Document Technique

Table des matières

[Adresses IP : 1](#_Toc90280708)

[Classes : 1](#_Toc90280709)

[MCD 2](#_Toc90280710)

[Diagramme de déploiement 2](#_Toc90280711)

[Class Diagram 3](#_Toc90280712)

[Explication méthodes 3](#_Toc90280713)

# Adresses IP:

IP WEB :192.168.65.124

IP DB: 192.168.65.137

User: root

Pass: root

IP C++: 192.168.65.28

Pour lancer exécutable : cd Camera/

Puis ./Camera

# Classes :

Web :

-User : Contient les infos utilisateurs.

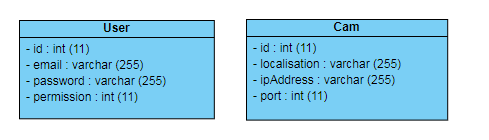
-Cam : Initialise et modifie une caméra.

-CamMove : Permet d’envoyer un JSON pour déplacer la caméra

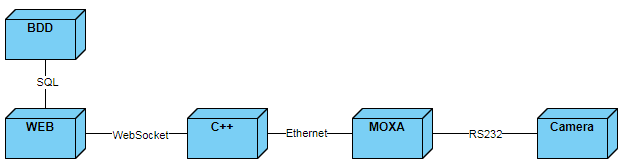
C++ :

-Camera : Effectue la communication entre le site et la caméra

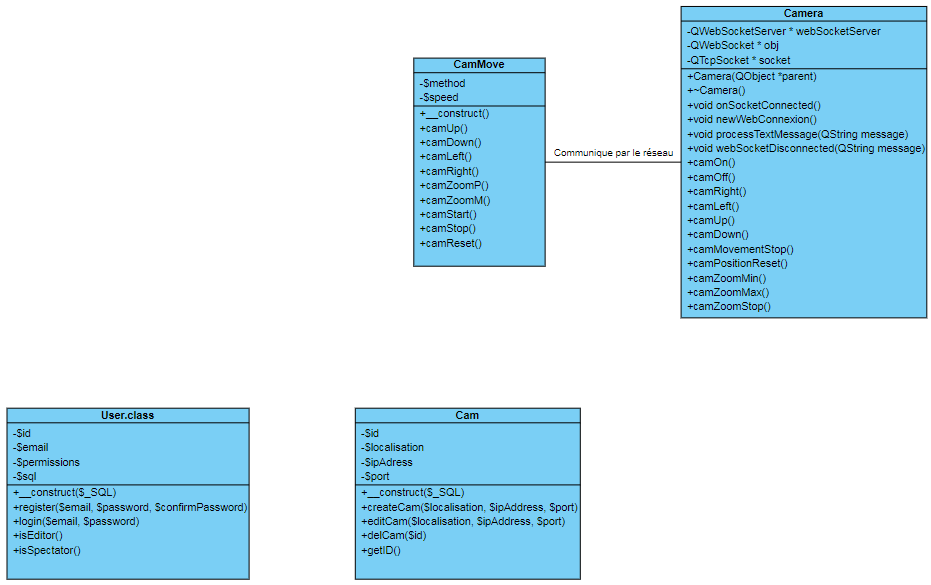
# MCD

****

# Diagramme de déploiement



# Class Diagram



# Explication méthodes

Camera :

|  |  |
| --- | --- |
| Méthode webSocket | Méthode C++ |
| camOn | camOn |
| camOff | camOff |
| MoveUp | camUp |
| MoveDown | camDown |
| MoveRight | camRight |
| moveLeft | camLeft |
| movementStop | camMovementStop |
| moveReset | camPositionReset |
| zoomMin | camZoomMin |
| zoomMax | camZoomMax |
| zoomStop | camZoomStop |

Camera(QWidget \*parent)

Constructeur, initialise un web serveur et un serveur connecte les slots nécessaires et définit l’écoute des serveurs.

onSocketConnected()

Informe que le socket est bien connecté

newWebConnexion()

Effectue la connexion du client, et connecte les slots nécessaires

webSocketDisonnected()

Détruit la connexion du client puis déconnecte les slots

processTextMessage(QString message)

Récupère le message envoyé par le client, le traite et envoie une trame au MOXA en fonction du message traité

camOn()

Allume la caméra

camOff()

Eteint la caméra

camUp()

Fait monter la caméra

camDown()

Fait descendre la caméra

camLeft()

Fait tourner la caméra à gauche

camRight()

Fait tourner la caméra à droite

camPositionReset()

Fait retourner la caméra à sa position d’origine

camMovementStop()

Stop le mouvement de la caméra

camZoomMin ()

Fait dézoomer la caméra

camZoomMax()

Fait zoomer la caméra

camZoomStop()

Stop le zoom de la caméra

issue(QByteArray data)

Envoie au MOXA le message de type QByteArray passé en paramètre

CamMove :

\_\_construct()

Initialise la classe CamMove

camUp()

Envoie un message pour lever la caméra

camDown($socket)

Envoie un message pour baisser la caméra qui prend en paramètre le socket

camLeft($socket)

Envoie un message pour tourner la caméra à gauche qui prend en paramètre le socket

camRight($socket)

Envoie un message pour tourner la caméra à droite qui prend en paramètre le socket

camZoomP($socket)

Envoie un message pour zoomer qui prend en paramètre le socket

camZoomM($socket)

Envoie un message pour dézoomer qui prend en paramètre le socket

camStart()

Envoie un message pour démarrer la caméra

camStop()

Envoie un message pour éteindre la caméra

camReset()

Envoie un message pour remettre la caméra à zero

Cam :

\_\_construct($\_SQL)

Initialise la classe cam en passant l’objet PDO en paramètre

createCam($localisation, $ipAddress, $port)

Créer une nouvelle connexion à une caméra en récupérant les informations d’un formulaire

editCam($localisation, $ipAddress, $port)

Modifie les informations d’une caméra en récupérant les informations d’un formulaire

delCam($id)

Supprime la connexion à une caméra en passant l’ID de la caméra souhaitée en paramètre

getID()

Retourne l’id de la caméra

verifCam()

Vérifie quelle caméra est utilisée

User :

\_\_construct($\_SQL)

Initialise la classe User en passant l’objet PDO en paramètre

register($email, $password, $confirmPassword)

Compare les informations d’inscription passées en paramètre avec la DB et créer un compte si les informations sont nouvelles

login($email, $password)

Compare les informations d’inscription passées en paramètre avec la DB et connecte la personne si le compte est existant

isEditor($ID)

Vérifie si l’utilisateur est un éditeur

isSpectator($ID)

Vérifie si l’utilisateur est un spectateur