# ***Dossier Technique***

## *Campus Serre*

Di Sario Laurent IR

Nanta Jérémy EC

Raux Mathieu EC

## Revue finale

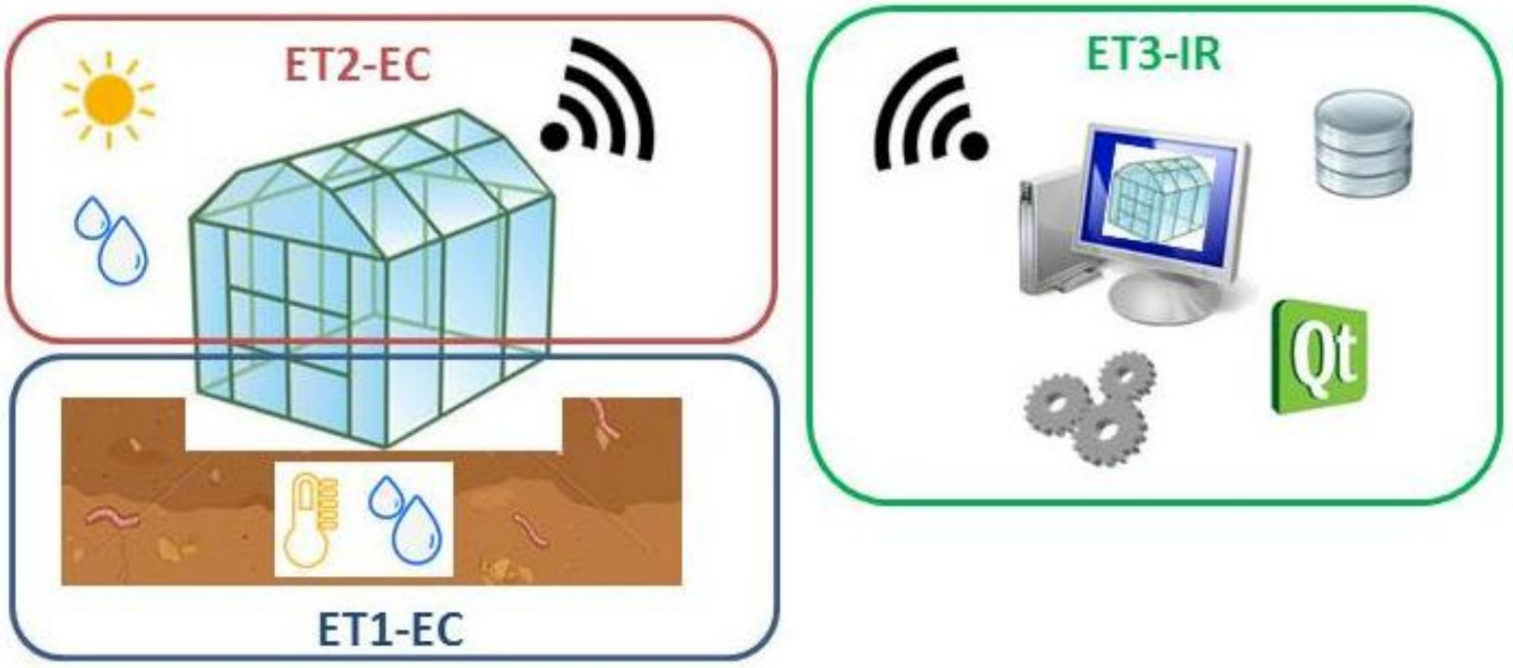
**version 1.0**

# Sommaire

# Présentation du Projet

Le Projet Campus Serre a pour but la gestion d'une serre à partir d'un ordinateur de supervision équipé d'un logiciel permettant la visualisation des informations sur les conditions de la serre (Température, Humidité) et le contrôle à distance d'appareils permettant de réguler ces conditions.

De nos jour où la concurrence est rude, il est important d'avoir une production et une gestion excellente de ses cultures. Avoir à sa disposition un logicielle permettant d'effectuer les points précédents est devenue presque indispensable.



**Ressources matérielles**

Lors de la création de ce projet, nous disposons de différents matériels :

- De deux cartes Xbee pro.

  Le XBee est un microcontrôleur sans fil fabriqué par Digi qui utilise un émetteur-récepteur sans fil 2,4 GHz pour communiquer avec un autre module XBee. Ces modules sont capables de communiquer avec plus d'un module XBee, ce qui signifie que vous pouvez créer un réseau de modules de partout, du moment qu'ils sont à portée. Il existe plusieurs catégories de modules. Le XBee standard a une puissance d'émission de 1mW avec une portée de 10 à 100 mètres (série 1 et 2) et le XBee Pro dispose d'une puissance d'émission de 60 mW avec une portée pouvant aller jusqu'à 1000 mètres.

- D'un PC pour le développement du programme, sous GNU/Linux Ubuntu 12.04 LTS

**Objectifs**

- Réception des données.

- Affichage des données.

- Réglage des consignes et des seuils.

- Lancer des actions sur des appareils.

# Répartition des tâches

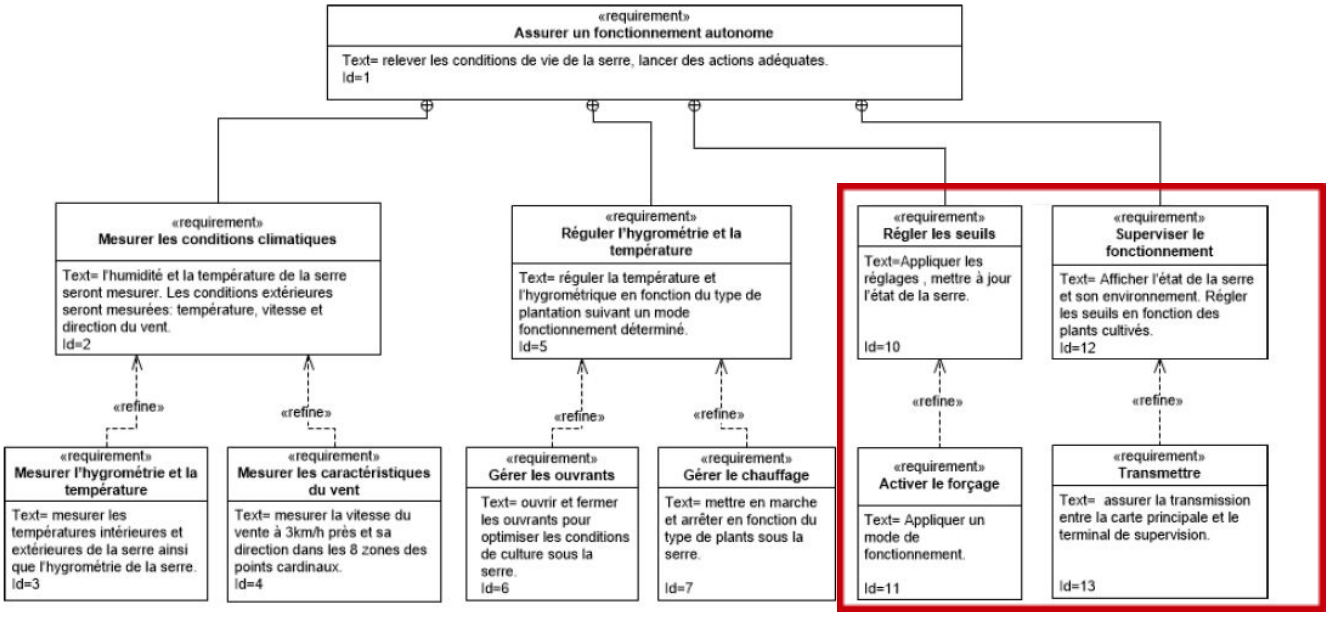
**<u>Di Sario Laurent (IR)</u> : Création de l'IHM ,réception des données, gestion de la base de Données, paramétrage des cultures (seuils et consignes), action sur les différents appareils.**

<u>Nanta Jérémy (EC)</u> : Gestion des capteurs température et hygrométrie air , Gestion de la girouette et de l'anémomètre, gestion du vérin.

<u>Raux Mathieu (EC)</u> : Gestion des capteurs température et hygrométrie sol ,gestion des actionneur et pré-actionneur arrosage et chauffage et de la communication Xbee.

# Diagrammes

## Exigences



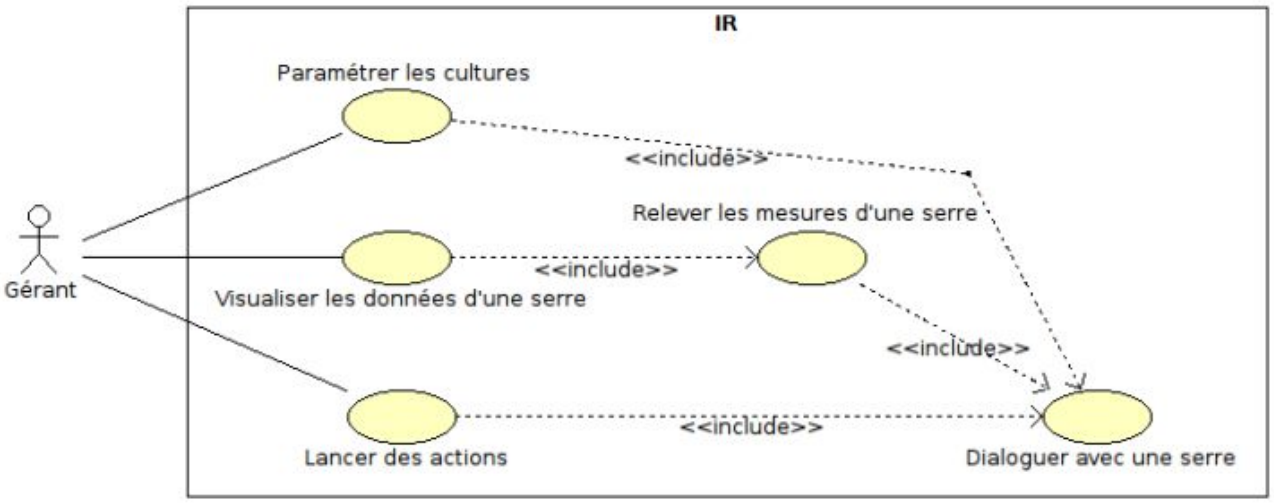

Transmettre : L'application doit être capable de recevoir les données envoyées par la serre ainsi que d'en envoyer.

Superviser le fonctionnement : Afficher les données de l'état de la serre.

Activer le forçage : Activer manuellement un ou plusieur appareils.

Régler les seuils : Envoyer les nouveaux réglage à la serre.

## Les cas d'utilisation




L'interface Homme-Machine est divisée en 4 parties :

- La visualisation des informations sur la serre dans l'onglet « Télémétrie »
- Le paramétrage des seuils et des consignes dans l'onglet « Culture »
- Le contrôle manuel des différents appareils dans l'onglet « Actions »
- La visualisation de l'historique de toutes les informations dans l'onglet « Historique »

Le gérant peut visualiser en continu les données de la serre, régler les différents seuils ou consignes, lancer des actions aux appareils sur une ou plusieurs cultures.

Les données visualisées sont :

- La température du sol et de l'air (°C)
- L'hygrométrie du sol et de l'air (%)
- La vitesse du vent (Km/h)
- La direction du vent (°)
- L'ouvrant (pourcentage d'ouverture)
- Le chauffage (éteint/allumé)
- La vanne (ouvert/fermé)

Les seuils sont des valeurs limites que la serre ne doit pas atteindre , si c'est le cas la serre renvoi une erreur.

Les seuils paramétrables sont :

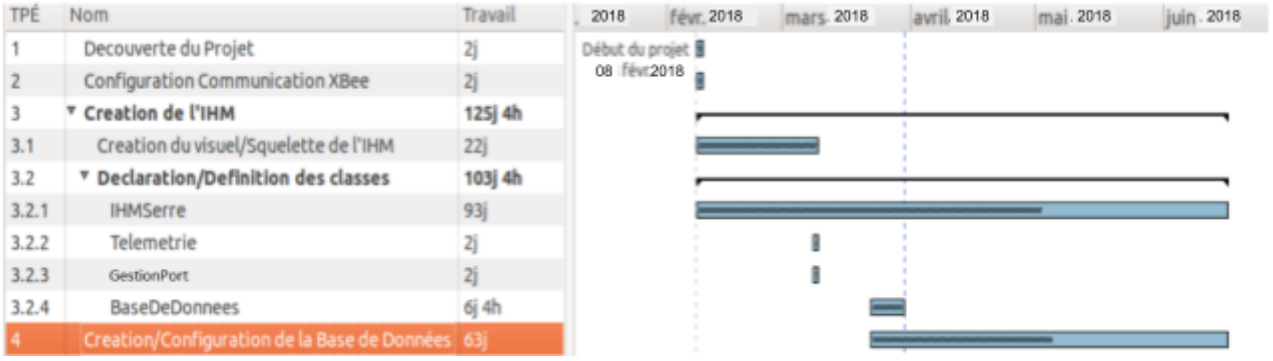
- La température maximum.
- La température minimum.
- La vitesse du vent.

Les consignes sont des valeurs qui servent à la régulation de la serre.

Les consignes paramétrables sont :

- La température minimum du jour et de la nuit.
- La température maximum du jour et de la nuit.
- L'hygrométrie minimum du jour et de la nuit.
- L'hygrométrie maximum du jour et de la nuit

## Gantt



| TPÉ | Nom | Travail |
|---|---|---|
| 1 | Decouverte du Projet | 2j |
| 2 | Configuration Communication XBee | 2j |
| 3 | **Creation de l'IHM** | **125j 4h** |
| 3.1 | Creation du visuel/Squelette de l'IHM | 22j |
| 3.2 | **Declaration/Definition des classes** | **103j 4h** |
| 3.2.1 | IHMSerre | 93j |
| 3.2.2 | Telemetrie | 2j |
| 3.2.3 | GestionPort | 2j |
| 3.2.4 | BaseDeDonnees | 6j 4h |
| 4 | Creation/Configuration de la Base de Données | 63j |



Nous pouvons voir la planification et le déroulement des différentes tâches durant le projet.

# Ressources

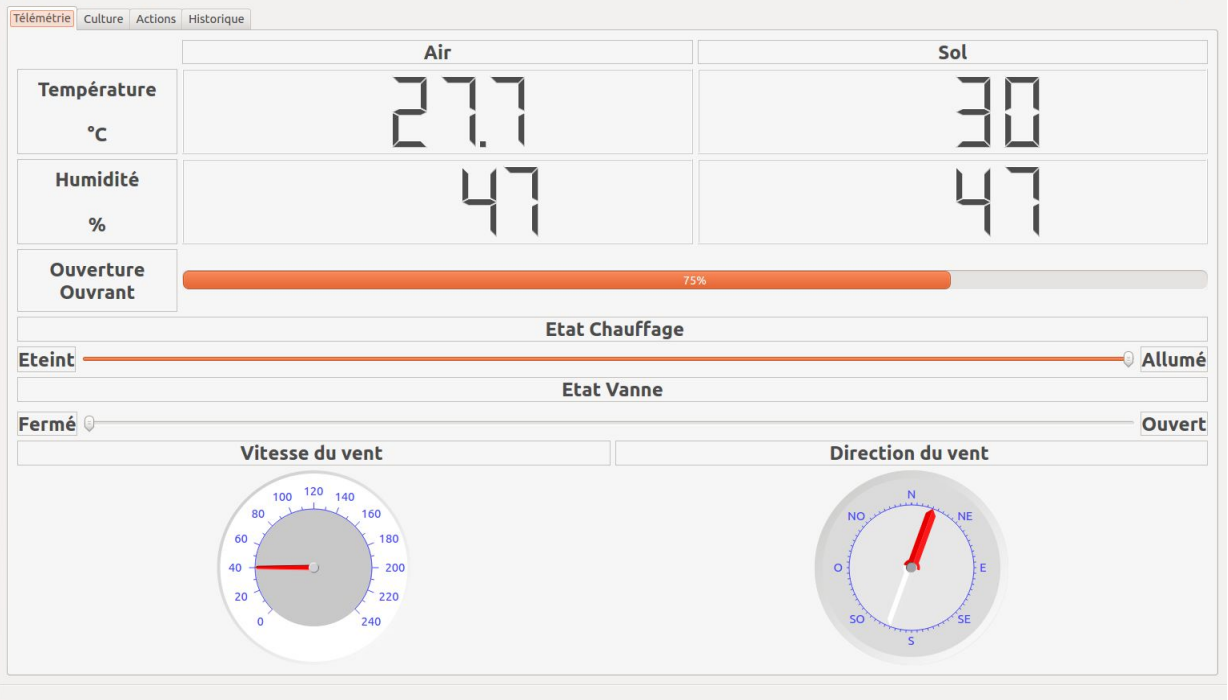
Les ressources matérielles

| Ressource | Description |
|---|---|
| Serre de jardin polycarbonate | - De 5,7m² , structure en aluminium, panneaux en |
| Electrovannes | - 2 électrovannes 24V HUNTER Pgv100mmb |
| Tableau électrique | - 1 disjoncteur 2x20A, une PC 16A<br>- 3 interrupteurs statiques 3.3V<br>- 240V AC<br>- Alimentation 230V<br>- 24Vcc<br>- Un voyant 240V |
| Carte de développement SAM4S Xplained | - Carte ATMEL SAM4S ou Carte ATMEL SAM7 ou Carte |

Les ressources logicielles

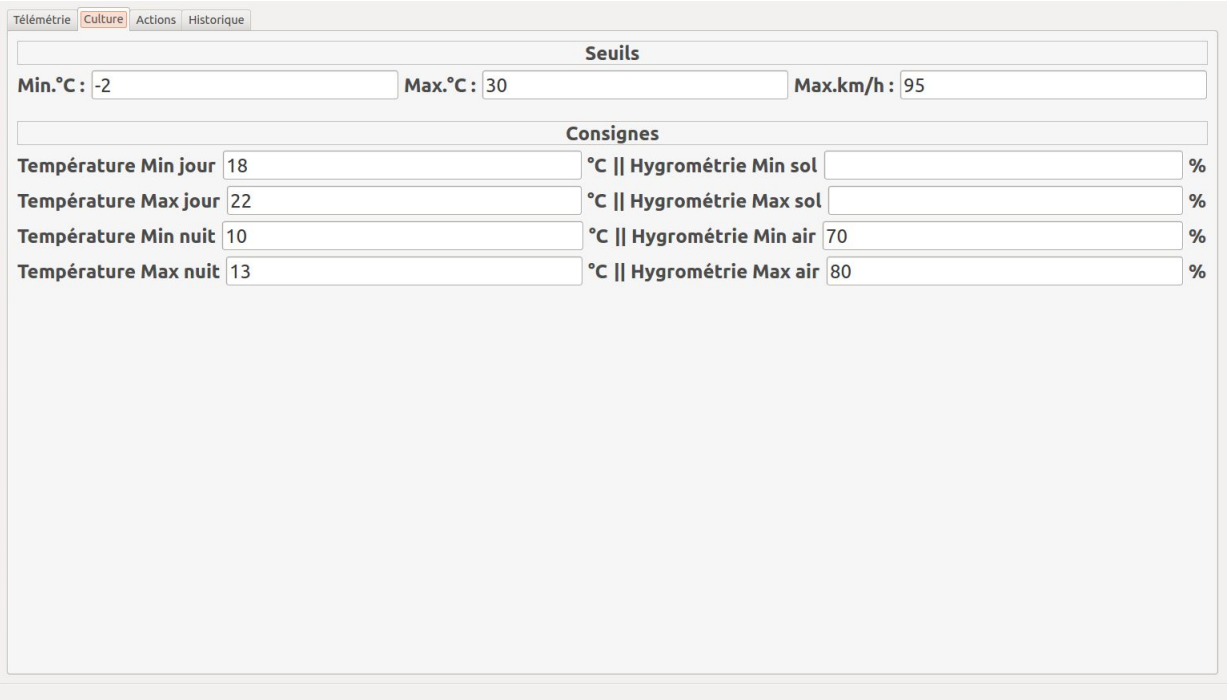
| Ressource | Version |
|---|---|
| OS | GNU Linux (Ubuntu 12.04.5 LTS) |
| EDI | Qt Creator 2.4.1 |
| Compilateur | GNU g++/gcc version 4.6.3 |
| Débuguer | GNU gdb 7.4 |
| Fabrication | QMake 2.01a et GNU make 3.81 |
| API GUI | Qt 4.8.1 |
| UML | bouml 7.4 |
| Versions | subversion (client svn 1.6.17) |
| Documentation | Doxygen version 1.7.6.1 |
| Gantt | Planner (version 0.14.5) |

# IHM Campus Serre

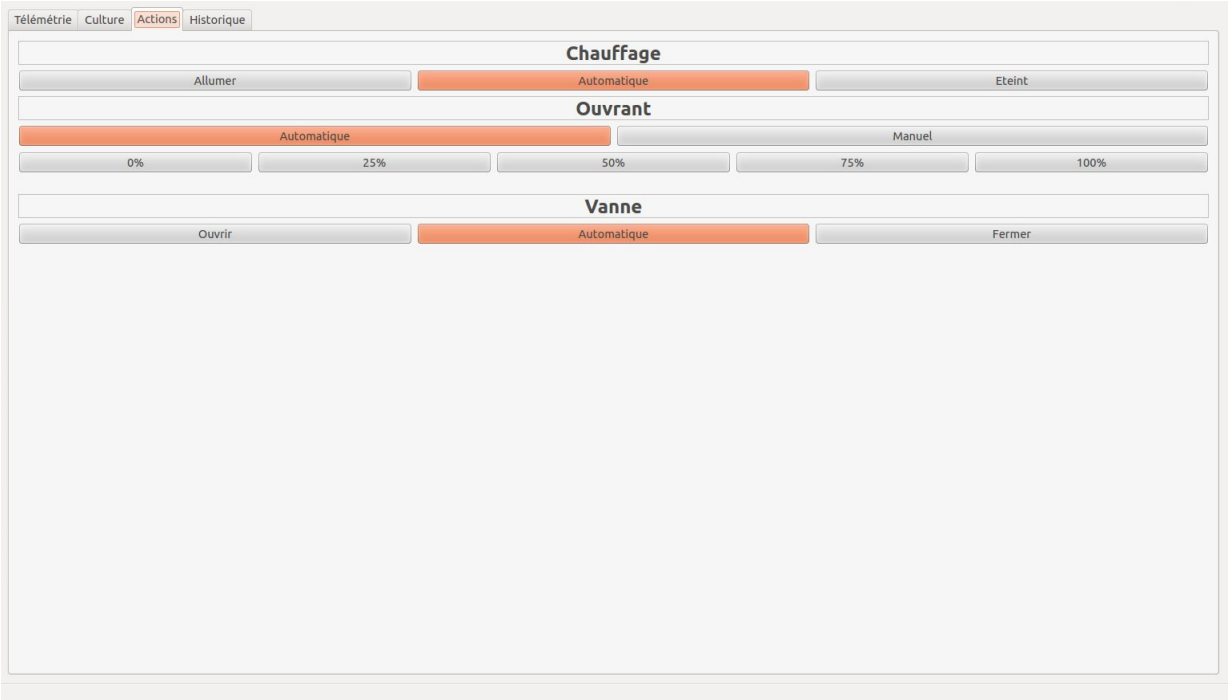



Dans l'onglet « Télémétrie », nous pouvons voir afficher :

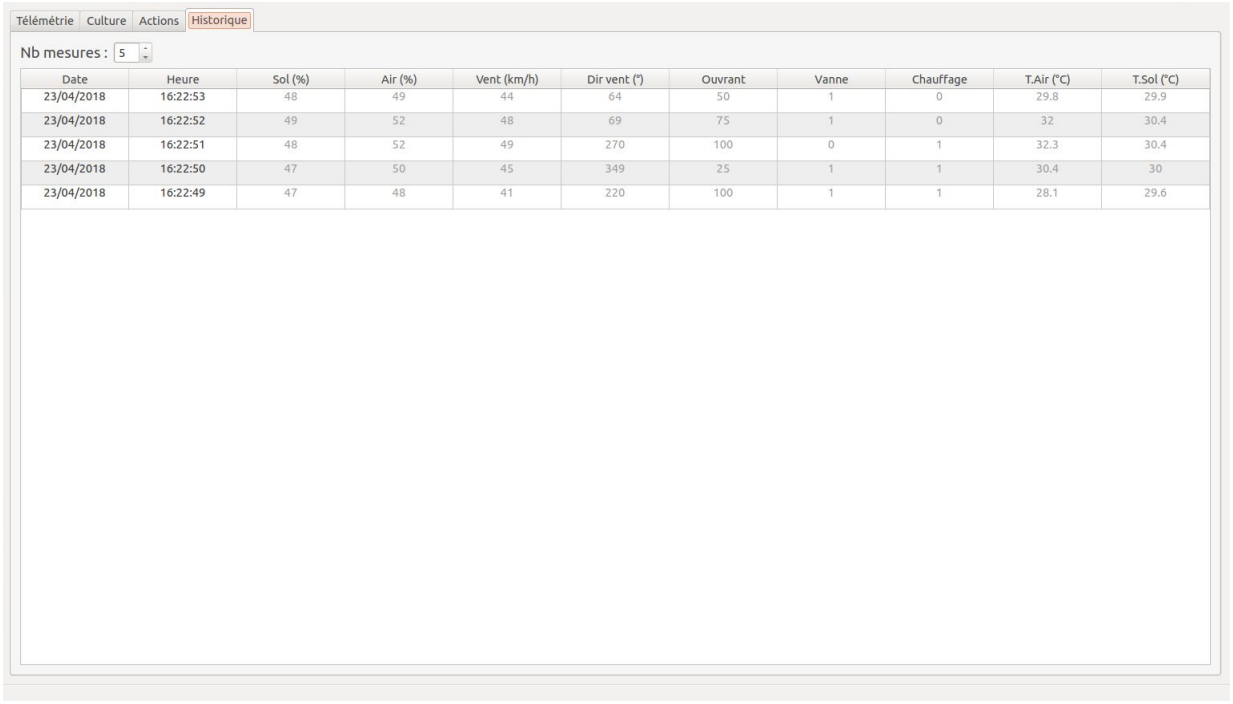
- La température de l’air
- La température du sol
- L’hygrométrie de l’air
- L’hygrométrie du sol
- L’état de l’ouverture de l’ouvrant en pourcentage
- L'état du chauffage
- L'état de la vanne
- La vitesse du vent en km/h
- La direction du vent




Dans l'onglet « Culture », il est possible de régler les seuils tolérés pour la vitesse du vent ainsi que la plage demandé pour la température. Pour le réglage des consignes, il est possible de configurer la température minimale et maximale du jour ainsi que de la nuit, l'hygrométrie minimale et maximale du sol ainsi que de l'air.



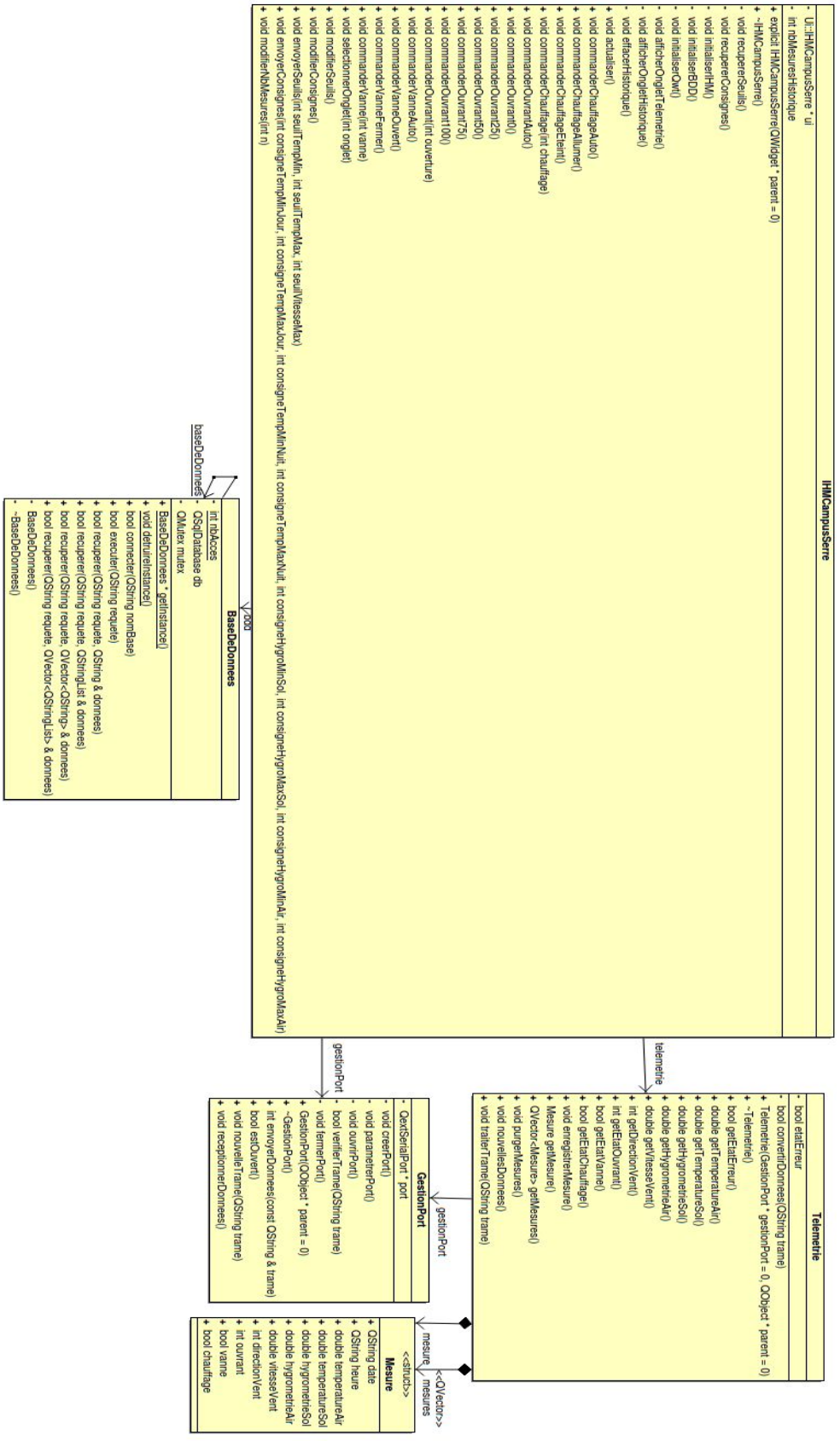

Dans l'onglet « actions », il est possible de piloter manuellement les différents appareils dans la serre (Chauffage, ouvrant ou vanne).




| Date | Heure | Sol (%) | Air (%) | Vent (km/h) | Dir vent (°) | Ouvrant | Vanne | Chauffage | T.Air (°C) | T.Sol (°C) |
|---|---|---|---|---|---|---|---|---|---|---|
| 23/04/2018 | 16:22:53 | 48 | 49 | 44 | 64 | 50 | 1 | 0 | 29.8 | 29.9 |
| 23/04/2018 | 16:22:52 | 49 | 52 | 48 | 69 | 75 | 1 | 0 | 32 | 30.4 |
| 23/04/2018 | 16:22:51 | 48 | 52 | 49 | 270 | 100 | 0 | 1 | 32.3 | 30.4 |
| 23/04/2018 | 16:22:50 | 47 | 50 | 45 | 349 | 25 | 1 | 1 | 30.4 | 30 |
| 23/04/2018 | 16:22:49 | 47 | 48 | 41 | 220 | 100 | 1 | 1 | 28.1 | 29.6 |

Dans l’onglet « Historique », il est possible de voir les dernières  informations transmise par la serre (Ici 5 mesures sont affichées mais il est possible d'en afficher plus).

# Diagramme de Classes



L'application est composée des classes suivantes :

IHMCampusSerre : Assure la visualisation des données de la serre, le paramétrage des cultures ainsi que le lancement des actions

BaseDeDonnees : Assure l'enregistrement des seuils ainsi que des consignes

Telemetrie: Permet de relever les mesures de la serre

GestionPort: assure le dialogue avec la serre

Avec ce diagramme de classe, on voit les principales classes du projet.

La classe **`GestionPort`** est la classe gérant la réception et l'envoi des données grâce à une carte Xbee, la classe **`Telemetrie`** fait le traitement de données en provenance de la classe **`GestionPort`**.

La classe **`IHMCampusSerre`** est la classe gérant l'interface de l'application. La plupart de ses attributs sont donc des widgets. La classe peut afficher la « télémétrie » sur l'interface.

**`Mesure`** est un conteneur qui permet de stocker les valeurs qui seront affichés dans l'onglet Historique de l'application.

# Protocole de communication

La serre et l'application doivent pouvoir communiquer ensemble pour cela nous avons établi un protocole de communication, il y a en tout 5 trames différentes. Les données sont codées en ASCII sur 3 chiffres :

T -> température air (°C)

S -> température sol (°C)

H -> hygrométrie sol (%)

A -> hygrométrie air (%)

V -> vitesse vent (Km/h)

D -> direction vent (°)

X -> ouvrant (pourcentage ouverture)

N -> vanne (ouvert/fermé)

C -> chauffage (allumé/éteint)

E -> erreur (

TMIN -> Température minimum (seuils)

TMAX -> Température maximum (seuils)

VMAX -> Vitesse maximum (seuils)

TMINJ -> Température minimum jour (consignes)

TMAXJ -> Température maximum jour (consignes)

TMINN -> Température minimum nuit (consignes)

TMAXN -> Température maximum nuit (consignes)

HMINS -> Hygrométrie minimum Sol (consignes)

HMAXS -> Hygrométrie maximum Sol (consignes)

HMINA -> Hygrométrie minimum Sol (consignes)

HMAXA -> Hygrométrie maximum Sol (consignes)

Une trame allant de la serre à l'application :

$PCS.EEE.TTT.SSS.HHH.AAA.VVV.DDD.XXX.NNN.CCC.*

$PCS.000.235.220.800.600.020.001.025.000.001.*

Le reste des trames sont des trames de l'application vers la serre :

$PCS.1.CCC.* (Trame pour commander le chauffage)

$PCS.1.001.*

$PCS.2.XXX.* (Trame pour commander l'ouvrant)

$PCS.2.075.*

$PCS.3.NNN.* (Trame pour commander la vanne)

$PCS.3.001.*

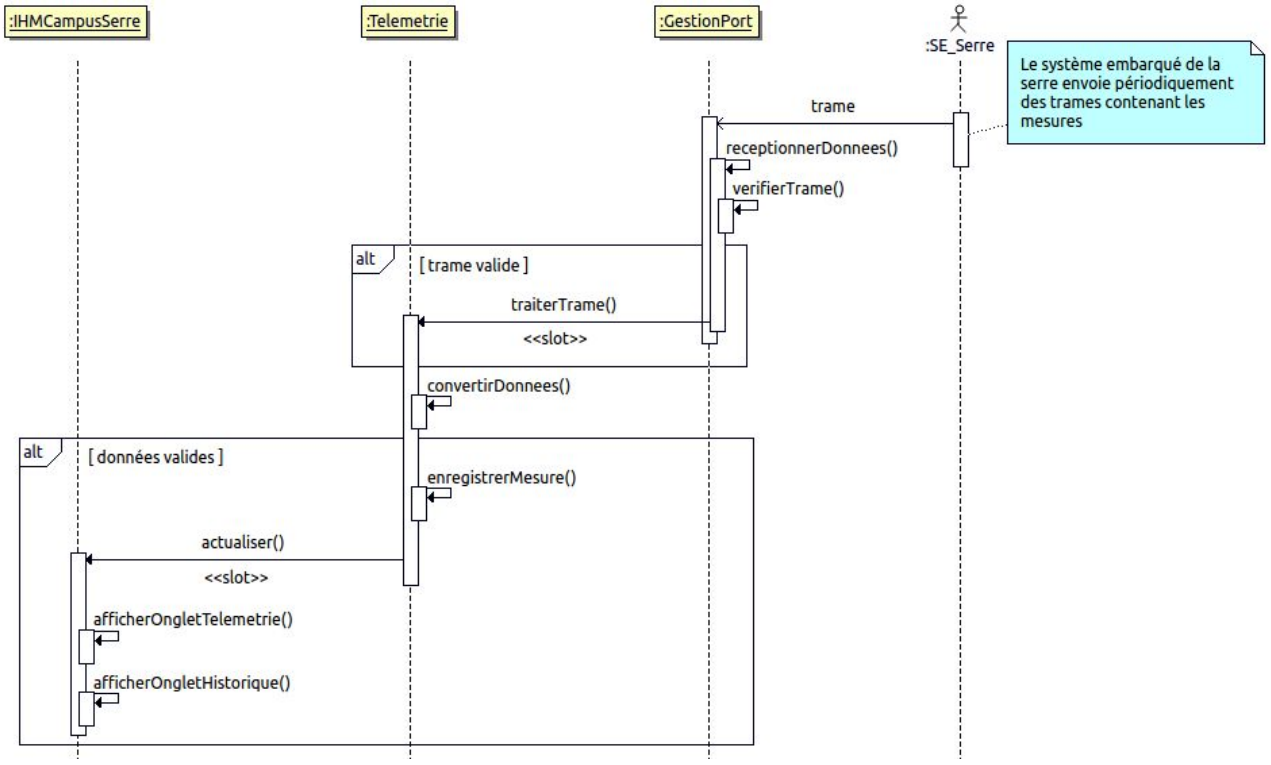
$PCS.4.TMIN.TMAX.VMAX.* (Trame pour envoyer les seuils)

$PCS.4.-02.030.095.*

$PCS.5.TMINJ.TMAXJ.TMINN.TMAXN.HMINS.HMAXS.HMINA.HMAXA.* (Trame pour envoyer les consignes)

$PCS.5.018.022.10.13.000.000.070.080.*

# Cas d'utilisation : relever d'une mesure




L'application attend la réception d'une nouvelle trame, la méthode **`receptionnerDonnees()`** est appelée. La trame est ensuite vérifiée grâce à **`verifierTrame()`**. Si la trame est une trame valide (répond a certain critère voir annexe) elle est convertie grâce à **`convertirDonnees()`** puis stocker dans un conteneur. Les données sont ensuite afficher dans l'IHM ainsi que dans l'historique grâce au conteneur.

```
bool GestionPort::verifierTrame(QString trame)
{
    QString checksum;
    const QString debutTrame = "$";
    const QString typeTrame = "PCS";
    const QString debutChecksum = "*";

/* ...*/
    // phrase vide ?
    if(trame.length() != 0)
    {
        // est-ce une phrase PCS ?
        if(trame.startsWith(debutTrame))
        {
            // est-ce la bonne phrase ?
            if(trame.startsWith(debutTrame + typeTrame))
            {
                // y-a-t-il un checksum ?
                if(trame.contains(debutChecksum))
                {
                    checksum = trame.section(debutChecksum, 1, 1);
                    qDebug() << Q_FUNC_INFO << "checksum : 0x" << checksum;
                    return 1;
                }
                else
                    qDebug() << Q_FUNC_INFO << "Attention : il n'y a pas de checksum dans cette phrase !";
                    return 0;
            }
            else
                qDebug() << Q_FUNC_INFO << "Erreur : ce n'est pas une trame PCS !";
                return 0;
        }
        else
            qDebug() << Q_FUNC_INFO << "Erreur : ce n'est pas une trame PCS !";
            return 0;
    }
    else
        qDebug() << Q_FUNC_INFO << "Erreur : phrase vide !";
        return 0;
}
```

Nous pouvons voir ici la méthode **`verifierTrame()`**, on récupère la trame pour s'assurer qu'il s'agit bien de la bonne trame (dans notre cas, la trame est appelé "PCS").

```
void Telemetrie::enregistrerMesure()
{
    Mesure mesure;
    QDateTime horodatage = QDateTime::currentDateTime();

    mesure.date = horodatage.toString("dd/MM/yyyy");
    mesure.heure = horodatage.toString("hh:mm:ss");

    mesure.temperatureAir = getTemperatureAir();
    mesure.temperatureSol = getTemperatureSol();
    mesure.hygrometrieAir = getHygrometrieAir();
    mesure.hygrometrieSol = getHygrometrieSol();
    mesure.vitesseVent = getVitesseVent();
    mesure.directionVent = getDirectionVent();
    mesure.ouvrant = getEtatOuvrant();
    mesure.vanne = getEtatVanne();
    mesure.chauffage = getEtatChauffage();

    mesures.push_back(mesure);
}
```

La méthode **`enregistrerMesure()`** permet de stocker dans un conteneur toutes les mesures ainsi que la date et l'heure à la sorti de la méthode **`convertirDonnees()`**.

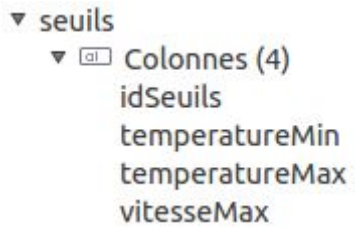
# Base de Données (SQLite)

SQLite est une bibliothèque écrite en langage C qui propose un moteur de base de données relationnelle accessible par le langage SQL.

Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

La base de données contient 4 tables , une pour chaque type d'information stockées (Consignes, Seuils, Cultures, Mesures)

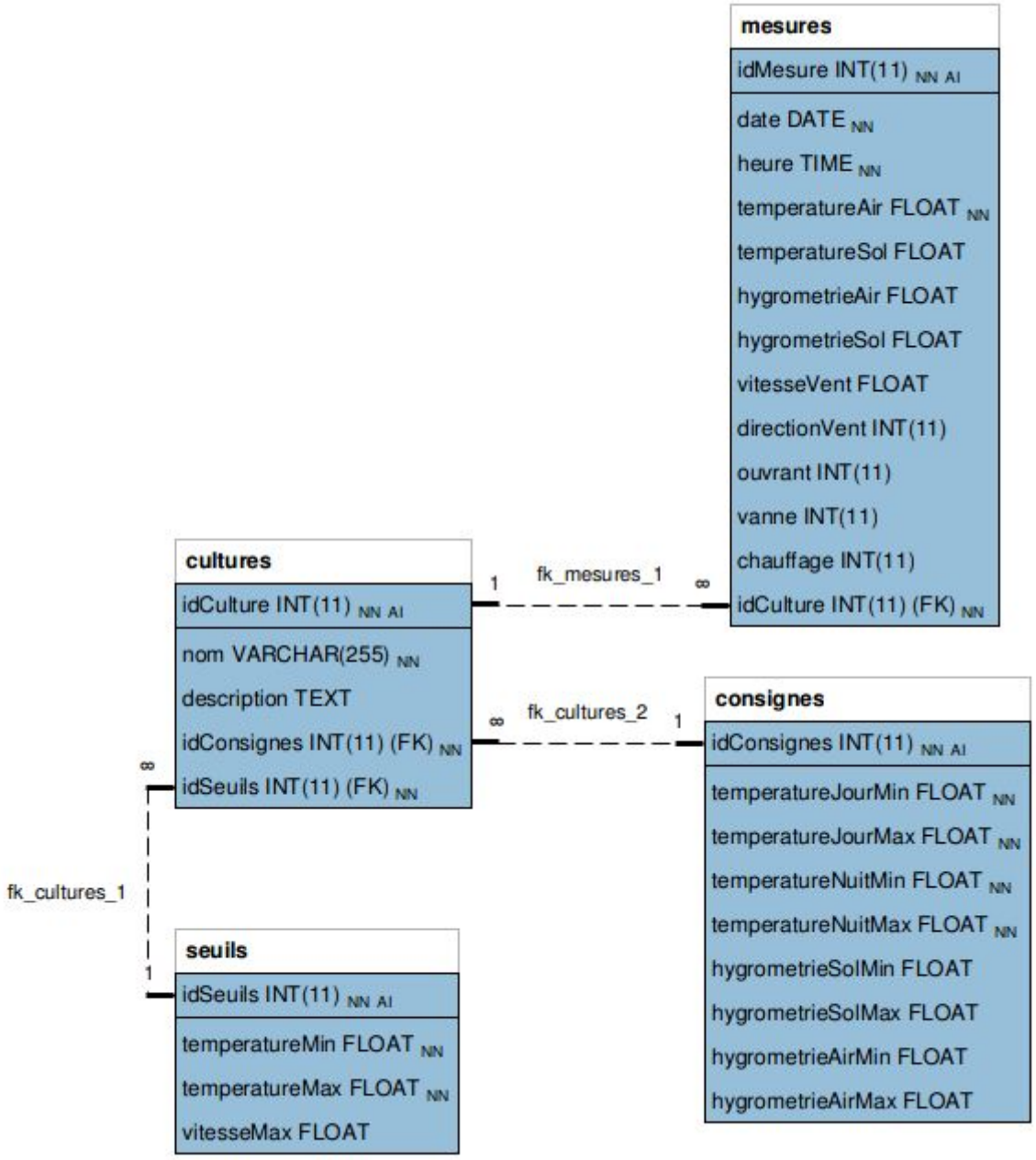
Nous pouvons voir le contenu de la table des seuils.




| | idSeuils | temperatureMin | temperatureMax | vitesseMax |
|---|---|---|---|---|
| 1 | 1 | -2 | 30 | 95 |

```
void IHMCampusSerre::recupererSeuils()
{
    QStringList seuils;
    QString requete = "select * from seuils";
    bool retour = bdd->recuperer(requete, seuils);
    if(retour)
    {
        ui->EditTempMin->setText(seuils.at(1));
        ui->EditTempMax->setText(seuils.at(2));
        ui->EditVitMax->setText(seuils.at(3));
    }
}
```

La méthode **`recupererSeuils()`** utilise une requête SQL pour récupérer toute les informations au lancement de l'application de la tables seuils pour ensuite les afficher dans l'onglet culture.


mesures
idMesure INT(11) NN AI
date DATE NN
heure TIME NN
temperatureAir FLOAT NN
temperatureSol FLOAT
hygrometrieAir FLOAT
hygrometrieSol FLOAT
vitesseVent FLOAT
directionVent INT(11)
ouvrant INT(11)
vanne INT(11)
chauffage INT(11)
idCulture INT(11) (FK) NN
cultures
idCulture INT(11) NN AI
nom VARCHAR(255) NN
description TEXT
idConsignes INT(11) (FK) NN
idSeuils INT(11) (FK) NN
fk_mesures_1
1
∞
consignes
idConsignes INT(11) NN AI
temperatureJourMin FLOAT NN
temperatureJourMax FLOAT NN
temperatureNuitMin FLOAT NN
temperatureNuitMax FLOAT NN
hygrometrieSolMin FLOAT
hygrometrieSolMax FLOAT
hygrometrieAirMin FLOAT
hygrometrieAirMax FLOAT
fk_cultures_2
∞
1
fk_cultures_1
∞
1
seuils
idSeuils INT(11) NN AI
temperatureMin FLOAT NN
temperatureMax FLOAT NN
vitesseMax FLOAT

# Recette

| Description | O/N |
|---|---|
| La base de données est fonctionnelle et complétée | O |
| Le système est paramétrable | O |
| Les appareils sont pilotables | O |
| Les informations de la serre sont consultables | O |