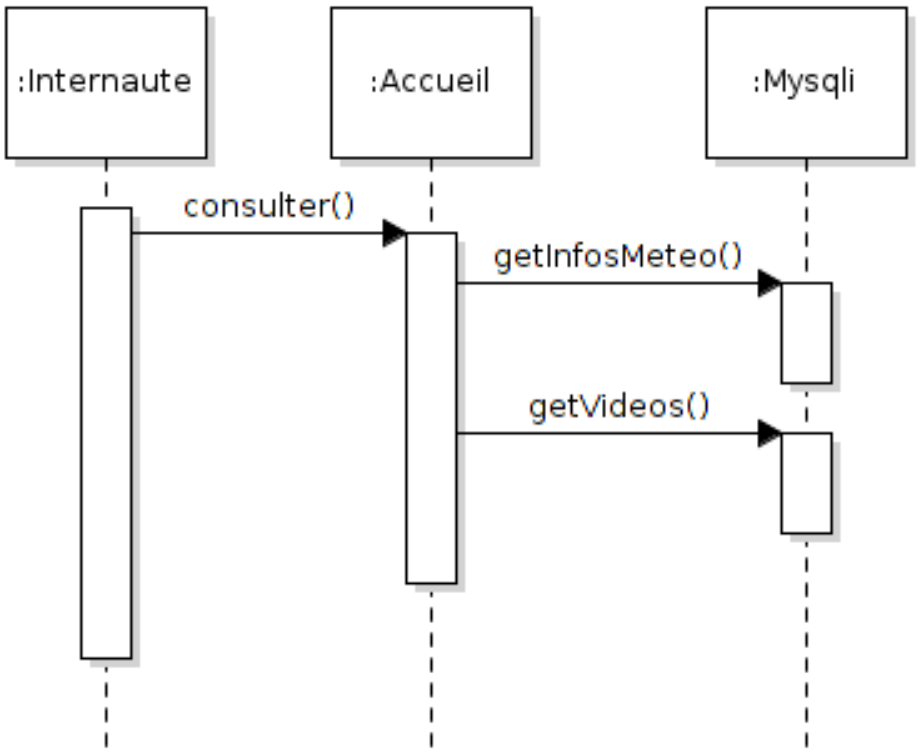
- Infos météorologiques basiques
- Dernière vidéos enregistré
- Divers photos
- Résumé des activités possible

La classe Webcam permet d'accéder aux vidéos enregistrées sur les sites via les webcam, en sélectionnant à partir de la date et de l'horodatage la vidéo que l'internaute veut visionner.

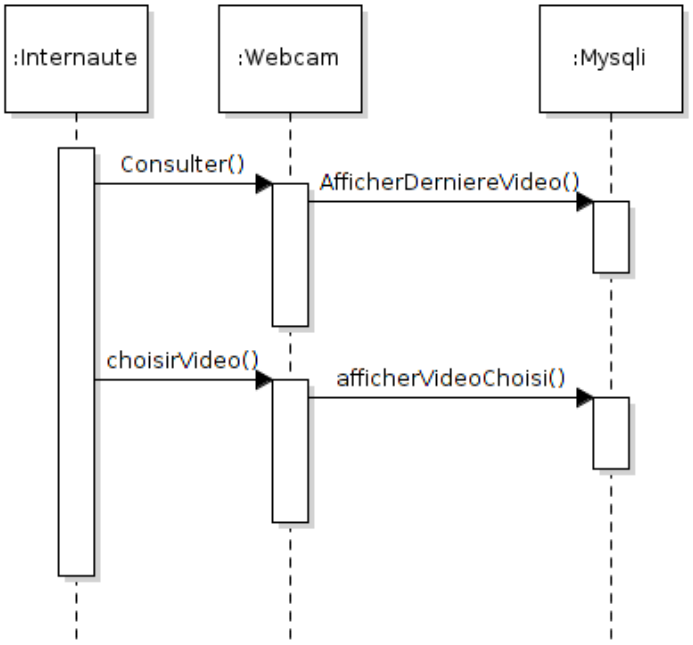
La Classe Découverte fournit à l'internaute des informations complémentaires sur la station, elle donne donc toutes les informations météorologiques correspondantes aux sites de la station, et décrit de manière plus précise la station et les activités sportives disponibles en fonction de la période de l'année.

La classe Activités décrit si il y a des évènements prévu au cours de l'année (course, festival), décrit les différentes activités praticable au sein de la station,

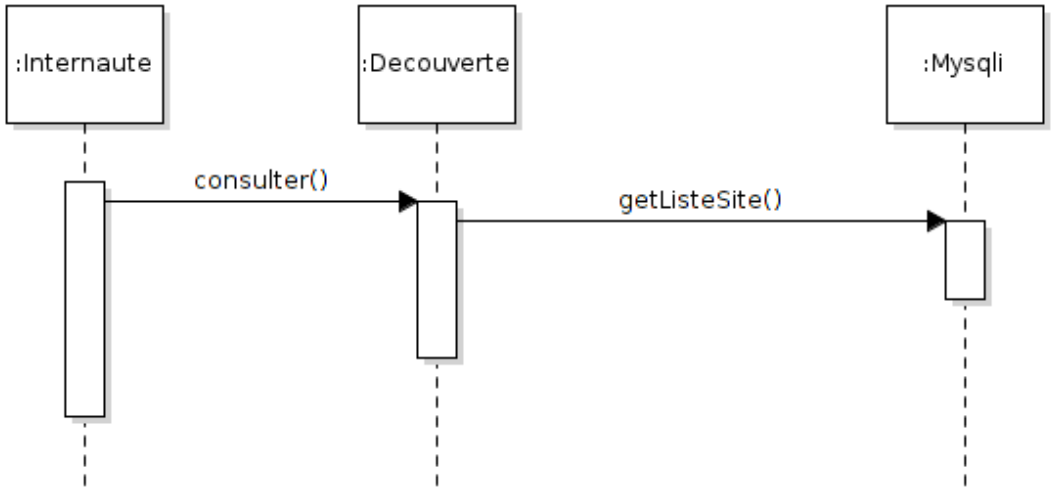
3.4) Diagramme de Séquence



L'internaute consulte la page Accueil, qui récupère les dernières données météorologiques des trois sites et dernière vidéo du site alpin (site ayant l'idSite 1).



L'internaute consulte le site, la page webcam affiche par défaut la dernière vidéo du site Alpin (requête MySQL afin de récupérer la dernière vidéos du site ayant l'id 1)



L'internaute consulte la page Découverte, qui affiche les différents sites de la station récupéré via une requête MySQL.

La Classe MySQL est la classe permettant l'interaction avec la base de données MySQL.

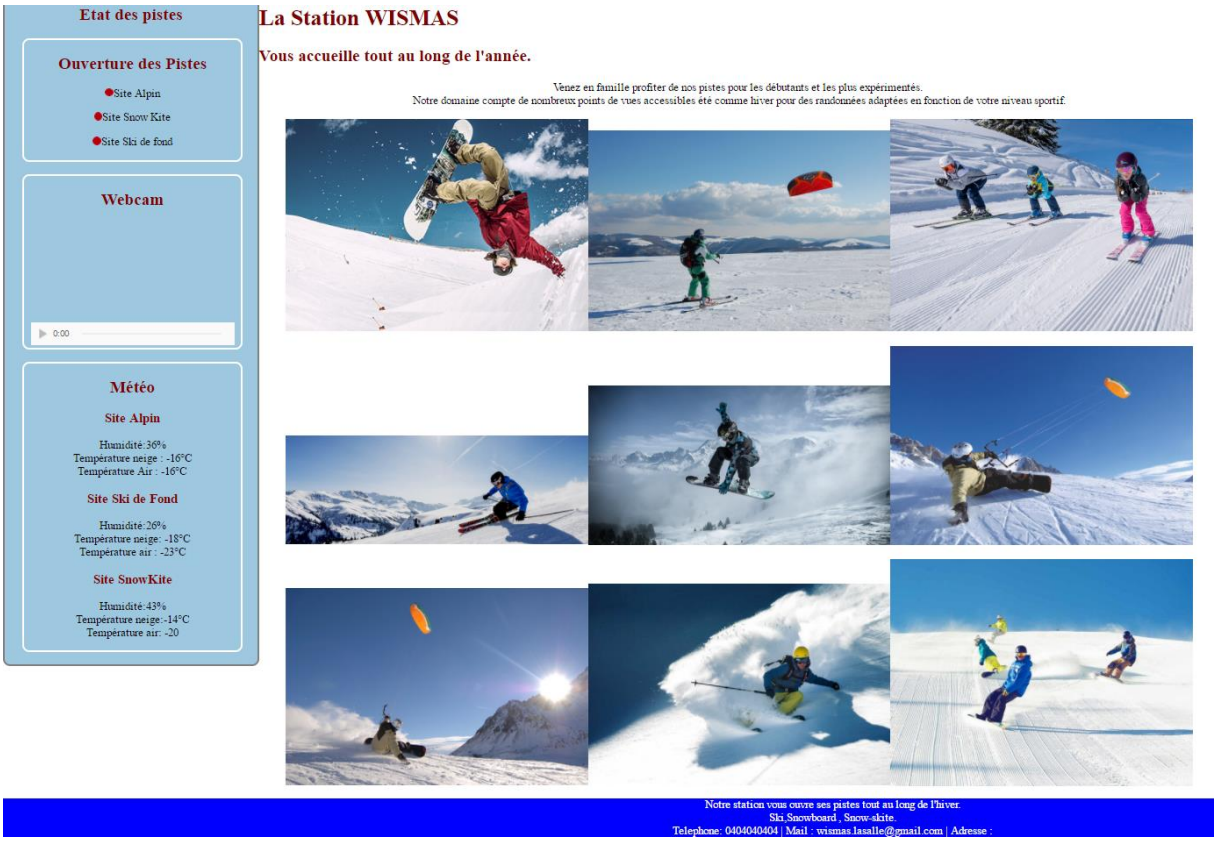
3.5) Le Site Internet

Le Site internet se découpe en cinq pages :

- L'accueil (page Principale, présentant la station et les informations météorologiques)
- La page Activités
  - Présente les différentes activités de la station
  - Donne les dates d'évènements prochains (festival/courses/épreuves de ski)
- La page contact
- La page Webcam
  - Permet le visionnage de vidéos plus ancienne enregistré par les cameras disposé sur les Sites de la station

3.5.1) La page Accueil

La première page du site est la page Accueil (index.php), on y retrouve une rapide introduction présentant la station et les activités praticable au sein de la station, illustré par des images.



On y retrouve aussi dans le bandeau gauche l'état des pistes (ouvertes/fermé/en damage), la dernière vidéo enregistré par les cameras visionnable sur les sites, et les dernières informations météorologiques enregistré sur la base de donnée

3.5.2) La page activités

La page activités présente les activités proposé par la station, ainsi que les animations diverses prévues prochainement tels que les festivals ou des concours

Accueil
Webcam
Contact
WISMAS
LASALLE
ACTIVITES
La Station Wismas propose de nombreuses activités adaptées en fonction de vos envies et de votre niveau d'aptitude.
été
hiver
Randonnée
Point de Vue
Activités : Randonnée
Si la montagne vous appelle mais que la neige de l'hiver ne fait pas partie de vos préférences, notre station reste ouverte l'été afin de vous permettre d'accéder à nos chemins entretenus par nos services.
Ces chemins sont marqués afin de correspondre au mieux à votre niveau sportif.
Toutes nos routes mènent à un point de vue à partir duquel vous pourrez admirer la région.
Notre station vous ouvre ses pistes tout au long de l'hiver.
Ski,Snowboard , Snow-skite.
Telephone: 0404040404 | Mail : wismas.lasalle@gmail.com | Adresse :

Accueil
Webcam
Contact
WISMAS
LASALLE
ACTIVITES
La Station Wismas propose de nombreuses activités adaptées en fonction de vos envies et de votre niveau d'aptitude.
été
hiver
Ski
Snow Board
Kite Snow
Activités : Ski
Profitez de nos pistes de ski adaptées pour tout les niveaux et tout les âges.
Seul ou en groupe selon vos préférences, nos moniteurs vous accompagneront dans votre progression afin de vous fournir les meilleurs conseils et les meilleures techniques.
Nos pistes vous attendent.
Notre station vous ouvre ses pistes tout au long de l'hiver.
Ski.Snowboard , Snow-skite.
Telephone: 0404040404 | Mail : wismas lasalle@gmail.com | Adresse :

3.6) Base de Données

Afin de stocker les données reçues par les capteurs et les chemins d'accès aux vidéos, une base de données MySQL (voir 3.6 logiciel utilisé) a été créé (voir annexe).

Elle est composée de 5 tables :

- sites

Cette table contient les les idSites (Clé Primaire), le nom des sites et l'état (ouvert ou fermé) des diffèrent site.

- Cameras

Cette table permet de stocker le chemin d'accès des vidéos enregistré à partir des caméras situé sur les sites.
Il y a donc un id (Primary key), le nom de la camera, une potentiel description de la vidéo (char(99)), le chemin d'accès aux vidéos, le préfixe des vidéos, le type de la vidéo (mjpg, mp4, etc), l'état de la camera (si la camera est en fonctionnement ou non), la résolution de la vidéo enregistré (hauteur et largeur), et l'idSite(clé étrangère)

idCamera	nom	description	chemin	prefixe	extension	etat	largeur	hauteur	idSite
1	camera1	...	/var/www/wismas/videos	camera1	mjpg	1	640	480	1
2	camera2	...	/var/www/wismas/videos	camera2	mjpg	1	640	480	2
3	camera3	...	/var/www/wismas/videos	camera3	mjpg	1	640	480	3

Table Camera

- HauteurNeige/tempNeige/tempAir

  Ces tables sont construites de la même manière.
  Elles contiennent un id (Clé primaire auto incrémenté), l'id site (clé étrangère) correspondant au site d'où proviennent les valeurs et les valeurs des données relevées.

id	idSite	hauteur	horodatage
1	2	76	2017-05-03 10:00:03
2	3	63	2017-05-03 10:00:03
3	1	57	2017-05-03 10:00:03

Table hauteur

La hauteur de la neige est enregistrée en Cm

id	idSite	temperature	horodatage
1	1	-16	2017-05-03 09:29:37
2	2	-18	2017-05-03 09:49:00
3	3	-14	2017-05-03 09:53:34

Table Température Air

La température de l'air est enregistrée en degré Celsius

id	idSite	temperature	horodatage
1	3	-20	2017-05-03 09:45:00
2	1	-16	2017-05-03 09:46:19
3	2	-23	2017-05-03 09:47:34

Table température Neige

La température de la neige est enregistrée en degré Celsius

3.7) Logiciel utilisé:

- MySQL

MySQL est un système de gestion de bases de données disponible sous licence GPL et propriétaire et est actuellement l'un des plus utilisés à travers le monde.
SQL fait référence au Structured Query Language, qui est le langage de requête utilisé.

- Bootstrap

Bootstrap est une collection d'outils utile à la création du design (graphisme, animation et interactions avec la page dans le navigateur ... etc. ...) de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.

- VIM

VIM est un éditeur de texte, un logiciel permettant la manipulation de fichiers texte, provenant de VI, éditeur présent sur les systèmes d'exploitation de type UNIX), d'ailleurs son nom signifie Vi IMproved (VI amélioré).

- BOUML 

Bouml est un logiciel permettant la création de diagrammes UML, ce logiciel payant permet la retro-ingénierie et la génération de code à partir des informations entré dans les diagrammes

- Violet UML Editor

Violet UML Editor est comme son nom l'indique un logiciel gratuit permettant la création et l'Edition de diagramme UML

4) ANNEXE

Diagramme de la base de données