Projet e-stok

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet

e-stock est un système de gestion de stock automatisé qui permettra :

- de contrôler et gérer l'utilisation de produits stockés dans une armoire sensible
- d'assurer la traçabilité de l'attribution du matériel et des consommables stockés
- de sécuriser l'accès par un contrôle d'accès par badge RFID

1.1 Table des matières

1.2 Informations

Auteur

Pierre-Antoine Legger pierreantoinelegger@gmail.com
Joffrey Tranchat joffrey.tranchat@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/e-stock

2 Changelog

r82 | palegger | 2020-04-02 05 :30 :03 +0200 (jeu. 02 avril 2020) | 1 ligne

correction diagramme

r81 | jtranchat | 2020-04-01 23 :57 :17 +0200 (mer. 01 avril 2020) | 1 ligne

ajout diagramme de classe scénario mettre à jour le stock

r80 | jtranchat | 2020-04-01 21 :47 :00 +0200 (mer. 01 avril 2020) | 1 ligne

ajout de la documentation doxygen pour la version 0.1

r79 | jtranchat | 2020-04-01 21 :16 :21 +0200 (mer. 01 avril 2020) | 1 ligne

création du tag 0.1

r78 | palegger | 2020-04-01 20 :13 :44 +0200 (mer. 01 avril 2020) | 1 ligne

ajout diagramme de sequence et de classe

r77 | palegger | 2020-03-31 19 :27 :41 +0200 (mar. 31 mars 2020) | 1 ligne

Amelioration de la gestion de plusieurs casiers

r76 | jtranchat | 2020-03-31 12 :04 :31 +0200 (mar. 31 mars 2020) | 1 ligne

ajout fonction envoyerRequetePoid

r75 | jtranchat | 2020-03-29 18 :40 :55 +0200 (dim. 29 mars 2020) | 1 ligne

modification diagramme mettre à jour le stock

r74 | jtranchat | 2020-03-28 17 :06 :48 +0100 (sam. 28 mars 2020) | 1 ligne

Ajout/modification de commentaire dans le projet

r73 | jtranchat | 2020-03-28 15 :18 :53 +0100 (sam. 28 mars 2020) | 1 ligne

ajout de commentaire dans la classe article

r72 | jtranchat | 2020-03-28 15 :03 :00 +0100 (sam. 28 mars 2020) | 1 ligne

ajout envoie requete trame poid au démarrage

r71 | jtranchat | 2020-03-28 14 :48 :25 +0100 (sam. 28 mars 2020) | 1 ligne

realisation todo dans traiterTramePoids()

r70 | tvaira | 2020-03-28 09 :39 :43 +0100 (sam. 28 mars 2020) | 1 ligne

Mise a jour Bouml

r69 | tvaira | 2020-03-28 09 :22 :52 +0100 (sam. 28 mars 2020) | 2 lignes

Ajout TODO pour l'itération 2

r68 | tvaira | 2020-03-28 08 :45 :30 +0100 (sam. 28 mars 2020) | 2 lignes

Révision de code

r67 | tvaira | 2020-03-28 08 :37 :15 +0100 (sam. 28 mars 2020) | 1 ligne

Modification BD

r66 | palegger | 2020-03-28 04 :48 :31 +0100 (sam. 28 mars 2020) | 1 ligne

Ajout fonctionnalite prise en charge un article dans plusieurs casiers

r65 | palegger | 2020-03-27 19 :17 :54 +0100 (ven. 27 mars 2020) | 1 ligne

resolution erreur liée a l'affichage

r64 | jtranchat | 2020-03-26 23 :56 :51 +0100 (jeu. 26 mars 2020) | 1 ligne

fonction traiter trame poids dans supervision + fonction compter() + fonction arrondie()

r63 | jtranchat | 2020-03-24 18 :09 :49 +0100 (mar. 24 mars 2020) | 1 ligne

definition fonction mettreAJourQuantite dans article

r62 | tvaira | 2020-03-22 16 :29 :34 +0100 (dim. 22 mars 2020) | 1 ligne

Modification exemples requêtes SQL

r61 | tvaira | 2020-03-22 16 :29 :03 +0100 (dim. 22 mars 2020) | 2 lignes

Révision du code pour Article et Supervision

r60 | tvaira | 2020-03-22 15 :34 :38 +0100 (dim. 22 mars 2020) | 1 ligne

Modification v0.3 SQL de la base de données

r59 | tvaira | 2020-03-22 15 :33 :48 +0100 (dim. 22 mars 2020) | 2 lignes

Ajout de la classe Armoire

r58 | tvaira | 2020-03-22 09 :11 :24 +0100 (dim. 22 mars 2020) | 2 lignes

Révision de code de la classe Communication

r57 | jtranchat | 2020-03-22 00 :36 :09 +0100 (dim. 22 mars 2020) | 1 ligne

ajout diagramme de mise à jour du stock

r56 | palegger | 2020-03-21 20 :50 :21 +0100 (sam. 21 mars 2020) | 1 ligne

Ajout Diagrame de séquence pour authentification par champs

r55 | palegger | 2020-03-21 20 :37 :57 +0100 (sam. 21 mars 2020) | 1 ligne

Ajout Diagrame de séquence pour authentification par champs

r54 | tvaira | 2020-03-21 18 :07 :02 +0100 (sam. 21 mars 2020) | 1 ligne

Validation des diagrammes de sequence

r53 | palegger | 2020-03-21 04 :52 :49 +0100 (sam. 21 mars 2020) | 1 ligne

Modification diagrame de séquence connexion par badge et ajout diagrame connexion par champs

r52 | jtranchat | 2020-03-20 06 :00 :57 +0100 (ven. 20 mars 2020) | 1 ligne

ajout de commentaire de la classe article plus ajout de commentaire dans Supervision

r51 | tvaira | 2020-03-19 16 :10 :41 +0100 (jeu. 19 mars 2020) | 1 ligne

Ajout du fichier SQL de base

r50 | palegger | 2020-03-19 03 :18 :49 +0100 (jeu. 19 mars 2020) | 1 ligne

Ajout de commentaires et de laisser vide si pas de mots de passe

r49 | tvaira | 2020-03-17 12 :06 :14 +0100 (mar. 17 mars 2020) | 2 lignes

Validation bouton Se déconnecter

r48 | tvaira | 2020-03-17 11 :54 :05 +0100 (mar. 17 mars 2020) | 2 lignes

Validation de crypterMotDepasse() (cf. define CHANGE_PASSWORD_BEFORE)

r47 | tvaira | 2020-03-17 11 :24 :31 +0100 (mar. 17 mars 2020) | 2 lignes

Exemple recherche articles pour FenetreMenu

r46 | tvaira | 2020-03-17 10 :36 :28 +0100 (mar. 17 mars 2020) | 2 lignes

Révision de code

r45 | jtranchat | 2020-03-13 16 :01 :20 +0100 (ven. 13 mars 2020) | 1 ligne

supression ROADBOOK et mise à jour du todo

r44 | palegger | 2020-03-13 15 :38 :50 +0100 (ven. 13 mars 2020) | 1 ligne

Liaison avec Esp effectuer

r43 | palegger | 2020-03-13 14 :00 :32 +0100 (ven. 13 mars 2020) | 1 ligne

Ajout page stock Ihm

r42 | jtranchat | 2020-03-13 10 :30 :28 +0100 (ven. 13 mars 2020) | 1 ligne

ajout fonction berifierTypeTrame

r41 | jtranchat | 2020-03-13 10 :27 :47 +0100 (ven. 13 mars 2020) | 1 ligne

correction bug comptage automatique

r40 | jtranchat | 2020-03-12 15 :33 :28 +0100 (jeu. 12 mars 2020) | 1 ligne

msie en place traiter trame

r39 | jtranchat | 2020-03-12 12 :32 :06 +0100 (jeu. 12 mars 2020) | 1 ligne

mise en place prendre et rapporter article automatique

r38 | jtranchat | 2020-03-12 11 :47 :15 +0100 (jeu. 12 mars 2020) | 1 ligne

mise en place du comptage automatique

r37 | palegger | 2020-03-12 10 :48 :23 +0100 (jeu. 12 mars 2020) | 1 ligne

Creation méthode de la classe Communication

r36 | palegger | 2020-03-12 10 :40 :05 +0100 (jeu. 12 mars 2020) | 1 ligne

Mise a jour Ihm

r35 | palegger | 2020-03-12 10 :01 :13 +0100 (jeu. 12 mars 2020) | 2 lignes

Mise a jour ROADBOOK

r34 | palegger | 2020-03-11 11 :57 :51 +0100 (mer. 11 mars 2020) | 1 ligne

Ajout crytage mot de passe

r33 | palegger | 2020-03-11 10 :47 :57 +0100 (mer. 11 mars 2020) | 1 ligne

Ajout verification de la verification de la date de validite

r32 | jtranchat | 2020-03-11 10 :44 :39 +0100 (mer. 11 mars 2020) | 1 ligne

ajout de la classe Article

r31 | tvaira | 2020-03-07 09 :47 :19 +0100 (sam. 07 mars 2020) | 2 lignes

Révision du code

r30 | jtranchat | 2020-03-06 17 :01 :26 +0100 (ven. 06 mars 2020) | 1 ligne

mise à jour ROADBOOK

r29 | jtranchat | 2020-03-06 16 :36 :20 +0100 (ven. 06 mars 2020) | 1 ligne

ajout de la fonction ajouter article

r28 | palegger | 2020-03-06 16 :10 :24 +0100 (ven. 06 mars 2020) | 1 ligne

Correction orthographe, Ajout fonction, Ajout fichier bouml avec diagramme de classe

r27 | jtranchat | 2020-03-06 11 :37 :55 +0100 (ven. 06 mars 2020) | 1 ligne

r26 | jtranchat | 2020-03-06 10 :35 :58 +0100 (ven. 06 mars 2020) | 1 ligne

ajout de m'essage d'erreur dans ajouter article

r25 | palegger | 2020-03-06 10 :22 :50 +0100 (ven. 06 mars 2020) | 1 ligne

Ajout connexion RFID

r24 | jtranchat | 2020-03-05 15 :54 :22 +0100 (jeu. 05 mars 2020) | 1 ligne

ajout page prendre et rapporter artiv

r23 | palegger | 2020-03-05 12 :37 :45 +0100 (jeu. 05 mars 2020) | 1 ligne

Ajustement pour connexion identifiant

r22 | palegger | 2020-03-05 12 :29 :45 +0100 (jeu. 05 mars 2020) | 1 ligne

Connexion par identifiant fonctionnel

r21 | palegger | 2020-03-05 10 :47 :21 +0100 (jeu. 05 mars 2020) | 1 ligne

Connection bouton Se Connecter

r20 | jtranchat | 2020-03-05 10 :45 :31 +0100 (jeu. 05 mars 2020) | 1 ligne

correction bug

r19 | jtranchat | 2020-03-05 10 :28 :50 +0100 (jeu. 05 mars 2020) | 1 ligne

sa marche

r18 | jtranchat | 2020-03-05 00 :23 :18 +0100 (jeu. 05 mars 2020) | 1 ligne

ajout page menu principal et prendre ou rajouter un objet

r17 | jtranchat | 2020-03-05 00 :03 :05 +0100 (jeu. 05 mars 2020) | 1 ligne

mise a jour roadbook

r16 | palegger | 2020-03-04 16 :09 :22 +0100 (mer. 04 mars 2020) | 1 ligne

correction classe Utilisateur

r15 | palegger | 2020-03-04 13 :50 :55 +0100 (mer. 04 mars 2020) | 1 ligne

Ajout classe utilisateur

r14 | jtranchat | 2020-03-04 12 :32 :39 +0100 (mer. 04 mars 2020) | 1 ligne

modification pageAjouter

r13 | jtranchat | 2020-03-04 12 :16 :06 +0100 (mer. 04 mars 2020) | 1 ligne

ajout page pour ajuter un article

r12 | palegger | 2020-03-04 01 :34 :59 +0100 (mer. 04 mars 2020) | 1 ligne

Mise en place Ihm identifiant

r11 | palegger | 2020-03-03 22 :00 :30 +0100 (mar. 03 mars 2020) | 1 ligne

verification identifiant badge et affichage message d'erreur et passage a la fenetre identifiant

r10 | tvaira | 2020-02-15 14 :05 :26 +0100 (sam. 15 févr. 2020) | 3 lignes

Ajout des méthodes pour effecteur des requêtes SQL dans la classe Bdd Ajout des mécanismes Qt aux classes de l'application

r9 | jtranchat | 2020-02-14 11 :36 :16 +0100 (ven. 14 févr. 2020) | 1 ligne

ajout de commentaire

r8 | jtranchat | 2020-02-14 10 :31 :04 +0100 (ven. 14 févr. 2020) | 1 ligne

mise en place d'un singleton pour la classe Bdd

r7 | palegger | 2020-02-14 10 :23 :07 +0100 (ven. 14 févr. 2020) | 1 ligne

Lecture badge Rfid

r6 | jtranchat | 2020-02-14 09 :54 :13 +0100 (ven. 14 févr. 2020) | 1 ligne

ajout de la connection avec la Bdd

r5 | palegger | 2020-02-14 09 :01 :45 +0100 (ven. 14 févr. 2020) | 1 ligne

Ajout relation entre classes

r4 | jtranchat | 2020-02-13 15 :57 :31 +0100 (jeu. 13 févr. 2020) | 1 ligne

mise à jour ROADBOOK

r3 | jtranchat | 2020-02-13 12 :29 :48 +0100 (jeu. 13 févr. 2020) | 1 ligne

ajout des fichiers sources du projet

r2 | jtranchat | 2020-02-13 10 :34 :06 +0100 (jeu. 13 févr. 2020) | 1 ligne

ajout fichier ROADBOOK + TODO

r1 | www-data | 2020-02-01 15 :03 :10 +0100 (sam. 01 févr. 2020) | 1 ligne

Creating initial repository structure

3 README

3.1 Projet

3.1.1 Présentation

e-stock est un système de gestion de stock automatisé qui permettra :

- de contrôler et gérer l'utilisation de produits stockés dans une armoire sensible
- d'assurer la traçabilité de l'attribution du matériel et des consommables stockés
- de sécuriser l'accès par un contrôle d'accès par badge RFID

Une armoire sera composée de 8 casiers maximum. Chaque casier sera équipé :

- d'une gâche électrique afin d'assurer son ouverture/fermeture ;
- d'une balance pour assurer le comptage automatique des articles.

Le comptage automatique de la quantité est déterminé en fonction du poids unitaire et du poids mesuré sur la balance.

Un lecteur de badge RFID est intégré à chaque armoire pour contrôler l'accès. L'exploitation de l'armoire e-stock est possible à partir de l'écran tactile intégré.

On distinguera deux type d'articles :

- les « consommables » qui sortent définitivement du stock
- les « empruntables » qui peuvent être restitués après leur utilisation

3.1.2 Base de données MySQL

```
DROP DATABASE IF EXISTS `e-stock`;
CREATE DATABASE IF NOT EXISTS `e-stock`;
USE `e-stock`;

-- Création du compte d'accès à la base de données e-stock
-- CREATE USER 'estock'@'%' IDENTIFIED BY 'password';
-- GRANT ALL PRIVILEGES ON `e-stock`.* TO 'estock'@'%';
-- FLUSH PRIVILEGES;
-- --------------------------------------------------------

CREATE TABLE IF NOT EXISTS `Profil` (
  `idProfil` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  PRIMARY KEY (`idProfil`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Profil` (`Nom`) VALUES
('Administrateur'),
('Gestionnaire'),
('Utilisateur');

-- --------------------------------------------------------

CREATE TABLE IF NOT EXISTS `Groupe` (
  `idGroupe` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  PRIMARY KEY (`idGroupe`),
  CONSTRAINT Unique_Groupe UNIQUE (`Nom`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

INSERT INTO `Groupe` (`Nom`) VALUES
('PROFESSEUR'),
('1-BTS-SN'),
('T-BTS-SN');

-- --------------------------------------------------------

CREATE TABLE IF NOT EXISTS `Utilisateur` (
  `idUtilisateur` int(11) NOT NULL AUTO_INCREMENT,
  `idProfil` int(11) NOT NULL,
```

```
  `idGroupe` int(11) NOT NULL,
  `Nom` varchar(64) NOT NULL,
  `Prenom` varchar(64) NOT NULL,
  `DateValidite` date NOT NULL,
  `Identifiant` varchar(255) DEFAULT NULL,
  `MotDePasse` varchar(255) DEFAULT NULL,
  `Badge` varchar(11) NOT NULL,
  `Email` varchar(64) NOT NULL,
  PRIMARY KEY (`idUtilisateur`),
--  CONSTRAINT Unique_Utilisateur UNIQUE (`Badge`),
  CONSTRAINT Utilisateur_fk_1 FOREIGN KEY (`idProfil`) REFERENCES Profil(`idProfil`) ON DELETE CASCADE,
  CONSTRAINT Utilisateur_fk_2 FOREIGN KEY (`idGroupe`) REFERENCES Groupe(`idGroupe`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Armoire` (
  `idArmoire` int(11) NOT NULL,
  `Nom` varchar(255) NOT NULL,
  `Description` varchar(255) DEFAULT NULL,
  `nbCasiers` int(11) NOT NULL DEFAULT 8,
  PRIMARY KEY (`idArmoire`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Armoire` (`idArmoire`,`Nom`,`Description`,`nbCasiers`) VALUES('1','B22','Atelier','2');

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Type` (
  `idType` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  PRIMARY KEY (`idType`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Type` (`Nom`) VALUES
('Equipement'),
('Consommable');

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Unite` (
  `idUnite` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  PRIMARY KEY (`idUnite`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Unite` (`Nom`) VALUES
('Metre'),
('Piece'),
('Pourcentage'),
('Poids g'),
('Poids kg');

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Comptage` (
  `idComptage` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  PRIMARY KEY (`idComptage`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Comptage` (`Nom`) VALUES
('Aucun'),
('Automatique'),
('CodeBarre');

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Action` (
  `idAction` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  PRIMARY KEY (`idAction`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Action` (`Nom`) VALUES
('Entree'),
('Sortie');

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Article` (
  `idArticle` int(11) NOT NULL AUTO_INCREMENT,
  `idType` int(11) NOT NULL,
  --   `Type` enum('Equipement','Consommable'),
  `Nom` varchar(255) NOT NULL,
  `Code` varchar(255) NOT NULL,
  `Designation` varchar(255) NOT NULL,
  `Poids` int(11) NOT NULL,
  PRIMARY KEY (`idArticle`),
  CONSTRAINT Article_fk_1 FOREIGN KEY (`idType`) REFERENCES Type(`idType`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Stock` (
  `idStock` int(11) NOT NULL AUTO_INCREMENT,
  `idArticle` int(11) NOT NULL,
  `idComptage` int(11) NOT NULL,
  `idUnite` int(11) NOT NULL,
  `Quantite` int(11) DEFAULT 0,
  `Disponible` int(11) DEFAULT 0,
  `Tare` int(11) NOT NULL,
  `numeroCasier` int(11) NOT NULL,
  PRIMARY KEY (`idStock`),
  CONSTRAINT Unique_NumeroCasier UNIQUE (`numeroCasier`),
  CONSTRAINT Stock_fk_2 FOREIGN KEY (`idArticle`) REFERENCES Article(`idArticle`) ON DELETE CASCADE,
  CONSTRAINT Stock_fk_3 FOREIGN KEY (`idComptage`) REFERENCES Comptage(`idComptage`) ON DELETE CASCADE,
  CONSTRAINT Stock_fk_4 FOREIGN KEY (`idUnite`) REFERENCES Unite(`idUnite`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Mouvement` (
  `idMouvement` int(11) NOT NULL AUTO_INCREMENT,
  `idUtilisateur` int(11) NOT NULL,
  `idStock` int(11) NOT NULL,
  `idAction` int(11) NOT NULL,
  --    `Action` enum('Entree','Sortie'),
  `Quantite` int(11) NOT NULL,
  `Horodatage` datetime NOT NULL,
  PRIMARY KEY (`idMouvement`),
  CONSTRAINT Mouvement_fk_1 FOREIGN KEY (`idUtilisateur`) REFERENCES Utilisateur(`idUtilisateur`) ON DELETE
      CASCADE,
  CONSTRAINT Mouvement_fk_2 FOREIGN KEY (`idStock`) REFERENCES Stock(`idStock`) ON DELETE CASCADE,
  CONSTRAINT Mouvement_fk_3 FOREIGN KEY (`idAction`) REFERENCES Action(`idAction`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

3.1.3 Recette

- Pierre-Antoine Legger
  - S'authentifier
  - Rechercher un article
  - Consulter le stock
  - Communiquer avec le SE pour :
    - Commander l'ouverture/fermeture des casiers
    - Afficher l'état ouvert/fermé des casiers
- Joffrey Tranchat
  - Prendre et rapporter un article
  - Mettre à jour le stock et les mouvements
  - Consulter les mouvements
  - Communiquer avec le SE pour :
  - Récupérer les pesées des casiers
  - Assurer le comptage automatique

3.1.4 Informations

Auteur

Pierre-Antoine Legger pierreantoinelegger@gmail.com
Joffrey Tranchat joffrey.tranchat@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/e-stock

4 A propos

Auteur

Pierre-Antoine Legger pierreantoinelegger@gmail.com
Joffrey Tranchat joffrey.tranchat@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/e-stock

5 Licence GPL

6 Liste des choses à faire

Membre Casier : :Casier (int numero, QWidget ∗parent=0)

Définir une constante pour une taille minimum du casier dans l'IHM

Gérer les différentes couleurs de fond par rapport à l'état (ouvert/fermé, vide, ...)

Connecter signal/slot si nécessaire

Membre Casier : :gererEtat ()

Créer les méthode pour gérer un casier, notamment les signaux et les slots

Membre Casier : :ouvrir ()

Envoyer trame ouverture

7 Documentation des espaces de nommage

7.1 Référence de l'espace de nommage Ui

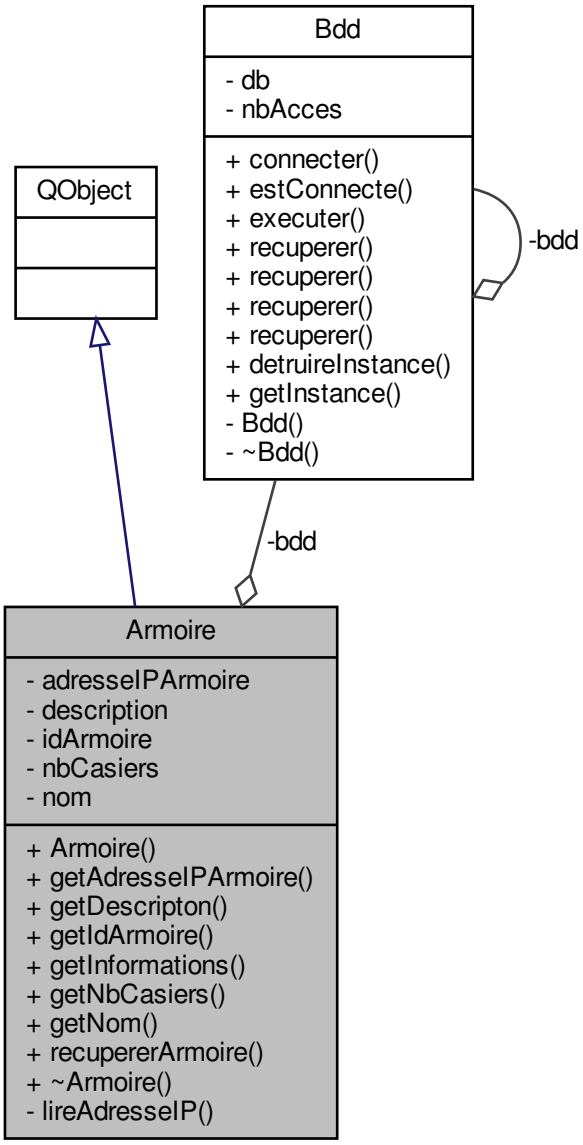
8 Documentation des classes

8.1 Référence de la classe Armoire

La classe Armoire traite les articles.

`#include <Armoire.h>`

Graphe de collaboration de Armoire :



Signaux

— void informationsArmoire (QStringList)

Fonctions membres publiques

— Armoire (QObject ∗parent=nullptr)
Définition du constructeur de la classe Armoire.
— QString getAdresseIPArmoire () const
Définition de la méthode getAdresseIPArmoire.
— QString getDescripton () const
Définition de la méthode getDescripton.
— QString getIdArmoire () const
Définition de la méthode getIdArmoire.
— QStringList getInformations ()
Définition de la méthode getInformations.
— QString getNbCasiers () const
Définition de la méthode getNbCasiers.
— QString getNom () const
Définition de la méthode getNom.
— void recupererArmoire (QString idArmoire="1")
Définition de la méthode recupererArmoire.
— ∼Armoire ()
Définition du destructeur de la classe Armoire.

Fonctions membres privées

— QString lireAdresseIP ()
Définition de la méthode lireAdresseIP.

Attributs privés

— QString adresseIPArmoire
l'adresse IP de la Raspberry Pi
— Bdd ∗ bdd
association d'un objet Bdd (accès à la base de données)
— QString description
la description de l'armoire
— QString idArmoire
l'id de l'armoire
— QString nbCasiers
le nombre de casiers dans l'armoire
— QString nom
le nom de l'armoire

8.1.1 Description détaillée

La classe Armoire traite les articles.

Auteur

Tranchat Joffrey

Version

1.0

Date

Dimanche 22 Mars 2020

Définition à la ligne 49 du fichier Armoire.h.

8.1.2 Documentation des constructeurs et destructeur

8.1.2.1 Armoire()

```
Armoire::Armoire (
            QObject * parent = nullptr )
```

Définition du constructeur de la classe Armoire.

Initialise un objet Armoire

Paramètres

parent	l'objet QObject parent

Définition à la ligne 22 du fichier Armoire.cpp.

Références adresseIPArmoire, bdd, Bdd : :getInstance(), lireAdresseIP(), et recupererArmoire().

```
00022                                : QObject(parent)
00023 {
00024     #ifdef DEBUG_ARMOIRE
00025         qDebug() << Q_FUNC_INFO;
00026     #endif
00027     bdd = Bdd::getInstance();
00028     adresseIPArmoire = lireAdresseIP();
00029     recupererArmoire();
00030 }
```

8.1.2.2 ~Armoire()

```
Armoire::~Armoire ( )
```

Définition du destructeur de la classe Armoire.

Detruit un objet Armoire

Définition à la ligne 36 du fichier Armoire.cpp.

Références Bdd : :detruireInstance().

```
00037 {
00038     Bdd::detruireInstance();
00039     #ifdef DEBUG_ARMOIRE
00040         qDebug() << Q_FUNC_INFO;
00041     #endif
00042 }
```

8.1.3 Documentation des fonctions membres

8.1.3.1 getAdresseIPArmoire()

```
QString Armoire::getAdresseIPArmoire ( ) const
```

Définition de la méthode getAdresseIPArmoire.

renvoie l'adresse IP de l'armoire

Renvoie

l'adresse IP de l'armoire

Définition à la ligne 138 du fichier Armoire.cpp.

Références adresseIPArmoire.

```
00139 {
00140     return adresseIPArmoire;
00141 }
```

8.1.3.2 getDescripton()

```
QString Armoire::getDescripton ( ) const
```

Définition de la méthode getDescripton.

renvoie la description de l'armoire

Renvoie

la description de l'armoire

Définition à la ligne 116 du fichier Armoire.cpp.

Références description.

```
00117 {
00118     return description;
00119 }
```

8.1.3.3 getIdArmoire()

```
QString Armoire::getIdArmoire ( ) const
```

Définition de la méthode getIdArmoire.

renvoie l'id de l'armoire

Renvoie

id de l'armoire

Définition à la ligne 94 du fichier Armoire.cpp.

Références idArmoire.

```
00095 {
00096     return idArmoire;
00097 }
```

8.1.3.4 getInformations()

```
QStringList Armoire::getInformations ( )
```

Définition de la méthode getInformations.

renvoie les informations de l'armoire

Renvoie

informations de l'armoire

Définition à la ligne 78 du fichier Armoire.cpp.

Références adresseIPArmoire, description, idArmoire, informationsArmoire(), nbCasiers, et nom.

Référencé par Supervision : :getInformationsArmoire().

```
00079 {
00080     QStringList informations;
00081
00082     informations << idArmoire << nom << description <<
      nbCasiers << adresseIPArmoire;
00083
00084     emit informationsArmoire(informations);
00085
00086     return informations;
00087 }
```

8.1.3.5 getNbCasiers()

```
QString Armoire::getNbCasiers ( ) const
```

Définition de la méthode getNbCasiers.

renvoie le nombre de casiers dans l'armoire

Renvoie

le nombre de casiers dans l'armoire

Définition à la ligne 127 du fichier Armoire.cpp.

Références nbCasiers.

Référencé par Supervision : :creerCasiers().

```
00128 {
00129     return nbCasiers;
00130 }
```

8.1.3.6 getNom()

```
QString Armoire::getNom ( ) const
```

Définition de la méthode getNom.

renvoie le nom de l'armoire

Renvoie

le nom de l'armoire

Définition à la ligne 105 du fichier Armoire.cpp.

Références nom.

```
00106 {
00107     return nom;
00108 }
```

8.1.3.7 informationsArmoire

```
void Armoire::informationsArmoire (
             QStringList  )  [signal]
```

Référencé par getInformations().

8.1.3.8 lireAdresseIP()

```
QString Armoire::lireAdresseIP ( )  [private]
```

Définition de la méthode lireAdresseIP.

Récupère l'adresse IP de la Raspberry Pi

Renvoie

l'adresse IP de la Raspberry Pi

Définition à la ligne 149 du fichier Armoire.cpp.

Référencé par Armoire().

```
00150 {
00151     QStringList adresses;
00152     foreach(QHostAddress adresse, QNetworkInterface::allAddresses())
00153     {
00154         // Filtre les adresses localhost ...
00155         if(adresse != QHostAddress::LocalHostIPv6
00156            && adresse != QHostAddress::LocalHost
00157            // ... APIPA ...
00158            && !adresse.isInSubnet(QHostAddress::parseSubnet("169.254.0.0/16"))
00159            // ... Lien Local IPv6
00160            && !adresse.isInSubnet(QHostAddress::parseSubnet("FE80::/64")))
00161         {
00162             qDebug() << Q_FUNC_INFO << adresse.toString();
00163             adresses << adresse.toString();
00164         }
00165     }
00166
00167     foreach(QString adresse, adresses)
00168     {
00169         #ifdef DEBUG_ARMOIRE
00170             qDebug() << Q_FUNC_INFO << adresse;
00171         #endif
00172         if(adresse.contains("192."))
00173             return adresse;
00174     }
00175
00176     /*if(adresses.count() > 0)
00177     {
00178         return adresses.at(0);
00179     }*/
00180
00181     return QString("");
00182 }
```

8.1.3.9 recupererArmoire()

```
void Armoire::recupererArmoire (
            QString idArmoire = "1" )
```

Définition de la méthode recupererArmoire.

Récupère les données de l'armoire dans la base de données

Paramètres

idArmoire	l'id de l'armoire

Définition à la ligne 49 du fichier Armoire.cpp.

Références bdd, description, nbCasiers, nom, Bdd : :recuperer(), TABLE_ARMOIRE_DESCRIPTION, TABLE_ARMOIRE_ID_ARMOIRE, TABLE_ARMOIRE_NB_CASIERS, et TABLE_ARMOIRE_NOM.

Référencé par Armoire().

```
00050 {
00051     QString requeteBDD;
00052
00053     if(!idArmoire.isEmpty()) // par id
00054     {
00055         requeteBDD = "SELECT idArmoire, Nom, Description, nbCasiers from Armoire where idArmoire = '" +
      idArmoire + "'";
00056         QStringList donnees;
00057         bdd->recuperer(requeteBDD, donnees);
00058
00059         #ifdef DEBUG_ARMOIRE
00060             qDebug() << Q_FUNC_INFO << donnees;
00061         #endif
00062
00063         if(donnees.size() > 0)
00064         {
00065             this->idArmoire = donnees.at(TABLE_ARMOIRE_ID_ARMOIRE);
00066             nom = donnees.at(TABLE_ARMOIRE_NOM);
00067             description = donnees.at(TABLE_ARMOIRE_DESCRIPTION);
00068             nbCasiers = donnees.at(TABLE_ARMOIRE_NB_CASIERS);
00069         }
00070     }
00071 }
```

8.1.4 Documentation des données membres

8.1.4.1 adresseIPArmoire

```
QString Armoire::adresseIPArmoire  [private]
```

l'adresse IP de la Raspberry Pi

Définition à la ligne 70 du fichier Armoire.h.

Référencé par Armoire(), getAdresseIPArmoire(), et getInformations().

8.1.4.2 bdd

```
Bdd* Armoire::bdd  [private]
```

association d'un objet Bdd (accès à la base de données)

Définition à la ligne 65 du fichier Armoire.h.

Référencé par Armoire(), et recupererArmoire().

8.1.4.3 description

```
QString Armoire::description  [private]
```

la description de l'armoire

Définition à la ligne 68 du fichier Armoire.h.

Référencé par getDescripton(), getInformations(), et recupererArmoire().

8.1.4.4 idArmoire

```
QString Armoire::idArmoire  [private]
```

l'id de l'armoire

Définition à la ligne 66 du fichier Armoire.h.

Référencé par getIdArmoire(), et getInformations().

8.1.4.5 nbCasiers

```
QString Armoire::nbCasiers  [private]
```

le nombre de casiers dans l'armoire

Définition à la ligne 69 du fichier Armoire.h.

Référencé par getInformations(), getNbCasiers(), et recupererArmoire().

8.1.4.6 nom

```
QString Armoire::nom  [private]
```

le nom de l'armoire

Définition à la ligne 67 du fichier Armoire.h.

Référencé par getInformations(), getNom(), et recupererArmoire().

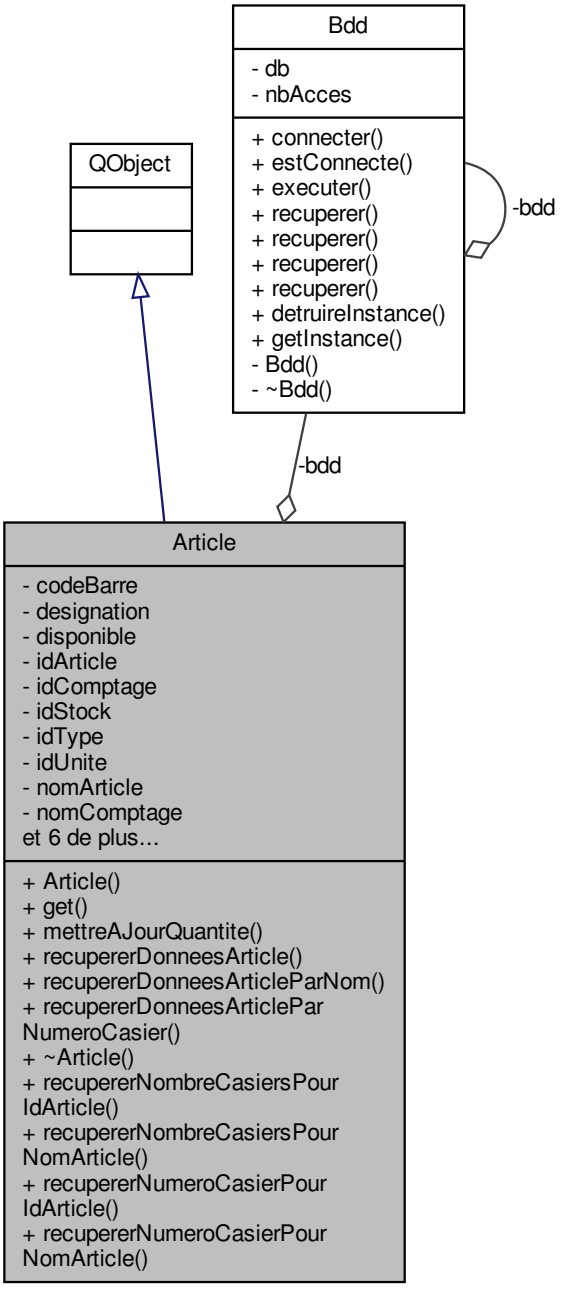
La documentation de cette classe a été générée à partir des fichiers suivants :
- Armoire.h
- Armoire.cpp

8.2 Référence de la classe Article

La classe Article traite les articles.

`#include <Article.h>`

Graphe de collaboration de Article :



Fonctions membres publiques

— Article (QObject *parent=nullptr)

Définition du constructeur de la classe Article.

— QString get (ChampArticle champ)
Définition de la méthode get.
— void mettreAJourQuantite (QString quantite)
Définition de la méthode mettreAJourQuantite.
— bool recupererDonneesArticle (QString idArticle, int numCasier=0)
Récupère les données d'un article de l'armoire dans la base de données par son idArticle.
— bool recupererDonneesArticleParNom (QString nomArticle, int numCasier=0)
Récupère les données d'un article de l'armoire dans la base de données par son nomArticle.
— bool recupererDonneesArticleParNumeroCasier (QString numeroCasier)
Définition de la méthode recupererDonneesArticleParNumeroCasier.
— ∼Article ()
Définition de la méthode ∼Article.

Fonctions membres publiques statiques

— static unsigned int recupererNombreCasiersPourIdArticle (QString idArticle)
Définition de la méthode recupererNombreCasiersPourIdArticle.
— static unsigned int recupererNombreCasiersPourNomArticle (QString nomArticle)
Définition de la méthode recupererNombreCasiersPourNomArticle.
— static QVector< QString > recupererNumeroCasierPourIdArticle (QString idArticle)
Définition de la méthode recupererNumeroCasierPourIdArticle.
— static QVector< QString > recupererNumeroCasierPourNomArticle (QString nomArticle)
Définition de la méthode recupererNumeroCasierPourNomArticle.

Attributs privés

— QString codeBarre
codeBarre de l'article récupéré
— QString designation
designation de l'article récupéré
— QString disponible
disponibilité de l'article récupéré
— QString idArticle
idArticle de l'article récupéré
— QString idComptage
idComptage de l'article récupéré
— QString idStock
idStock de l'article récupéré
— QString idType
idType de l'article récupéré
— QString idUnite
idUnite de l'article récupéré
— QString nomArticle
nomArticle de l'article récupéré
— QString nomComptage
nomComptage de l'article récupéré
— QString nomType
nomType de l'article récupéré
— QString nomUnite
nomUnite de l'article récupéré
— QString numeroCasier
numeroCasier de l'article récupéré
— QString poidsArticle
poidsArticle de l'article récupéré
— QString quantite
quantite de l'article récupéré
— QString tare
tare du numéro de casier de l'article récupéré

Attributs privés statiques

— static Bdd ∗ bdd = Bdd : :getInstance()
association d'un objet Bdd (accès à la base de données)

8.2.1 Description détaillée

La classe Article traite les articles.

Auteur

Tranchat Joffrey

Version

1.0

Date

Mercredi 11 Mars 2020

Définition à la ligne 62 du fichier Article.h.

8.2.2 Documentation des constructeurs et destructeur

8.2.2.1 Article()

```
Article::Article (
            QObject * parent = nullptr )
```

Définition du constructeur de la classe Article.

Initialise un objet Article

Paramètres

parent	l'objet QObject parent

Définition à la ligne 24 du fichier Article.cpp.

```
00024                                : QObject(parent)
00025 {
00026     #ifdef DEBUG_ARTICLE
00027         qDebug() << Q_FUNC_INFO;
00028     #endif
00029     //bdd = Bdd::getInstance();
00030 }
```

8.2.2.2 ~Article()

```
Article::~Article ( )
```

Définition de la méthode ~Article.

Détruit un objet Article

Définition à la ligne 36 du fichier Article.cpp.

```
00037 {
00038     //Bdd::detruireInstance();
00039     #ifdef DEBUG_ARTICLE
00040         qDebug() << Q_FUNC_INFO;
00041     #endif
00042 }
```

8.2.3 Documentation des fonctions membres

8.2.3.1 get()

```
QString Article::get (
            ChampArticle champ )
```

Définition de la méthode get.

Accesseur get pour les différents champs d'un Article

Paramètres

champ	un champ de la table = un attribut

Renvoie

Définition à la ligne 265 du fichier Article.cpp.

Références codeBarre, designation, disponible, idArticle, idComptage, idStock, idType, idUnite, nomArticle, nomComptage, nomType, nomUnite, numeroCasier, poidsArticle, quantite, TABLE_ARTICLE_CODE_BARRE, TABLE_ARTICLE_DESIGNATION, TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_ID_ARTICLE, TABLE_ARTICLE_ID_COMPTAGE, TABLE_ARTICLE_ID_STOCK, TABLE_ARTICLE_ID_TYPE, TABLE_ARTICLE_ID_UNITE, TABLE_ARTICLE_NOM_ARTICLE, TABLE_ARTICLE_NOM_COMPTAGE, TABLE_ARTICLE_NOM_TYPE, TABLE_ARTICLE_NOM_UNITE, TABLE_ARTICLE_NUMERO_CASIER, TABLE_ARTICLE_POIDS, TABLE_ARTICLE_QUANTITE, TABLE_ARTICLE_TARE, et tare.

Référencé par Supervision : :ajouterDonneesArticle(), et Supervision : :traiterTramePoids().

```
00266 {
00267     switch(champ)
00268     {
00269         case TABLE_ARTICLE_ID_STOCK:
00270             return this->idStock;
00271             break;
00272         case TABLE_ARTICLE_ID_ARTICLE:
00273             return this->idArticle;
00274             break;
00275         case TABLE_ARTICLE_NOM_ARTICLE:
00276             return this->nomArticle;
00277             break;
00278         case TABLE_ARTICLE_ID_TYPE:
00279             return this->idType;
00280             break;
00281         case TABLE_ARTICLE_NOM_TYPE:
00282             return this->nomType;
00283             break;
00284         case TABLE_ARTICLE_ID_COMPTAGE:
00285             return this->idComptage;
00286             break;
00287         case TABLE_ARTICLE_NOM_COMPTAGE:
00288             return this->nomComptage;
00289             break;
00290         case TABLE_ARTICLE_CODE_BARRE:
00291             return this->codeBarre;
00292             break;
00293         case TABLE_ARTICLE_DESIGNATION:
00294             return this->designation;
00295             break;
00296         case TABLE_ARTICLE_QUANTITE:
00297             return this->quantite;
00298             break;
00299         case TABLE_ARTICLE_DISPONIBLE:
00300             return this->disponible;
00301             break;
00302         case TABLE_ARTICLE_POIDS:
00303             return this->poidsArticle;
```

```
00304             break;
00305         case TABLE_ARTICLE_TARE:
00306             return this->tare;
00307             break;
00308         case TABLE_ARTICLE_ID_UNITE:
00309             return this->idUnite;
00310             break;
00311         case TABLE_ARTICLE_NOM_UNITE:
00312             return this->nomUnite;
00313             break;
00314         case TABLE_ARTICLE_NUMERO_CASIER:
00315             return this->numeroCasier;
00316             break;
00317         default:
00318             qDebug() << Q_FUNC_INFO << champ << "champ inconnu";
00319     }
00320     return QString("");
00321 }
```

8.2.3.2 mettreAJourQuantite()

```
void Article::mettreAJourQuantite (
             QString quantite )
```

Définition de la méthode mettreAJourQuantite.

permet de mettre à jour la quantite disponible d'un article

Paramètres

quantite	

Définition à la ligne 328 du fichier Article.cpp.

Références bdd, Bdd : :executer(), idArticle, et quantite.

Référencé par Supervision : :traiterTramePoids().

```
00329 {
00330     if(idArticle.isEmpty())
00331         return;
00332     if(this->quantite != quantite)
00333     {
00334         #ifdef DEBUG_ARTICLE
00335             qDebug() << Q_FUNC_INFO << "quantite" << quantite;
00336         #endif
00337         this->quantite = quantite;
00338         QString requete = "UPDATE Stock SET Disponible =" + quantite + " WHERE idArticle =" +
      idArticle + ";";
00339         bdd->executer(requete);
00340     }
00341 }
```

8.2.3.3 recupererDonneesArticle()

```
bool Article::recupererDonneesArticle (
             QString idArticle,
             int numCasier = 0 )
```

Récupère les données d'un article de l'armoire dans la base de données par son idArticle.

Paramètres

idArticle	
numCasier	si égal à 0, l'article est dans un seul casier sinon égal au numéro de casier

Renvoie

true si l'article a été récupéré sinon false

Définition à la ligne 50 du fichier Article.cpp.

Références bdd, codeBarre, designation, disponible, idComptage, idStock, idType, idUnite, nomArticle, nomComptage, nomType, nomUnite, numeroCasier, poidsArticle, quantite, Bdd : :recuperer(), TABLE_ARTICLE_CODE_BARRE, TABLE_ARTICLE_DESIGNATION, TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_ID_ARTICLE, TABLE_ARTICLE_ID_COMPTAGE, TABLE_ARTICLE_ID_STOCK, TABLE_ARTICLE_ID_TYPE, TABLE_ARTICLE_ID_UNITE, TABLE_ARTICLE_NOM_ARTICLE, TABLE_ARTICLE_NOM_COMPTAGE, TABLE_ARTICLE_NOM_TYPE, TABLE_ARTICLE_NOM_UNITE, TABLE_ARTICLE_NUMERO_CASIER, TABLE_ARTICLE_POIDS, TABLE_ARTICLE_QUANTITE, TABLE_ARTICLE_TARE, et tare.

```
00051 {
00052     if(idArticle.isEmpty())
00053         return false;
00054
00055     QString requeteBDD;
00056
00057     if(numCasier == 0)
00058     {
00059         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
      Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
      JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Article.idArticle = '" + idArticle + "'";
00060     }
00061     else
00062     {
00063         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
      Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
      JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Article.idArticle = '" + idArticle + "' AND Stock.numeroCasier = '" + numCasier + "'";
00064     }
00065
00066     QStringList donnees; // un seul casier pour cet article
00067     bdd->recuperer(requeteBDD, donnees);
00068
00069     #ifdef DEBUG_ARTICLE
00070         qDebug() << Q_FUNC_INFO << donnees;
00071     #endif
00072
00073     if(donnees.size() > 0)
00074     {
00075         this->idStock = donnees.at(TABLE_ARTICLE_ID_STOCK);
00076         this->idArticle = donnees.at(TABLE_ARTICLE_ID_ARTICLE);
00077         this->nomArticle = donnees.at(TABLE_ARTICLE_NOM_ARTICLE);
00078         this->idType = donnees.at(TABLE_ARTICLE_ID_TYPE);
00079         this->nomType = donnees.at(TABLE_ARTICLE_NOM_TYPE);
00080         this->idComptage = donnees.at(TABLE_ARTICLE_ID_COMPTAGE);
00081         this->nomComptage = donnees.at(TABLE_ARTICLE_NOM_COMPTAGE);
00082         this->codeBarre = donnees.at(TABLE_ARTICLE_CODE_BARRE);
00083         this->designation = donnees.at(TABLE_ARTICLE_DESIGNATION);
00084         this->quantite = donnees.at(TABLE_ARTICLE_QUANTITE);
00085         this->disponible = donnees.at(TABLE_ARTICLE_DISPONIBLE);
00086         this->poidsArticle = donnees.at(TABLE_ARTICLE_POIDS);
00087         this->tare = donnees.at(TABLE_ARTICLE_TARE);
00088         this->idUnite = donnees.at(TABLE_ARTICLE_ID_UNITE);
00089         this->nomUnite = donnees.at(TABLE_ARTICLE_NOM_UNITE);
00090         this->numeroCasier = donnees.at(TABLE_ARTICLE_NUMERO_CASIER)
    ;
00091         return true;
00092     }
00093     return false;
00094 }
```

8.2.3.4 recupererDonneesArticleParNom()

```
bool Article::recupererDonneesArticleParNom (
            QString nomArticle,
            int numCasier = 0 )
```

Récupère les données d'un article de l'armoire dans la base de données par son nomArticle.

Paramètres

nomArticle	
numCasier	si égal à 0, l'article est dans un seul casier sinon égal au numéro de casier

Renvoie

true si l'article a été récupéré sinon false

Définition à la ligne 102 du fichier Article.cpp.

Références bdd, codeBarre, designation, disponible, idArticle, idComptage, idStock, idType, idUnite, nomComptage, nomType, nomUnite, numeroCasier, poidsArticle, quantite, Bdd : :recuperer(), TABLE_ARTICLE_CODE_BARRE, TABLE_ARTICLE_DESIGNATION, TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_ID_ARTICLE, TABLE_ARTICLE_ID_COMPTAGE, TABLE_ARTICLE_ID_STOCK, TABLE_ARTICLE_ID_TYPE, TABLE_ARTICLE_ID_UNITE, TABLE_ARTICLE_NOM_ARTICLE, TABLE_ARTICLE_NOM_COMPTAGE, TABLE_ARTICLE_NOM_TYPE, TABLE_ARTICLE_NOM_UNITE, TABLE_ARTICLE_NUMERO_CASIER, TABLE_ARTICLE_POIDS, TABLE_ARTICLE_QUANTITE, TABLE_ARTICLE_TARE, et tare.

Référencé par Supervision : :selectionnerArticle().

```
00103 {
00104     if(nomArticle.isEmpty())
00105         return false;
00106
00107     QString requeteBDD;
00108
00109     if(numCasier == 0)
00110     {
00111         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
      Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
      JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Article.Nom = '" + nomArticle + "'";
00112     }
00113     else
00114     {
00115         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
      Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
      JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Article.Nom = '" + nomArticle + "' AND Stock.numeroCasier = '" + numCasier + "'";
00116     }
00117
00118     QStringList donnees; // un seul casier pour cet article
00119     bdd->recuperer(requeteBDD, donnees);
00120
00121     #ifdef DEBUG_ARTICLE
00122         qDebug() << Q_FUNC_INFO << donnees;
00123     #endif
00124
00125     if(donnees.size() > 0)
00126     {
00127         this->idStock = donnees.at(TABLE_ARTICLE_ID_STOCK);
00128         this->idArticle = donnees.at(TABLE_ARTICLE_ID_ARTICLE);
00129         this->nomArticle = donnees.at(TABLE_ARTICLE_NOM_ARTICLE);
00130         this->idType = donnees.at(TABLE_ARTICLE_ID_TYPE);
00131         this->nomType = donnees.at(TABLE_ARTICLE_NOM_TYPE);
00132         this->idComptage = donnees.at(TABLE_ARTICLE_ID_COMPTAGE);
00133         this->nomComptage = donnees.at(TABLE_ARTICLE_NOM_COMPTAGE);
00134         this->codeBarre = donnees.at(TABLE_ARTICLE_CODE_BARRE);
00135         this->designation = donnees.at(TABLE_ARTICLE_DESIGNATION);
00136         this->quantite = donnees.at(TABLE_ARTICLE_QUANTITE);
00137         this->disponible = donnees.at(TABLE_ARTICLE_DISPONIBLE);
```

```
00138         this->poidsArticle = donnees.at(TABLE_ARTICLE_POIDS);
00139         this->tare = donnees.at(TABLE_ARTICLE_TARE);
00140         this->idUnite = donnees.at(TABLE_ARTICLE_ID_UNITE);
00141         this->nomUnite = donnees.at(TABLE_ARTICLE_NOM_UNITE);
00142         this->numeroCasier = donnees.at(TABLE_ARTICLE_NUMERO_CASIER)
    ;
00143         return true;
00144     }
00145
00146     return false;
00147 }
```

8.2.3.5 recupererDonneesArticleParNumeroCasier()

```
bool Article::recupererDonneesArticleParNumeroCasier (
            QString numeroCasier )
```

Définition de la méthode recupererDonneesArticleParNumeroCasier.

permet de recuperer les données d'un article grâçe au numéro du casier

Paramètres

numeroCasier	le numéro du casier

Renvoie

un booleen pour verifier si la mise à jour des données a bien était faite

Définition à la ligne 155 du fichier Article.cpp.

Références bdd, codeBarre, designation, disponible, idArticle, idComptage, idStock, idType, idUnite, nomArticle, nomComptage, nomType, nomUnite, poidsArticle, quantite, Bdd : :recuperer(), TABLE_ARTICLE_CODE_BARRE, TABLE_ARTICLE_DESIGNATION, TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_ID_ARTICLE, TABLE_ARTICLE_ID_COMPTAGE, TABLE_ARTICLE_ID_STOCK, TABLE_ARTICLE_ID_TYPE, TABLE_ARTICLE_ID_UNITE, TABLE_ARTICLE_NOM_ARTICLE, TABLE_ARTICLE_NOM_COMPTAGE, TABLE_ARTICLE_NOM_TYPE, TABLE_ARTICLE_NOM_UNITE, TABLE_ARTICLE_NUMERO_CASIER, TABLE_ARTICLE_POIDS, TABLE_ARTICLE_QUANTITE, TABLE_ARTICLE_TARE, et tare.

Référencé par Supervision : :selectionnerArticle(), et Supervision : :traiterTramePoids().

```
00156 {
00157     if(numeroCasier.isEmpty())
00158         return false;
00159
00160     QString requeteBDD;
00161
00162     requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType, Type.nom AS
      Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER JOIN
      Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Stock.numeroCasier = '" + numeroCasier + "'";
00163
00164     QStringList donnees;
00165     bdd->recuperer(requeteBDD, donnees);
00166
00167     #ifdef DEBUG_ARTICLE
00168         qDebug() << Q_FUNC_INFO << donnees;
00169     #endif
00170
00171     if(donnees.size() > 0)
00172     {
00173         this->idStock = donnees.at(TABLE_ARTICLE_ID_STOCK);
00174         this->idArticle = donnees.at(TABLE_ARTICLE_ID_ARTICLE);
00175         this->nomArticle = donnees.at(TABLE_ARTICLE_NOM_ARTICLE);
00176         this->idType = donnees.at(TABLE_ARTICLE_ID_TYPE);
00177         this->nomType = donnees.at(TABLE_ARTICLE_NOM_TYPE);
```

```
00178         this->idComptage = donnees.at(TABLE_ARTICLE_ID_COMPTAGE);
00179         this->nomComptage = donnees.at(TABLE_ARTICLE_NOM_COMPTAGE);
00180         this->codeBarre = donnees.at(TABLE_ARTICLE_CODE_BARRE);
00181         this->designation = donnees.at(TABLE_ARTICLE_DESIGNATION);
00182         this->quantite = donnees.at(TABLE_ARTICLE_QUANTITE);
00183         this->disponible = donnees.at(TABLE_ARTICLE_DISPONIBLE);
00184         this->poidsArticle = donnees.at(TABLE_ARTICLE_POIDS);
00185         this->tare = donnees.at(TABLE_ARTICLE_TARE);
00186         this->idUnite = donnees.at(TABLE_ARTICLE_ID_UNITE);
00187         this->nomUnite = donnees.at(TABLE_ARTICLE_NOM_UNITE);
00188         this->numeroCasier = donnees.at(TABLE_ARTICLE_NUMERO_CASIER)
    ;
00189         return true;
00190     }
00191
00192     return false;
00193 }
```

8.2.3.6 recupererNombreCasiersPourIdArticle()

```
unsigned int Article::recupererNombreCasiersPourIdArticle (
            QString idArticle )  [static]
```

Définition de la méthode recupererNombreCasiersPourIdArticle.

permet de recuperer le nombre de casiers à partir de l'id d'un article

Paramètres

idArticle	l'id d'un article

Renvoie

un unsigned int qui correspond au nombre de casier

Définition à la ligne 201 du fichier Article.cpp.

Références bdd, et Bdd : :recuperer().

```
00202 {
00203     QString requete = "SELECT COUNT(Stock.idArticle) FROM Stock INNER JOIN Article ON Stock.idArticle =
      Article.idArticle WHERE Article.idArticle = '" + idArticle + "'";
00204
00205     QString donnees;
00206     bdd->recuperer(requete, donnees);
00207
00208     return donnees.toUInt();
00209 }
```

8.2.3.7 recupererNombreCasiersPourNomArticle()

```
unsigned int Article::recupererNombreCasiersPourNomArticle (
            QString nomArticle )  [static]
```

Définition de la méthode recupererNombreCasiersPourNomArticle.

permet de recuperer le nombre de casiers à partir du nom d'un article

Paramètres

nomArticle	le nom d'un article

Renvoie

un unsigned int qui correspond au nombre de casier

Définition à la ligne 217 du fichier Article.cpp.

Références bdd, et Bdd : :recuperer().

Référencé par Supervision : :selectionnerArticle().

```
00218 {
00219     QString requete = "SELECT COUNT(Stock.idArticle) FROM Stock INNER JOIN Article ON Stock.idArticle =
      Article.idArticle WHERE Article.Nom = '" + nomArticle + "'";
00220
00221     QString donnees;
00222     bdd->recuperer(requete, donnees);
00223
00224     return donnees.toUInt();
00225 }
```

8.2.3.8 recupererNumeroCasierPourIdArticle()

```
QVector< QString > Article::recupererNumeroCasierPourIdArticle (
            QString idArticle )  [static]
```

Définition de la méthode recupererNumeroCasierPourIdArticle.

permet de recuperer le numero des casiers à partir de l'id d'un article

Paramètres

idArticle	l'id d'un article

Renvoie

le numero des casiers

Définition à la ligne 233 du fichier Article.cpp.

Références bdd, et Bdd : :recuperer().

```
00234 {
00235     QString requete = "SELECT Stock.numeroCasier FROM Stock INNER JOIN Article ON
      Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN Comptage ON
      Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE Article.idArticle = '" +
     idArticle + "'";
00236
00237     QVector<QString> donnees;
00238     bdd->recuperer(requete, donnees);
00239
00240     return donnees;
00241 }
```

8.2.3.9 recupererNumeroCasierPourNomArticle()

```
QVector< QString > Article::recupererNumeroCasierPourNomArticle (
            QString nomArticle )  [static]
```

Définition de la méthode recupererNumeroCasierPourNomArticle.

permet de recuperer le numero des casiers à partir du nom d'un article

Paramètres

nomArticle	le nom d'un article

Renvoie

le numero des casiers

Définition à la ligne 249 du fichier Article.cpp.

Références bdd, et Bdd : :recuperer().

Référencé par Supervision : :selectionnerArticle().

```
00250 {
00251     QString requete = "SELECT Stock.numeroCasier FROM Stock INNER JOIN Article ON
       Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN Comptage ON
       Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE Article.Nom = '" +
      nomArticle + "'";
00252
00253     QVector<QString> donnees;
00254     bdd->recuperer(requete, donnees);
00255
00256     return donnees;
00257 }
```

8.2.4 Documentation des données membres

8.2.4.1 bdd

```
Bdd * Article::bdd = Bdd::getInstance()  [static], [private]
```

association d'un objet Bdd (accès à la base de données)

Définition à la ligne 81 du fichier Article.h.

Référencé par mettreAJourQuantite(), recupererDonneesArticle(), recupererDonneesArticleParNom(), recupererDonneesArticle↩ ParNumeroCasier(), recupererNombreCasiersPourIdArticle(), recupererNombreCasiersPourNomArticle(), recupererNumeroCasier↩ PourIdArticle(), et recupererNumeroCasierPourNomArticle().

8.2.4.2 codeBarre

```
QString Article::codeBarre  [private]
```

codeBarre de l'article récupéré

Définition à la ligne 89 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.3 designation

```
QString Article::designation  [private]
```

designation de l'article récupéré

Définition à la ligne 90 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.4 disponible

```
QString Article::disponible  [private]
```

disponibilité de l'article récupéré

Définition à la ligne 92 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.5 idArticle

```
QString Article::idArticle  [private]
```

idArticle de l'article récupéré

Définition à la ligne 83 du fichier Article.h.

Référencé par get(), mettreAJourQuantite(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.6 idComptage

```
QString Article::idComptage  [private]
```

idComptage de l'article récupéré

Définition à la ligne 87 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.7 idStock

```
QString Article::idStock  [private]
```

idStock de l'article récupéré

Définition à la ligne 82 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.8 idType

```
QString Article::idType  [private]
```

idType de l'article récupéré

Définition à la ligne 85 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.9 idUnite

```
QString Article::idUnite  [private]
```

idUnite de l'article récupéré

Définition à la ligne 95 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.10 nomArticle

```
QString Article::nomArticle  [private]
```

nomArticle de l'article récupéré

Définition à la ligne 84 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), et recupererDonneesArticleParNumeroCasier().

8.2.4.11 nomComptage

```
QString Article::nomComptage  [private]
```

nomComptage de l'article récupéré

Définition à la ligne 88 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.12 nomType

```
QString Article::nomType  [private]
```

nomType de l'article récupéré

Définition à la ligne 86 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.13 nomUnite

```
QString Article::nomUnite  [private]
```

nomUnite de l'article récupéré

Définition à la ligne 96 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.14 numeroCasier

```
QString Article::numeroCasier  [private]
```

numeroCasier de l'article récupéré

Définition à la ligne 97 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), et recupererDonneesArticleParNom().

8.2.4.15 poidsArticle

```
QString Article::poidsArticle  [private]
```

poidsArticle de l'article récupéré

Définition à la ligne 93 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

8.2.4.16 quantite

```
QString Article::quantite  [private]
```

quantite de l'article récupéré

Définition à la ligne 91 du fichier Article.h.

Référencé par get(), mettreAJourQuantite(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonnees↩ ArticleParNumeroCasier().

8.2.4.17 tare

```
QString Article::tare  [private]
```

tare du numéro de casier de l'article récupéré

Définition à la ligne 94 du fichier Article.h.

Référencé par get(), recupererDonneesArticle(), recupererDonneesArticleParNom(), et recupererDonneesArticleParNumeroCasier().

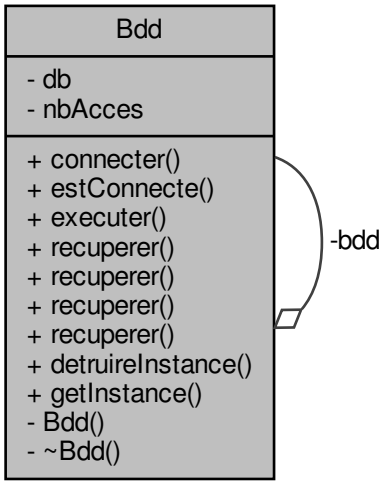
La documentation de cette classe a été générée à partir des fichiers suivants :
- Article.h
- Article.cpp

8.3 Référence de la classe Bdd

Déclaration de la classe utilisant la base de données.

`#include <Bdd.h>`

Graphe de collaboration de Bdd :



Fonctions membres publiques

- bool connecter ()
  Définition méthose connecter()
- bool estConnecte ()
  retourne l'état de connexion à la base de données
- bool executer (QString requete)
  exécute une requête SQL de type UPDATE, INSERT et DELETE
- bool recuperer (QString requete, QString &donnees)
  exécute une requête SQL de type SELECT et récupère un champ d'un seul enregistrement
- bool recuperer (QString requete, QStringList &donnees)
  exécute une requête SQL de type SELECT et récupère plusieurs champs d'un seul enregistrement
- bool recuperer (QString requete, QVector< QString > &donnees)
  exécute une requête SQL de type SELECT et récupère un seul champ de plusieurs enregistrements
- bool recuperer (QString requete, QVector< QStringList > &donnees)
  exécute une requête SQL de type SELECT et récupère plusieurs champs de plusieurs enregistrements

Fonctions membres publiques statiques

- static void detruireInstance ()
  Définition méthode detruireInstance()
- static Bdd * getInstance ()
  Définition méthode getInstance()

Fonctions membres privées

- Bdd ()
  Définition du constructeur de la classe Bdd.
- ~Bdd ()
  Définition du destructeur de la classe Bdd.

Attributs privés

- QSqlDatabase db
  pour la connexion à la base de données MySQL

Attributs privés statiques

- static Bdd * bdd = NULL
  pointeur sur l'instance unique
- static int nbAcces = 0
  compte le nombre d'accès à l'instance unique

8.3.1 Description détaillée

Déclaration de la classe utilisant la base de données.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Vendredi 14 Février

Définition à la ligne 42 du fichier Bdd.h.

8.3.2 Documentation des constructeurs et destructeur

8.3.2.1 Bdd()

```
Bdd::Bdd ( )  [private]
```

Définition du constructeur de la classe Bdd.

initialise le type MySQL pour la connexion à la base de données

Définition à la ligne 27 du fichier Bdd.cpp.

Références db.

Référencé par getInstance().

```
00028 {
00029     #ifdef DEBUG_BDD
00030     qDebug() << Q_FUNC_INFO;
00031     #endif
00032     db = QSqlDatabase::addDatabase("QMYSQL");
00033 }
```

8.3.2.2 ~Bdd()

```
Bdd::~Bdd ( )  [private]
```

Définition du destructeur de la classe Bdd.

destructeur de la classe Bdd

Définition à la ligne 40 du fichier Bdd.cpp.

```
00041 {
00042     #ifdef DEBUG_BDD
00043     qDebug() << Q_FUNC_INFO;
00044     #endif
00045 }
```

8.3.3 Documentation des fonctions membres

8.3.3.1 connecter()

```
bool Bdd::connecter ( )
```

Définition méthose connecter()

permet de se connecter à la base de données

Renvoie

boolean

Définition à la ligne 93 du fichier Bdd.cpp.

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

Référencé par Supervision : :Supervision().

```
00094 {
00095     if(!db.isOpen())
00096     {
00097         db.setHostName(HOSTNAME);
00098         db.setUserName(USERNAME);
00099         db.setPassword(PASSWORD);
00100         db.setDatabaseName(DATABASENAME);
00101 
00102         #ifdef DEBUG_BDD
00103         qDebug() << Q_FUNC_INFO << "HostName" << db.hostName();
00104         qDebug() << Q_FUNC_INFO << "UserName" << db.userName();
00105         qDebug() << Q_FUNC_INFO << "DatabaseName" << db.databaseName();
00106         #endif
00107         if(db.open())
00108         {
00109             #ifdef DEBUG_BDD
00110             qDebug() << Q_FUNC_INFO << QString::fromUtf8("connexion réussie à %1").arg(
      db.hostName());
00111             #endif
00112 
00113             return true;
00114         }
00115         else
00116         {
00117             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur : impossible de se connecter à la base de
       données !");
00118 
00119             QMessageBox::critical(0, QString::fromUtf8("e-stock"), QString::fromUtf8("Impossible de se
      connecter à la base de données !"));
00120             return false;
00121         }
00122 
00123     }
00124     else
00125         return true;
00126 }
```

8.3.3.2 detruireInstance()

```
void Bdd::detruireInstance ( )  [static]
```

Définition méthode detruireInstance()

détruit l'instance de la Bdd si elle existe et si personne l'utilise

Définition à la ligne 73 du fichier Bdd.cpp.

Références bdd, et nbAcces.

Référencé par Armoire : :~Armoire(), et CodeBarre : :~CodeBarre().

```
00074 {
00075     if(bdd != NULL)
00076     {
00077         nbAcces--;
00078         #ifdef DEBUG_BASEDEDONNEES
00079         qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00080         #endif
00081
00082         if(nbAcces == 0)
00083             delete bdd;
00084     }
00085 }
```

8.3.3.3 estConnecte()

```
bool Bdd::estConnecte ( )
```

retourne l'état de connexion à la base de données

retourne l'état de connexion à la base de données

Renvoie

boolean true si connecté à la base de données sinon false

Définition à la ligne 134 du fichier Bdd.cpp.

Références db.

```
00135 {
00136     return db.isOpen();
00137 }
```

8.3.3.4 executer()

```
bool Bdd::executer (
            QString requete )
```

exécute une requête SQL de type UPDATE, INSERT et DELETE

exécute une requête SQL de type UPDATE, INSERT et DELETE

Paramètres

in	*requete*	une requête SQL de type UPDATE, INSERT et DELETE

Renvoie

boolean true si la requête a été exécutée avec succès sinon false

Définition à la ligne 146 du fichier Bdd.cpp.

Références db.

Référencé par Supervision : :ajouterObjetAvecCodeBarre(), Supervision : :mettreAJourMouvement(), Article : :mettreAJourQuantite(), Supervision : :prendreObjetAvecCodeBarre(), et Supervision : :verifierAuthentificationIdentifiant().

```
00147 {
00148     QSqlQuery r;
00149     bool retour;
00150 
00151     if(db.isOpen())
00152     {
00153         if(requete.contains("UPDATE") || requete.contains("INSERT") || requete.contains("DELETE"))
00154         {
00155             retour = r.exec(requete);
00156             #ifdef DEBUG_BASEDEDONNEES
00157             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00158             #endif
00159             if(retour)
00160             {
00161                 return true;
00162             }
00163             else
00164             {
00165                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00166                 return false;
00167             }
00168         }
00169         else
00170         {
00171             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00172             return false;
00173         }
00174     }
00175     else
00176         return false;
00177 
00178 }
```

8.3.3.5 getInstance()

```
Bdd * Bdd::getInstance ( )  [static]
```

Définition méthode getInstance()

permet l'instanciation d'un objet Bdd en vérifiant qu'il n'en existe pas déja un

Renvoie

bdd l'instance unique

Définition à la ligne 53 du fichier Bdd.cpp.

Références Bdd(), bdd, et nbAcces.

Référencé par Armoire : :Armoire(), CodeBarre : :CodeBarre(), et Supervision : :Supervision().

```
00054 {
00055     if(bdd == NULL)
00056         bdd = new Bdd();
00057
00058     nbAcces++;
00059
00060     #ifdef DEBUG_BDD
00061     qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00062     #endif
00063
00064     return bdd;
00065
00066 }
```

8.3.3.6 recuperer() [1/4]

```
bool Bdd::recuperer (
             QString requete,
             QString & donnees )
```

exécute une requête SQL de type SELECT et récupère un champ d'un seul enregistrement

Paramètres

in	*requete*	une requête SQL de type SELECT
out	*donnees*	le champ QString récupéré

Renvoie

boolean true si la requête a été exécutée avec succès sinon false

Définition à la ligne 187 du fichier Bdd.cpp.

Références db.

Référencé par Supervision : :rechercherArticle(), Armoire : :recupererArmoire(), Article : :recupererDonneesArticle(), Article : :recupererDonneesArticleParNom(), Article : :recupererDonneesArticleParNumeroCasier(), Supervision : :recupererDonneesUtilisateur(), CodeBarre : :recupererIdArticleAvecCodeBarres(), Supervision : :recupererIdStockAvecNumeroCasier(), Article : :recupererNombreCasiersPourIdArticle(), Article : :recupererNombreCasiersPourNomArticle(), Article : :recupererNumeroCasierPourIdArticle(), Article : :recupererNumeroCasierPourNomArticle(), CodeBarre : :recupererQuantiteDisponibleParNumeroCasier(), CodeBarre : :recupererQuantiteMaxParNumeroCasier(), et Supervision : :verifierArticlePresentDansCasier().

```
00188 {
00189     QSqlQuery r;
00190     bool retour;
00191
00192     if(db.isOpen())
00193     {
00194         if(requete.contains("SELECT"))
00195         {
00196             retour = r.exec(requete);
```

```
00197             #ifdef DEBUG_BASEDEDONNEES
00198             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00199             #endif
00200             if(retour)
00201             {
00202                 // on se positionne sur l'enregistrement
00203                 r.first();
00204 
00205                 // on vérifie l'état de l'enregistrement retourné
00206                 if(!r.isValid())
00207                 {
00208                     #ifdef DEBUG_BASEDEDONNEES
00209                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00210                     #endif
00211                     return false;
00212                 }
00213 
00214                 // on récupère sous forme de QString la valeur du champ
00215                 if(r.isNull(0))
00216                 {
00217                     #ifdef DEBUG_BASEDEDONNEES
00218                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucun résultat !");
00219                     #endif
00220                     return false;
00221                 }
00222                 donnees = r.value(0).toString();
00223                 #ifdef DEBUG_BASEDEDONNEES
00224                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00225                 #endif
00226                 return true;
00227             }
00228             else
00229             {
00230                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00231                 return false;
00232             }
00233         }
00234         else
00235         {
00236             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00237             return false;
00238         }
00239     }
00240     else
00241         return false;
00242 }
```

8.3.3.7 recuperer() [2/4]

```
bool Bdd::recuperer (
            QString requete,
            QStringList & donnees )
```

exécute une requête SQL de type SELECT et récupère plusieurs champs d'un seul enregistrement

Paramètres

in	*requete*	une requête SQL de type SELECT
out	*donnees*	plusieurs champs d'un seul enregistrement dans un QStringList

Renvoie

boolean true si la requête a été exécutée avec succès sinon false

Définition à la ligne 251 du fichier Bdd.cpp.

Références db.

```
00252 {
00253     QSqlQuery r;
00254     bool retour;
00255
00256     if(db.isOpen())
00257     {
00258         if(requete.contains("SELECT"))
00259         {
00260             retour = r.exec(requete);
00261             #ifdef DEBUG_BASEDEDONNEES
00262             qDebug() << QString::fromUtf8("Retour %1 pour la requete : %2").arg(QString::number(retour)).
      arg(requete);
00263             #endif
00264             if(retour)
00265             {
00266                 // on se positionne sur l'enregistrement
00267                 r.first();
00268
00269                 // on vérifie l'état de l'enregistrement retourné
00270                 if(!r.isValid())
00271                 {
00272                     #ifdef DEBUG_BASEDEDONNEES
00273                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00274                     #endif
00275                     return false;
00276                 }
00277
00278                 // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00279                 // et on les stocke dans une liste de QString
00280                 for(int i=0;i<r.record().count();i++)
00281                     if(!r.isNull(i))
00282                         donnees << r.value(i).toString();
00283                 #ifdef DEBUG_BASEDEDONNEES
00284                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00285                 #endif
00286                 return true;
00287             }
00288             else
00289             {
00290                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00291                 return false;
00292             }
00293         }
00294         else
00295         {
00296             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00297             return false;
00298         }
00299     }
00300     else
00301         return false;
00302 }
```

8.3.3.8 recuperer() [3/4]

```
bool Bdd::recuperer (
            QString requete,
            QVector< QString > & donnees )
```

exécute une requête SQL de type SELECT et récupère un seul champ de plusieurs enregistrements

Paramètres

in	*requete*	une requête SQL de type SELECT
out	*donnees*	un seul champ de plusieurs enregistrements dans un QVector de QString

Renvoie

boolean true si la requête a été exécutée avec succès sinon false

Définition à la ligne 311 du fichier Bdd.cpp.

Références db.

```
00312 {
00313     QSqlQuery r;
00314     bool retour;
00315     QString data;
00316 
00317     if(db.isOpen())
00318     {
00319         if(requete.contains("SELECT"))
00320         {
00321             retour = r.exec(requete);
00322             #ifdef DEBUG_BASEDEDONNEES
00323             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00324             #endif
00325             if(retour)
00326             {
00327                 // pour chaque enregistrement
00328                 while ( r.next() )
00329                 {
00330                     // on récupère sous forme de QString la valeur du champs sélectionné
00331                     data = r.value(0).toString();
00332 
00333                     #ifdef DEBUG_BASEDEDONNEES
00334                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00335                     #endif
00336 
00337                     // on stocke l'enregistrement dans le QVector
00338                     donnees.push_back(data);
00339                 }
00340                 #ifdef DEBUG_BASEDEDONNEES
00341                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00342                 #endif
00343                 return true;
00344             }
00345             else
00346             {
00347                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00348                 return false;
00349             }
00350         }
00351         else
00352         {
00353             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00354             return false;
00355         }
00356     }
00357     else
00358         return false;
00359 }
```

8.3.3.9 recuperer() [4/4]

```
bool Bdd::recuperer (
            QString requete,
            QVector< QStringList > & donnees )
```

exécute une requête SQL de type SELECT et récupère plusieurs champs de plusieurs enregistrements

Paramètres

in	*requete*	une requête SQL de type SELECT
out	*donnees*	: plusieurs champs de plusieurs enregistrements dans un QVector de QStringList

Renvoie

boolean true si la requête a été exécutée avec succès sinon false

Définition à la ligne 367 du fichier Bdd.cpp.

Références db.

```
00368 {
00369     QSqlQuery r;
00370     bool retour;
00371     QStringList data;
00372 
00373     if(db.isOpen())
00374     {
00375         if(requete.contains("SELECT"))
00376         {
00377             retour = r.exec(requete);
00378             #ifdef DEBUG_BASEDEDONNEES
00379             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00380             #endif
00381             if(retour)
00382             {
00383                 // pour chaque enregistrement
00384                 while ( r.next() )
00385                 {
00386                     // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00387                     // et on les stocke dans une liste de QString
00388                     for(int i=0;i<r.record().count();i++)
00389                         data << r.value(i).toString();
00390 
00391                     #ifdef DEBUG_BASEDEDONNEES
00392                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00393                     /*for(int i=0;i<r.record().count();i++)
00394                         qDebug() << r.value(i).toString();*/
00395                     #endif
00396 
00397                     // on stocke l'enregistrement dans le QVector
00398                     donnees.push_back(data);
00399 
00400                     // on efface la liste de QString pour le prochain enregistrement
00401                     data.clear();
00402                 }
00403                 #ifdef DEBUG_BASEDEDONNEES
00404                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00405                 #endif
00406                 return true;
00407             }
00408             else
00409             {
00410                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00411                 return false;
00412             }
00413         }
00414         else
00415         {
00416             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00417             return false;
00418         }
00419     }
00420     else
00421         return false;
00422 }
```

8.3.4 Documentation des données membres

8.3.4.1 bdd

`Bdd * Bdd::bdd = NULL  [static], [private]`

pointeur sur l'instance unique

Définition à la ligne 64 du fichier Bdd.h.

Référencé par detruireInstance(), et getInstance().

8.3.4.2 db

```
QSqlDatabase Bdd::db  [private]
```

pour la connexion à la base de données MySQL

Définition à la ligne 63 du fichier Bdd.h.

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

8.3.4.3 nbAcces

```
int Bdd::nbAcces = 0  [static], [private]
```

compte le nombre d'accès à l'instance unique

Définition à la ligne 65 du fichier Bdd.h.

Référencé par detruireInstance(), et getInstance().

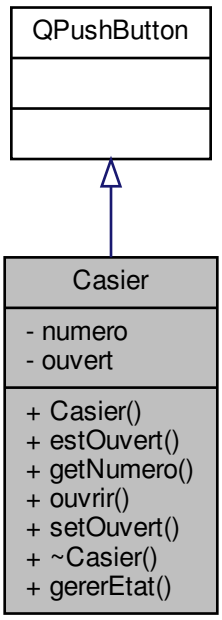
La documentation de cette classe a été générée à partir des fichiers suivants :
- Bdd.h
- Bdd.cpp

8.4 Référence de la classe Casier

La classe Casier gère le casier contenant des articles.

```
#include <Casier.h>
```

Graphe de collaboration de Casier :



Connecteurs publics

— void gererEtat ()
Définition de la méthode gererEtat.

Signaux

— void estOuvert (int numero, bool etat)

Fonctions membres publiques

— Casier (int numero, QWidget *parent=0)
Définition de la méthode Casier.
— bool estOuvert () const
Définition de la méthode estOuvert.
— int getNumero () const
Définition de la méthode getNumero.
— void ouvrir ()
Définition de la méthode ouvrir.
— void setOuvert (bool ouvert)
Définition de la méthode setOuvert.
— ~Casier ()
Définition du destructeur de la classe Casier.

Attributs privés

— int numero
numero du casier
— bool ouvert
état ouvert ou fermé du casier.

8.4.1 Description détaillée

La classe Casier gère le casier contenant des articles.

Auteur

Tranchat Joffrey
Legger Pierre-Antoine

Version

1.0

Date

samedi 28 Mars 2020

Définition à la ligne 34 du fichier Casier.h.

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 Casier()

```
Casier::Casier (
            int numero,
            QWidget * parent = 0 )
```

Définition de la méthode Casier.

Paramètres

numero	
parent	

initialise un objet Casier

A faire Définir une constante pour une taille minimum du casier dans l'IHM

A faire Gérer les différentes couleurs de fond par rapport à l'état (ouvert/fermé, vide, ...)

A faire Connecter signal/slot si nécessaire

Définition à la ligne 23 du fichier Casier.cpp.

Références gererEtat().

```
00023                                       : QPushButton(parent),
      numero(numero), ouvert(false)
00024 {
00025     qDebug() << Q_FUNC_INFO << numero << this;
00026     setText("Casier " + QString::number(numero));
00027
00031     setMaximumHeight(100);
00032     setContentsMargins(10, 0, 10, 0); // Marges : Gauche Haut Droite Bas
00036     //setStyleSheet("background-color: rgb(85, 85, 85);font-size: 18px;"); // inconnu
00037     setStyleSheet("background-color: rgb(239, 41, 41);font-size: 18px;"); // fermé
00038
00042     connect(this, SIGNAL(clicked(bool)), this, SLOT(gererEtat()));
00043 }
```

8.4.2.2 ~Casier()

```
Casier::~Casier ( )
```

Définition du destructeur de la classe Casier.

Détruit un objet Casier

Définition à la ligne 49 du fichier Casier.cpp.

Références numero.

```
00050 {
00051     qDebug() << Q_FUNC_INFO << numero << this;
00052 }
```

8.4.3 Documentation des fonctions membres

8.4.3.1 estOuvert() [1/2]

```
bool Casier::estOuvert ( ) const
```

Définition de la méthode estOuvert.

renvoie l'état ouvert/fermer du casier

Renvoie

état du casier

Définition à la ligne 69 du fichier Casier.cpp.

Références ouvert.

Référencé par setOuvert().

```
00070 {
00071     return ouvert;
00072 }
```

8.4.3.2 estOuvert [2/2]

```
void Casier::estOuvert (
            int numero,
            bool etat )  [signal]
```

8.4.3.3 gererEtat

```
void Casier::gererEtat ( )  [slot]
```

Définition de la méthode gererEtat.

A faire Créer les méthode pour gérer un casier, notamment les signaux et les slots

gère l'état ouvert ou fermer du casier

Définition à la ligne 114 du fichier Casier.cpp.

Références numero, ouvert, ouvrir(), et setOuvert().

Référencé par Casier().

```
00115 {
00116     qDebug() << Q_FUNC_INFO << numero << this;
00117     if(!ouvert)
00118     {
00119         ouvrir();
00120     }
00121     #ifdef SIMULATION_CASIER
00122     else
00123     {
00124         // simule le casier fermé
00125         setOuvert(false);
00126     }
00127     #endif
00128 }
```

8.4.3.4 getNumero()

```
int Casier::getNumero ( ) const
```

Définition de la méthode getNumero.

renvoie le numero du caiser

Renvoie

numero du casier

Définition à la ligne 59 du fichier Casier.cpp.

Références numero.

Référencé par Ihm : :placerCasier(), et Ihm : :placerCasiers().

```
00060 {
00061     return numero;
00062 }
```

8.4.3.5 ouvrir()

```
void Casier::ouvrir ( )
```

Définition de la méthode ouvrir.

envoie la trame d'ouverture du casier

A faire Envoyer trame ouverture

Définition à la ligne 96 du fichier Casier.cpp.

Références setOuvert().

Référencé par gererEtat().

```
00097 {
00102     // simule le casier ouvert
00103     setOuvert(true);
00104 }
```

8.4.3.6 setOuvert()

```
void Casier::setOuvert (
            bool ouvert )
```

Définition de la méthode setOuvert.

modifie l'état puvert ou fermé du casier

Paramètres

bool	ouvert

Définition à la ligne 79 du fichier Casier.cpp.

Références estOuvert(), numero, et ouvert.

Référencé par gererEtat(), et ouvrir().

```
00080 {
00081     if(this->ouvert != ouvert)
00082     {
00083         this->ouvert = ouvert;
00084         if(ouvert)
00085             setStyleSheet("background-color: rgb(115, 210, 22);font-size: 18px;"); // ouvert
00086         else
00087             setStyleSheet("background-color: rgb(239, 41, 41);font-size: 18px;"); // fermé
00088         emit estOuvert(numero, ouvert);
00089     }
00090 }
```

8.4.4 Documentation des données membres

8.4.4.1 numero

```
int Casier::numero  [private]
```

numero du casier

Définition à la ligne 47 du fichier Casier.h.

Référencé par gererEtat(), getNumero(), setOuvert(), et ~Casier().

8.4.4.2 ouvert

```
bool Casier::ouvert  [private]
```

état ouvert ou fermé du casier.

Définition à la ligne 48 du fichier Casier.h.

Référencé par estOuvert(), gererEtat(), et setOuvert().

La documentation de cette classe a été générée à partir des fichiers suivants :

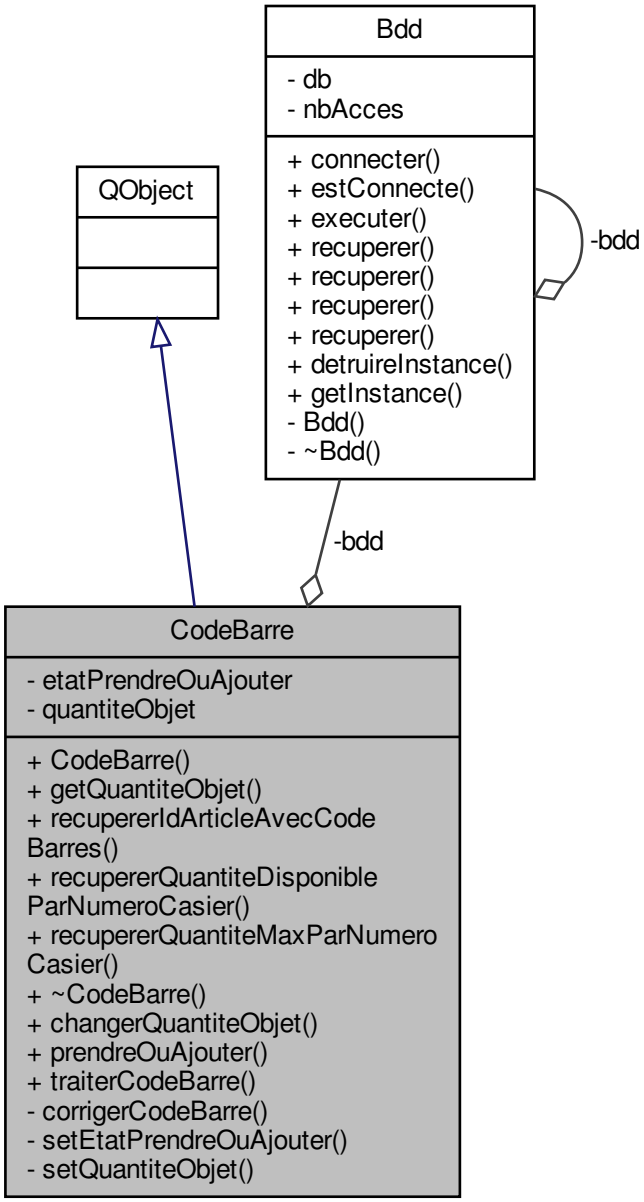
- Casier.h
- Casier.cpp

8.5 Référence de la classe CodeBarre

Déclaration de la classe CodeBarre.

`#include <CodeBarre.h>`

Graphe de collaboration de CodeBarre :



Connecteurs publics

- void changerQuantiteObjet (int quantite)
  Définition de la méthode changerQuantiteObjet.
- void prendreOuAjouter (bool etat)
  Définition de la méthode prendreOuAjouter.
- void traiterCodeBarre (QString codeBarre)
  Définition de la méthode traiterCodeBarre.

Signaux

- void ajouterObjet (QString)
- void prendreObjet (QString)

Fonctions membres publiques

- CodeBarre (QObject *parent=nullptr)
  Définition du constructeur de la classe CodeBare.
- unsigned int getQuantiteObjet ()
  Définition de la méthode getQuantiteObjet.
- unsigned int recupererIdArticleAvecCodeBarres (QString codeBarre)
  Définition de la méthode recupererIdArticleAvecCodeBarres.
- unsigned int recupererQuantiteDisponibleParNumeroCasier (QString numeroCasier)
  Définition de la méthode recupererQuantiteDisponibleParNumeroCasier.
- unsigned int recupererQuantiteMaxParNumeroCasier (QString numeroCasier)
  Définition de la méthode recupererQuantiteMaxParNumeroCasier.
- ~CodeBarre ()
  Définition du destructeur de la classe CodeBare.

Fonctions membres privées

- QString corrigerCodeBarre (QString codeBarre)
  Définition de la méthode corrigerCodeBarre.
- void setEtatPrendreOuAjouter (bool etat)
  Définition de la méthode setEtatPrendreOuAjouter.
- void setQuantiteObjet (unsigned int quantite)
  Définition de la méthode setQuantiteObjet.

Attributs privés

- Bdd * bdd
  association d'un objet Bdd (accès à la base de données)
- bool etatPrendreOuAjouter
  boolean pour savoir si l'on prend ou ajoute un objet(false = prendre, true = ajouter)
- unsigned int quantiteObjet
  quantité d'objet à prendre ou ajouter

8.5.1 Description détaillée

Déclaration de la classe CodeBarre.

Auteur

Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition à la ligne 35 du fichier CodeBarre.h.

8.5.2 Documentation des constructeurs et destructeur

8.5.2.1 CodeBarre()

```
CodeBarre::CodeBarre (
            QObject * parent = nullptr )
```

Définition du constructeur de la classe CodeBare.

Paramètres

parent	

initialise un objet CodeBarre

Définition à la ligne 23 du fichier CodeBarre.cpp.

Références bdd, et Bdd : :getInstance().

```
00023                                     : QObject(parent),
      etatPrendreOuAjouter(false), quantiteObjet(0)
00024 {
00025     #ifdef DEBUG_CODE_BARRE
00026         qDebug() << Q_FUNC_INFO;
00027     #endif
00028     bdd = Bdd::getInstance();
00029 }
```

8.5.2.2 ∼CodeBarre()

```
CodeBarre::∼CodeBarre ( )
```

Définition du destructeur de la classe CodeBare.

Détruit un objet CodeBarre

Définition à la ligne 35 du fichier CodeBarre.cpp.

Références Bdd : :detruireInstance().

```
00036 {
00037     Bdd::detruireInstance();
00038     #ifdef DEBUG_CODE_BARRE
00039         qDebug() << Q_FUNC_INFO;
00040     #endif
00041 }
```

8.5.3 Documentation des fonctions membres

8.5.3.1 ajouterObjet

```
void CodeBarre::ajouterObjet (
             QString  )  [signal]
```

Référencé par traiterCodeBarre().

8.5.3.2 changerQuantiteObjet

```
void CodeBarre::changerQuantiteObjet (
             int quantite )  [slot]
```

Définition de la méthode changerQuantiteObjet.

slot permettant de modifier la valeur de la variable de classe quantiteObjet

Paramètres

quantite	

Définition à la ligne 82 du fichier CodeBarre.cpp.

Références quantiteObjet, et setQuantiteObjet().

```
00083 {
00084     setQuantiteObjet(quantite);
00085
00086     #ifdef DEBUG_CODE_BARRE
00087         qDebug() << Q_FUNC_INFO << "quantiteObjet" << quantiteObjet;
00088     #endif
00089 }
```

8.5.3.3 corrigerCodeBarre()

```
QString CodeBarre::corrigerCodeBarre (
            QString codeBarre )  [private]
```

Définition de la méthode corrigerCodeBarre.

méthode permettant de corriger le code barre en AZERTY au cas où ce dernier aurait était écrit en QWERTY

Paramètres

QString	codeBarre

Définition à la ligne 130 du fichier CodeBarre.cpp.

Référencé par traiterCodeBarre().

```
00131 {
00132     QString codeBarreCorrige = "";
00133
00134     if(!codeBarre.isEmpty())
00135     {
00136         // effectue les remplacements des touches QWERTY en touches AZERTY
00137         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("à"), "0");
00138         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("&"), "1");
00139         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("é"), "2");
00140         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("\""), "3");
00141         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("'"), "4");
00142         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("("), "5");
00143         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("-"), "6");
00144         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("è"), "7");
00145         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("_"), "8");
00146         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("ç"), "9");
00147         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("\n"), "");
00148     }
00149     #ifdef DEBUG_CODE_BARRE
00150         qDebug() << Q_FUNC_INFO << "codeBarreCorrige" << codeBarreCorrige;
00151     #endif
00152     return codeBarreCorrige;
00153 }
```

8.5.3.4 getQuantiteObjet()

```
unsigned int CodeBarre::getQuantiteObjet ( )
```

Définition de la méthode getQuantiteObjet.

accesseur de la variable de classe quantiteObjet

Renvoie

unsigned int quantiteObjet

Définition à la ligne 120 du fichier CodeBarre.cpp.

Références quantiteObjet.

```
00121 {
00122     return quantiteObjet;
00123 }
```

8.5.3.5 prendreObjet

```
void CodeBarre::prendreObjet (
             QString  )  [signal]
```

Référencé par traiterCodeBarre().

8.5.3.6 prendreOuAjouter

```
void CodeBarre::prendreOuAjouter (
             bool etat )  [slot]
```

Définition de la méthode prendreOuAjouter.

slot appeller quand on clique sur le bouton ajouter ou le bouton prendre de la page codeBarre

Paramètres

boolean	etat

Définition à la ligne 72 du fichier CodeBarre.cpp.

Références setEtatPrendreOuAjouter().

```
00073 {
00074     setEtatPrendreOuAjouter(etat);
00075 }
```

8.5.3.7 recupererIdArticleAvecCodeBarres()

```
unsigned int CodeBarre::recupererIdArticleAvecCodeBarres (
            QString codeBarre )
```

Définition de la méthode recupererIdArticleAvecCodeBarres.

méthode permettant de récupérer la quantité disponible d'un objet dans un casier avec son numéro

Paramètres

QString	codeBarre

Renvoie

un unsigned int qui correspond à l'idArticle

Définition à la ligne 193 du fichier CodeBarre.cpp.

Références bdd, et Bdd : :recuperer().

```
00194 {
00195     QString requete = "SELECT idArticle FROM Article WHERE Code = '" + codeBarre + "'";
00196
00197     QString donnees;
00198     bdd->recuperer(requete, donnees);
00199
00200     return donnees.toUInt();
00201 }
```

8.5.3.8 recupererQuantiteDisponibleParNumeroCasier()

```
unsigned int CodeBarre::recupererQuantiteDisponibleParNumeroCasier (
            QString numeroCasier )
```

Définition de la méthode recupererQuantiteDisponibleParNumeroCasier.

méthode permettant de récupérer la quantité disponible d'un objet dans un casier avec son numéro

Paramètres

QString	numeroCasier

Renvoie

un unsigned int qui correspond à la quantité disponible possible dans un casier

Définition à la ligne 177 du fichier CodeBarre.cpp.

Références bdd, et Bdd : :recuperer().

```
00178 {
00179     QString requete = "SELECT Stock.Disponible FROM Stock WHERE Stock.numeroCasier = '" + numeroCasier + "'
    ";
00180
00181     QString donnees;
00182     bdd->recuperer(requete, donnees);
00183
00184     return donnees.toUInt();
00185 }
```

8.5.3.9 recupererQuantiteMaxParNumeroCasier()

```
unsigned int CodeBarre::recupererQuantiteMaxParNumeroCasier (
            QString numeroCasier )
```

Définition de la méthode recupererQuantiteMaxParNumeroCasier.

méthode permettant de récupérer la quantité maximum d'un objet dans un casier avec son numéro

Paramètres

QString	numeroCasier

Renvoie

un unsigned int qui correspond à la quantité maximum possible dans un casier

Définition à la ligne 161 du fichier CodeBarre.cpp.

Références bdd, et Bdd : :recuperer().

```
00162 {
00163     QString requete = "SELECT Stock.Quantite FROM Stock WHERE Stock.numeroCasier = '" + numeroCasier + "'";
00164
00165     QString donnees;
00166     bdd->recuperer(requete, donnees);
00167
00168     return donnees.toUInt();
00169 }
```

8.5.3.10 setEtatPrendreOuAjouter()

```
void CodeBarre::setEtatPrendreOuAjouter (
            bool etat )  [private]
```

Définition de la méthode setEtatPrendreOuAjouter.

mutateur de la variable de classe EtatPrendreOuAjouter

Paramètres

boolean	etat

Définition à la ligne 48 du fichier CodeBarre.cpp.

Références etatPrendreOuAjouter.

Référencé par prendreOuAjouter().

```
00049 {
00050     this->etatPrendreOuAjouter = etat;
00051
00052     #ifdef DEBUG_CODE_BARRE
00053         qDebug() << Q_FUNC_INFO << "etatPrendreOuAjouter" << this->
      etatPrendreOuAjouter ;
00054     #endif
00055 }
```

8.5.3.11 setQuantiteObjet()

```
void CodeBarre::setQuantiteObjet (
            unsigned int quantite )  [private]
```

Définition de la méthode setQuantiteObjet.

mutateur de la variable de classe quantiteObjet

Paramètres

unsigned	int quantite

Définition à la ligne 62 du fichier CodeBarre.cpp.

Références quantiteObjet.

Référencé par changerQuantiteObjet().

```
00063 {
00064     quantiteObjet = quantite;
00065 }
```

8.5.3.12 traiterCodeBarre

```
void CodeBarre::traiterCodeBarre (
            QString codeBarre )  [slot]
```

Définition de la méthode traiterCodeBarre.

slot appeller quand on a scanné un code barres

Paramètres

QString	codeBarre

Définition à la ligne 96 du fichier CodeBarre.cpp.

Références ajouterObjet(), corrigerCodeBarre(), etatPrendreOuAjouter, et prendreObjet().

```
00097 {
00098     #ifdef DEBUG_CODE_BARRE
00099         qDebug() << Q_FUNC_INFO << "codeBarre" << codeBarre;
00100     #endif
00101     QString codeBarreCorriger = corrigerCodeBarre(codeBarre);
00102
00103     if(!etatPrendreOuAjouter)
00104     {
00105         // prendre objet
00106         emit prendreObjet(codeBarreCorriger);
00107     }
00108     else
00109     {
00110         // ajouter objet
00111         emit ajouterObjet(codeBarreCorriger);
00112     }
00113 }
```

8.5.4 Documentation des données membres

8.5.4.1 bdd

```
Bdd* CodeBarre::bdd  [private]
```

association d'un objet Bdd (accès à la base de données)

Définition à la ligne 49 du fichier CodeBarre.h.

Référencé par CodeBarre(), recupererIdArticleAvecCodeBarres(), recupererQuantiteDisponibleParNumeroCasier(), et recuperer↩QuantiteMaxParNumeroCasier().

8.5.4.2 etatPrendreOuAjouter

```
bool CodeBarre::etatPrendreOuAjouter  [private]
```

boolean pour savoir si l'on prend ou ajoute un objet(false = prendre, true = ajouter)

Définition à la ligne 50 du fichier CodeBarre.h.

Référencé par setEtatPrendreOuAjouter(), et traiterCodeBarre().

8.5.4.3 quantiteObjet

```
unsigned int CodeBarre::quantiteObjet  [private]
```

quantité d'objet à prendre ou ajouter

Définition à la ligne 51 du fichier CodeBarre.h.

Référencé par changerQuantiteObjet(), getQuantiteObjet(), et setQuantiteObjet().

La documentation de cette classe a été générée à partir des fichiers suivants :

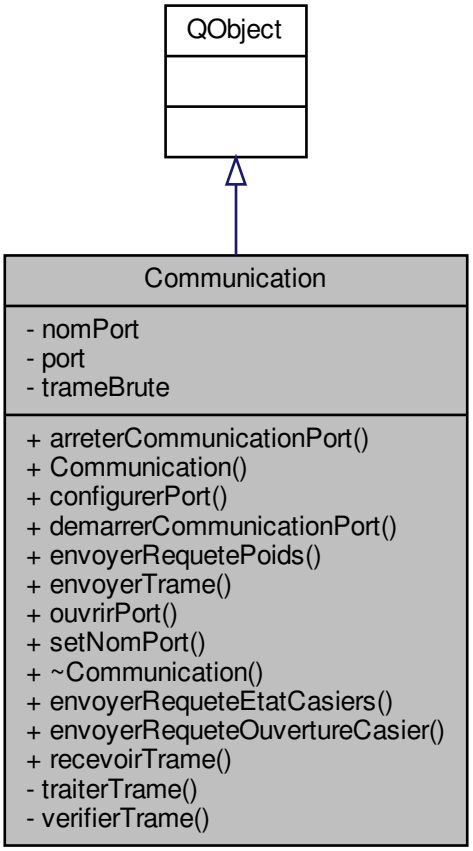
- CodeBarre.h
- CodeBarre.cpp

8.6 Référence de la classe Communication

La classe Communication permet de communiquer avec le port série.

`#include <Communication.h>`

Graphe de collaboration de Communication :



Connecteurs publics

- void envoyerRequeteEtatCasiers (QString numeroCasier)
  Définition de la méthode envoyerRequeteEtatCasiers.
- void envoyerRequeteOuvertureCasier (QString numeroCasier)
  Définition de la méthode envoyerRequeteOuvertureCasier.
- void recevoirTrame ()
  Définition de la méthode recevoirTrame.

Signaux

- void envoieTrameEtat (QString trame)
- void envoieTrameOuverture (QString trame)
- void envoieTramePoids (QString trame)

Fonctions membres publiques

- void arreterCommunicationPort ()
  Définition de la méthode arreterCommunicationPort.
- Communication (QObject *parent=nullptr)
  Constructeur de la classe Communication.
- void configurerPort ()
  Définition de la méthode configurerPort.
- void demarrerCommunicationPort ()
  Définition de la méthode demarrerCommunicationPort.
- void envoyerRequetePoids (QString numeroCasier=0)
  Définition de la méthode envoyerRequetePoids.
- void envoyerTrame (QString trame)
  Définition de la méthode envoyerTrame.
- void ouvrirPort ()
  Définition de la métohde ouvrirPort.
- void setNomPort (QString nouveauPortSerie)
  Définition de la méthode setNomPort.
- ~Communication ()
  Destructeur de la classe Communication.

Fonctions membres privées

- void traiterTrame (QString trame)
  Définition de la méthode TraiterTrame.
- bool verifierTrame (QString trame)
  Définition de la méthode verifierTrame.

Attributs privés

- QString nomPort
  Variable qui contient le nom du port serie.
- QSerialPort * port
  Variable pointeur sur le port.
- QString trameBrute
  Variable qui contient la trame brute.

8.6.1 Description détaillée

La classe Communication permet de communiquer avec le port série.

Auteur

Tranchat Joffrey
Legger Pierre-Antoine

Version

1.0

Date

jeudi 12 Mars 2020

Définition à la ligne 48 du fichier Communication.h.

8.6.2 Documentation des constructeurs et destructeur

8.6.2.1 Communication()

```
Communication::Communication (
            QObject * parent = nullptr )
```

Constructeur de la classe Communication.

initialise un objet Communication

Paramètres

parent	

Définition à la ligne 23 du fichier Communication.cpp.

```
00023                                           : QObject(parent), port(new QSerialPort(this)),
      trameBrute("\0"), nomPort(SERIAL_PORT_NAME)
00024 {
00025     #ifdef DEBUG_COMMUNICATION
00026         qDebug() << Q_FUNC_INFO;
00027     #endif
00028 }
```

8.6.2.2 ~Communication()

```
Communication::~Communication ( )
```

Destructeur de la classe Communication.

Détruit uun objet Communication et ferme le port série

Définition à la ligne 34 du fichier Communication.cpp.

Références port.

```
00035 {
00036     port->close();
00037     #ifdef DEBUG_COMMUNICATION
00038         qDebug() << Q_FUNC_INFO;
00039     #endif
00040 }
```

8.6.3 Documentation des fonctions membres

8.6.3.1 arreterCommunicationPort()

```
void Communication::arreterCommunicationPort ( )
```

Définition de la méthode arreterCommunicationPort.

Méthode qui ferme le port série

Définition à la ligne 59 du fichier Communication.cpp.

Références port.

```
00060 {
00061     #ifdef DEBUG_COMMUNICATION
00062         qDebug() << Q_FUNC_INFO;
00063     #endif
00064     port->close();
00065 }
```

8.6.3.2 configurerPort()

```
void Communication::configurerPort ( )
```

Définition de la méthode configurerPort.

Méthode qui configure le port serie par défaut

Définition à la ligne 71 du fichier Communication.cpp.

Références nomPort, et port.

Référencé par demarrerCommunicationPort().

```
00072 {
00073     #ifdef DEBUG_COMMUNICATION
00074         qDebug() << Q_FUNC_INFO;
00075     #endif
00076     port->setPortName(nomPort);
00077     port->setBaudRate(QSerialPort::Baud9600);
00078     port->setDataBits(QSerialPort::Data8);
00079     port->setParity(QSerialPort::NoParity);
00080     port->setStopBits(QSerialPort::OneStop);
00081     port->setFlowControl(QSerialPort::NoFlowControl);
00082 }
```

8.6.3.3 demarrerCommunicationPort()

```
void Communication::demarrerCommunicationPort ( )
```

Définition de la méthode demarrerCommunicationPort.

Méthode qui démarre la configuration du port serie et ouvre le port serie

Définition à la ligne 46 du fichier Communication.cpp.

Références configurerPort(), et ouvrirPort().

Référencé par Supervision : :Supervision().

```
00047 {
00048     #ifdef DEBUG_COMMUNICATION
00049         qDebug() << Q_FUNC_INFO;
00050     #endif
00051     configurerPort();
00052     ouvrirPort();
00053 }
```

8.6.3.4 envoieTrameEtat

```
void Communication::envoieTrameEtat (
            QString trame )  [signal]
```

Référencé par traiterTrame().

8.6.3.5 envoieTrameOuverture

```
void Communication::envoieTrameOuverture (
            QString trame )  [signal]
```

Référencé par traiterTrame().

8.6.3.6 envoieTramePoids

```
void Communication::envoieTramePoids (
            QString trame )  [signal]
```

Référencé par traiterTrame().

8.6.3.7 envoyerRequeteEtatCasiers

```
void Communication::envoyerRequeteEtatCasiers (
            QString numeroCasier )  [slot]
```

Définition de la méthode envoyerRequeteEtatCasiers.

Méthode qui permet d'envoyer une requête pour demande l'état d'un ou plusieurs casiers

Paramètres

numeroCasier	

Définition à la ligne 219 du fichier Communication.cpp.

Références envoyerTrame().

```
00220 {
00221     QString trame = "CASIERS;2;" + numeroCasier + ";\r\n";
00222     envoyerTrame(trame);
00223 }
```

8.6.3.8 envoyerRequeteOuvertureCasier

```
void Communication::envoyerRequeteOuvertureCasier (
            QString numeroCasier )  [slot]
```

Définition de la méthode envoyerRequeteOuvertureCasier.

Méthode qui permet d'envoyer une requête pour ouvrire un casier

Paramètres

numeroCasier	

Définition à la ligne 208 du fichier Communication.cpp.

Références envoyerTrame().

```
00209 {
00210     QString trame = "CASIERS;1;" + numeroCasier + ";\r\n";
00211     envoyerTrame(trame);
00212 }
```

8.6.3.9 envoyerRequetePoids()

```
void Communication::envoyerRequetePoids (
            QString numeroCasier = 0 )
```

Définition de la méthode envoyerRequetePoids.

Méthode qui permet d'envoyer une requête pour peser un casier

Paramètres

numeroCasier	

Définition à la ligne 197 du fichier Communication.cpp.

Références envoyerTrame().

```
00198 {
00199     QString trame = "CASIERS;3;" + numeroCasier + ";\r\n";
00200     envoyerTrame(trame);
00201 }
```

8.6.3.10 envoyerTrame()

```
void Communication::envoyerTrame (
            QString trame )
```

Définition de la méthode envoyerTrame.

Méthode qui permet d'envoyer une trame via le port série

Paramètres

trame	

Définition à la ligne 123 du fichier Communication.cpp.

Références port.

Référencé par envoyerRequeteEtatCasiers(), envoyerRequeteOuvertureCasier(), et envoyerRequetePoids().

```
00124 {
00125     if (port->isOpen())
00126     {
00127        port->write(trame.toLatin1());
00128     }
00129 }
```

8.6.3.11 ouvrirPort()

```
void Communication::ouvrirPort ( )
```

Définition de la métohde ouvrirPort.

Méthode qui ouvre le port serie en lecture et écriture

Définition à la ligne 88 du fichier Communication.cpp.

Références nomPort, port, et recevoirTrame().

Référencé par demarrerCommunicationPort().

```
00089 {
00090     if (port->open(QIODevice::ReadWrite))
00091     {
00092         #ifdef DEBUG_COMMUNICATION
00093             qDebug() << Q_FUNC_INFO << "connecté au port" << nomPort;
00094         #endif
00095         connect(port, SIGNAL(readyRead()), this, SLOT(recevoirTrame()));
00096     }
00097     else
00098     {
00099         #ifdef DEBUG_COMMUNICATION
00100             qDebug() << Q_FUNC_INFO << "erreur ouverture du port" << port->error();
00101         #endif
00102     }
00103 }
```

8.6.3.12 recevoirTrame

```
void Communication::recevoirTrame ( )  [slot]
```

Définition de la méthode recevoirTrame.

Méthode qui permet de recevoir une trame via le port série

Définition à la ligne 135 du fichier Communication.cpp.

Références port, traiterTrame(), trameBrute, et verifierTrame().

Référencé par ouvrirPort().

```
00136 {
00137     trameBrute = "\0";
00138
00139     while (port->waitForReadyRead(500))
00140     {
00141         trameBrute.append(port->readAll());
00142     }
00143
00144     if(verifierTrame(trameBrute))
00145         traiterTrame(trameBrute);
00146 }
```

8.6.3.13 setNomPort()

```
void Communication::setNomPort (
             QString nouveauPortSerie )
```

Définition de la méthode setNomPort.

Méthode qui permet de définir le nom du port série à utiliser

Paramètres

nouveauPortSerie	

Définition à la ligne 110 du fichier Communication.cpp.

Références nomPort.

```
00111 {
00112     nomPort = nouveauPortSerie;
00113     #ifdef DEBUG_COMMUNICATION
00114         qDebug() << Q_FUNC_INFO << nomPort;
00115     #endif
00116 }
```

8.6.3.14 traiterTrame()

```
void Communication::traiterTrame (
            QString trame )  [private]
```

Définition de la méthode TraiterTrame.

Méthode qui signale le type de trame reçue

Paramètres

trame	

Définition à la ligne 176 du fichier Communication.cpp.

Références DELIMITEUR_CHAMP, EN_TETE, envoieTrameEtat(), envoieTrameOuverture(), envoieTramePoids(), TRAME_ETAT, TRAME_OUVERTURE, et TRAME_POIDS.

Référencé par recevoirTrame().

```
00177 {
00178     if(trame.startsWith(EN_TETE + DELIMITEUR_CHAMP +
      TRAME_OUVERTURE + DELIMITEUR_CHAMP))
00179     {
00180         emit envoieTrameOuverture(trame);
00181     }
00182     else if(trame.startsWith(EN_TETE + DELIMITEUR_CHAMP +
      TRAME_ETAT + DELIMITEUR_CHAMP))
00183     {
00184         emit envoieTrameEtat(trame);
00185     }
00186     else if(trame.startsWith(EN_TETE + DELIMITEUR_CHAMP +
      TRAME_POIDS + DELIMITEUR_CHAMP))
00187     {
00188         emit envoieTramePoids(trame);
00189     }
00190 }
```

8.6.3.15 verifierTrame()

```
bool Communication::verifierTrame (
            QString trame )  [private]
```

Définition de la méthode verifierTrame.

Méthode qui vérifie si la trame respecte le protocole

Paramètres

trame	

Renvoie

un booleen qui indique si la trame est correct ou nom

Définition à la ligne 154 du fichier Communication.cpp.

Références DELIMITEUR_FIN, et EN_TETE.

Référencé par recevoirTrame().

```
00155 {
00156     #ifdef DEBUG_COMMUNICATION
00157         qDebug() << Q_FUNC_INFO << trame;
00158     #endif
00159     if(!trame.startsWith(EN_TETE))
00160     {
00161         return false;
00162     }
00163     if(!trame.endsWith(DELIMITEUR_FIN))
00164     {
00165         return false;
00166     }
00167
00168     return true;
00169 }
```

8.6.4 Documentation des données membres

8.6.4.1 nomPort

```
QString Communication::nomPort  [private]
```

Variable qui contient le nom du port serie.

Définition à la ligne 73 du fichier Communication.h.

Référencé par configurerPort(), ouvrirPort(), et setNomPort().

8.6.4.2 port

```
QSerialPort* Communication::port  [private]
```

Variable pointeur sur le port.

Définition à la ligne 71 du fichier Communication.h.

Référencé par arreterCommunicationPort(), configurerPort(), envoyerTrame(), ouvrirPort(), recevoirTrame(), et ~Communication().

8.6.4.3 trameBrute

```
QString Communication::trameBrute  [private]
```

Variable qui contient la trame brute.

Définition à la ligne 72 du fichier Communication.h.

Référencé par recevoirTrame().

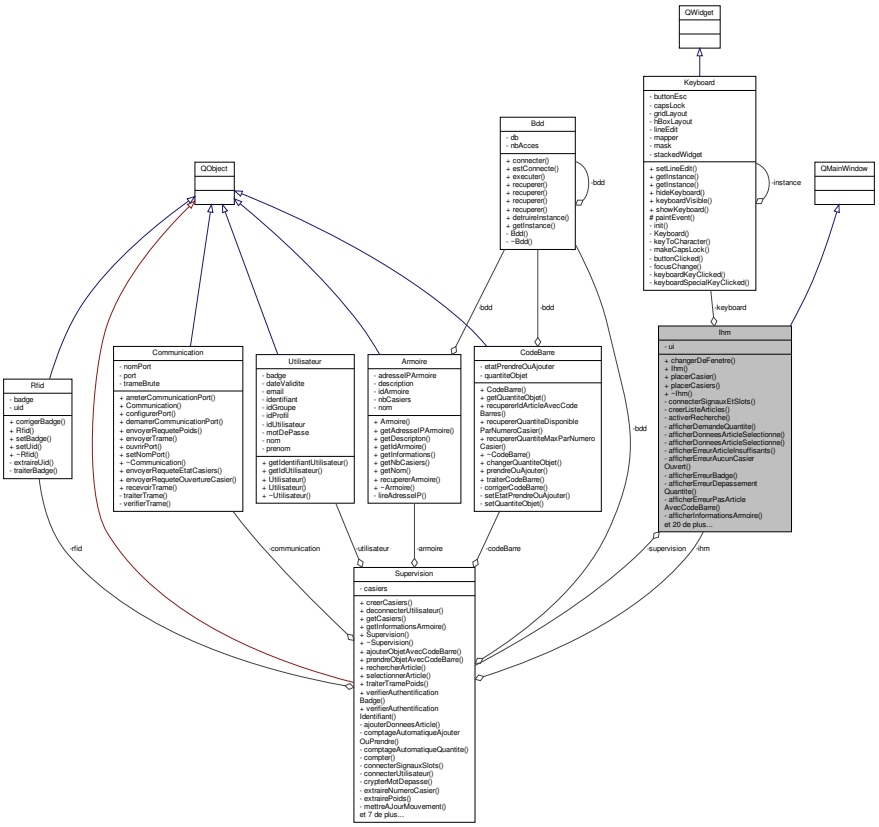
La documentation de cette classe a été générée à partir des fichiers suivants :
- Communication.h
- Communication.cpp

8.7 Référence de la classe Ihm

Déclaration de la classe Ihm.

```
#include <Ihm.h>
```

Graphe de collaboration de Ihm :



Signaux

- void articleSelectionne (QString)
- void badgeDetecte (QString)
- void boutonAjouter (bool)
- void boutonPrendre (bool)
- void codeBarreObjetScanner (QString)
- void envoyerQuantite (int)
- void identifiantDetecte (QString identifiant, QString motDePasse)
- void rechercheArticle (QString)

Fonctions membres publiques

- void changerDeFenetre (int fenetre)
  Définition de la méthode changerDeFenetre.
- Ihm (QWidget *parent=nullptr)
  Constructeur de la classe Ihm.
- void placerCasier (Casier *casier)
  Définition de la méthode placerCasier.
- void placerCasiers (const QVector< Casier *> &casiers, int fenetre)
- ~Ihm ()
  Destructeur de la classe Ihm.

Connecteurs privés

- void activerRecherche ()
  Définition de la méthode traiterDemandeDeConnexion.
- void afficherDemandeQuantite (int)
  définition de la méthode afficherDemandeQuantite
- void afficherDonneesArticleSelectionne (QStringList donneesArticle)
  Définition de la méthode afficherDonneesArticleSelectionne.
- void afficherDonneesArticleSelectionne (QVector< QStringList > donneesArticle)
  Définition de la méthode afficherDonneesArticleSelectionne.
- void afficherErreurArticleInsuffisants ()
  définition de la méthode afficherErreurArticleInsuffisants
- void afficherErreurAucunCasierOuvert ()
  définition de la méthode afficherErreurAucunCasierOuvert
- void afficherErreurBadge (QString message)
  Définition de la méthode afficherErreurBadge.
- void afficherErreurDepassementQuantite ()
  Définition de la méthode afficherErreurDepassementQuantite.
- void afficherErreurPasArticleAvecCodeBarre ()
  définition de la méthode afficherErreurPasArticleAvecCodeBarre
- void afficherInformationsArmoire (QStringList informationsArmoire)
  Définition de la méthode afficherInformationsArmoire.
- void allerFenetreBadge ()
  Définition de la méthode allerFenetreBadge.
- void allerFenetreIdentifiant ()
  Définition de la méthode allerFenetreIdentifiant.
- void allerFenetreMenu ()
  Définition de la méthode allerFenetreMenu.
- void allerFenetreScannerObjet ()
  Définition de la méthode allerFenetreScannerObjet.
- void appuyerBoutonAjouter ()
  Définition de la méthode appuyerBoutonAjouter.
- void appuyerBoutonPrendre ()
  Définition de la méthode appuyerBoutonPrendre.
- void authentifierParBadge ()
  Définition de la méthode authentifierParBadge.
- void authentifierParIdentifiant ()
  Définition de la méthode authentifierParIdentifiant.
- void changerTexteAjouter ()
  définition de la méthode changerTexteAjouter
- void changerTextePrendre ()
  définition de la méthode changerTextePrendre
- void connecterClavier1 ()
  définition de la méthode connecterClavier1
- void connecterClavier2 ()
  définition de la méthode connecterClavier2
- void deconnecterUtilisateur ()
  Définition de la méthode deconnecterUtilisateur.
- void effacerRechercheArticle ()
  Définition de la méthode effacerRechercheArticle.
- void gererPageScanObjet (int numeroCasier, bool etat)
  Définition de la méthode gererPageScanObjet.
- void mettreAJourListeArticles (QVector< QStringList > articlesTrouves)
  Définition de la méthode mettreAJourListeArticles.
- void rechercherArticle ()
  Définition de la méthode rechercherArticle.
- void scannerObjet ()
  Définition de la méthode objetScanner.

— void selectionnerArticle (int index)
Définition de la méthode selectionnerArticle.
— void traiterDemandeDeConnexion (bool reponse, QString message)
Définition de la méthode traiterDemandeDeConnexion.

Fonctions membres privées

— void connecterSignauxEtSlots ()
Définition de la méthode connecterSignauxEtSlots.
— void creerListeArticles (const QVector< QStringList > &articlesTrouves)
Définition de la méthode creerListeArticles.

Attributs privés

— Keyboard * keyboard
association vers le clavier virtuel
— Supervision * supervision
association vers supervision
— Ui : :Ihm * ui
contenu de l'interface utilisateur

8.7.1 Description détaillée

Déclaration de la classe Ihm.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Vendredi 12 Février 2020

Définition à la ligne 68 du fichier Ihm.h.

8.7.2 Documentation des constructeurs et destructeur

8.7.2.1 Ihm()

```
Ihm::Ihm (
            QWidget * parent = nullptr )  [explicit]
```

Constructeur de la classe Ihm.

Initialise un objet Ihm

Paramètres

parent	

Définition à la ligne 31 du fichier Ihm.cpp.

Références allerFenetreBadge(), connecterSignauxEtSlots(), Supervision : :creerCasiers(), Supervision : :getInformationsArmoire(), Keyboard : :getInstance(), keyboard, supervision, et ui.

```
00031                       : QMainWindow(parent), ui(new Ui::Ihm),
      supervision(new Supervision(this))
00032 {
00033     ui->setupUi(this);
00034     // Suppression des parties inutile du QMainWindow
00035     delete ui->menuBar;
00036     delete ui->mainToolBar;
00037     delete ui->statusBar;
00038
00039     // Récupère le clavier virtuelle
00040     keyboard = Keyboard::getInstance(nullptr, this);
00041
00042     // Affiche la fenêtre par défaut en plein écran
00043     allerFenetreBadge();
00044
00045     // Met la fenêtre en plein écran fenêtrer
00046     setWindowFlags(Qt::WindowStaysOnTopHint);
00047     setWindowFlags(Qt::FramelessWindowHint);
00048     // Pour la Raspberry Pi
00049     //showMaximized();
00050
00051     connecterSignauxEtSlots();
00052
00053     supervision->getInformationsArmoire();
00054     supervision->creerCasiers();
00055 }
```

8.7.2.2 ∼Ihm()

```
Ihm::∼Ihm ( )
```

Destructeur de la classe Ihm.

Détruit un objet Ihm

Définition à la ligne 61 du fichier Ihm.cpp.

Références ui.

```
00062 {
00063     delete ui;
00064 }
```

8.7.3 Documentation des fonctions membres

8.7.3.1 activerRecherche

```
void Ihm::activerRecherche ( )  [private], [slot]
```

Définition de la méthode traiterDemandeDeConnexion.

traite la demande de connexion

Définition à la ligne 332 du fichier Ihm.cpp.

Références ui.

Référencé par connecterSignauxEtSlots().

```
00333 {
00334     if(!ui->lineRecherche->text().isEmpty())
00335         ui->pushRecherche->setEnabled(true);
00336     else
00337         ui->pushRecherche->setEnabled(false);
00338 }
```

8.7.3.2 afficherDemandeQuantite

```
void Ihm::afficherDemandeQuantite (
             int quantite )  [private], [slot]
```

définition de la méthode afficherDemandeQuantite

methode permettant d'effectuer la saisie de la quantité

Paramètres

quantite	

Définition à la ligne 533 du fichier Ihm.cpp.

Références envoyerQuantite(), et ui.

Référencé par connecterSignauxEtSlots().

```
00534 {
00535     emit envoyerQuantite(quantite);
00536     ui->lineEditScanner->setFocus();
00537 }
```

8.7.3.3 afficherDonneesArticleSelectionne [1/2]

```
void Ihm::afficherDonneesArticleSelectionne (
             QStringList donneesArticle )  [private], [slot]
```

Définition de la méthode afficherDonneesArticleSelectionne.

Affiche les données de l'article sélectionnée

Paramètres

donneesArticle	

Définition à la ligne 393 du fichier Ihm.cpp.

Références ARTICLE_DISPONIBLE, ARTICLE_QUANTITE, NUMERO_CASIERS, et ui.

```
00394 {
00395     ui->labelCasier->setText("Casier :");
00396     ui->labelQuantiteNombre->setText(donneesArticle.at(ARTICLE_QUANTITE));
00397     ui->labelDisponibleNombre->setText(donneesArticle.at(ARTICLE_DISPONIBLE));
00398     ui->labelCasierNombre->setText(donneesArticle.at(NUMERO_CASIERS));
00399 }
```

8.7.3.4 afficherDonneesArticleSelectionne [2/2]

```
void Ihm::afficherDonneesArticleSelectionne (
            QVector< QStringList > donneesArticle )  [private], [slot]
```

Définition de la méthode afficherDonneesArticleSelectionne.

Affiche les données des articles sélectionnés

Paramètres

donneesArticle	

Définition à la ligne 406 du fichier Ihm.cpp.

Références ARTICLE_DISPONIBLE, ARTICLE_QUANTITE, NUMERO_CASIERS, et ui.

```
00407 {
00408     if(donneesArticle.size() <= 0)
00409         return;
00410     unsigned int articleQuantite = 0;
00411     unsigned int articleDisponible = 0;
00412     QString casiersQuantite;
00413     QString casiersDisponible;
00414     QString casiers;
00415     int nombreCasiers = donneesArticle.size();
00416
00417     for(int i = 0; i < nombreCasiers; i++)
00418     {
00419         #ifdef DEBUG_IHM
00420             qDebug() << Q_FUNC_INFO << "disponible" << (donneesArticle[i].at(
      ARTICLE_DISPONIBLE)).toUInt();
00421             qDebug() << Q_FUNC_INFO << "articleDisponible" << articleDisponible;
00422             qDebug() << Q_FUNC_INFO << "quantite" << (donneesArticle[i].at(
      ARTICLE_QUANTITE)).toUInt();
00423             qDebug() << Q_FUNC_INFO << "articleQuantite" << articleQuantite;
00424         #endif
00425         articleDisponible += (donneesArticle[i].at(ARTICLE_DISPONIBLE)).toUInt();
00426         articleQuantite += (donneesArticle[i].at(ARTICLE_QUANTITE)).toUInt();
00427
00428         if(i == 0)
00429         {
00430             casiers = donneesArticle[i].at(NUMERO_CASIERS);
00431             casiersDisponible = QString(" (") + donneesArticle[i].at(
      ARTICLE_DISPONIBLE);
00432             casiersQuantite = QString(" (") + donneesArticle[i].at(
      ARTICLE_QUANTITE);
00433         }
00434         else
00435         {
00436             casiers += " et " + donneesArticle[i].at(NUMERO_CASIERS);
```

```
00437             casiersDisponible += QString(" et ") + donneesArticle[i].at(
      ARTICLE_DISPONIBLE);
00438             casiersQuantite += QString(" et ") + donneesArticle[i].at(
      ARTICLE_QUANTITE);
00439         }
00440     }
00441     casiersDisponible += QString(")");
00442     casiersQuantite += QString(")");
00443
00444     ui->labelCasier->setText("Casiers :");
00445     ui->labelQuantiteNombre->setText(QString::number(articleQuantite) + casiersQuantite);
00446     ui->labelDisponibleNombre->setText(QString::number(articleDisponible) + casiersDisponible);
00447     ui->labelCasierNombre->setText(casiers);
00448 }
```

8.7.3.5 afficherErreurArticleInsuffisants

```
void Ihm::afficherErreurArticleInsuffisants ( )  [private], [slot]
```

définition de la méthode afficherErreurArticleInsuffisants

methode permettant d'afficher que le nombre d'article est insuffisant

Définition à la ligne 543 du fichier Ihm.cpp.

Références APPLICATION, et MESSAGE_ERREUR_QUANTITE_INSUFFISANTE.

```
00544 {
00545     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_QUANTITE_INSUFFISANTE));
00546 }
```

8.7.3.6 afficherErreurAucunCasierOuvert

```
void Ihm::afficherErreurAucunCasierOuvert ( )  [private], [slot]
```

définition de la méthode afficherErreurAucunCasierOuvert

methode permettant d'afficher que aucun casier n'est ouvert

Définition à la ligne 552 du fichier Ihm.cpp.

Références APPLICATION, et MESSAGE_ERREUR_AUCUN_CASIER_OUVERT.

```
00553 {
00554     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_AUCUN_CASIER_OUVERT));
00555 }
```

8.7.3.7 afficherErreurBadge

```
void Ihm::afficherErreurBadge (
             QString message )  [private], [slot]
```

Définition de la méthode afficherErreurBadge.

Affiche ErreurBadge

Paramètres

message	

Définition à la ligne 296 du fichier Ihm.cpp.

Références ui.

```
00297 {
00298     ui->labelErreurBadge->setText(message);
00299 }
```

8.7.3.8 afficherErreurDepassementQuantite

```
void Ihm::afficherErreurDepassementQuantite ( )  [private], [slot]
```

Définition de la méthode afficherErreurDepassementQuantite.

Affiche que la quantite est dépasser

Définition à la ligne 305 du fichier Ihm.cpp.

Références APPLICATION, et MESSAGE_ERREUR_DEPASSEMENT_QUANTITE.

```
00306 {
00307     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_DEPASSEMENT_QUANTITE));
00308 }
```

8.7.3.9 afficherErreurPasArticleAvecCodeBarre

```
void Ihm::afficherErreurPasArticleAvecCodeBarre ( )  [private], [slot]
```

définition de la méthode afficherErreurPasArticleAvecCodeBarre

methode permettant d'afficher que aucun article de correspond à ce code-barres

Définition à la ligne 561 du fichier Ihm.cpp.

Références APPLICATION, et MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE.

```
00562 {
00563     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE));
00564 }
```

8.7.3.10 afficherInformationsArmoire

```
void Ihm::afficherInformationsArmoire (
            QStringList informationsArmoire )  [private], [slot]
```

Définition de la méthode afficherInformationsArmoire.

Affiche les informations de l'armoire

Paramètres

informationsArmoire	

Définition à la ligne 176 du fichier Ihm.cpp.

Références TABLE_ARMOIRE_DESCRIPTION, TABLE_ARMOIRE_NB_CASIERS, TABLE_ARMOIRE_NOM, et ui.

```
00177 {
00178     #ifdef DEBUG_IHM
00179         qDebug() << Q_FUNC_INFO << "informationsArmoire" << informationsArmoire;
00180     #endif
00181     ui->labelNomArmoire->setText(informationsArmoire.at(TABLE_ARMOIRE_NOM) + " " +
      informationsArmoire.at(TABLE_ARMOIRE_DESCRIPTION) + " (" + informationsArmoire.at(
      TABLE_ARMOIRE_NB_CASIERS+1) + ")");
00182     ui->labelNbCasiers->setText(informationsArmoire.at(
      TABLE_ARMOIRE_NB_CASIERS));
00183 }
```

8.7.3.11 allerFenetreBadge

```
void Ihm::allerFenetreBadge ( )  [private], [slot]
```

Définition de la méthode allerFenetreBadge.

Permet de se rendre à la fenêtre badge

Définition à la ligne 239 du fichier Ihm.cpp.

Références changerDeFenetre(), FENETRE_BADGE, et ui.

Référencé par connecterSignauxEtSlots(), et Ihm().

```
00240 {
00241     changerDeFenetre(FENETRE_BADGE);
00242     ui->lineBadge->setFocus();
00243 }
```

8.7.3.12 allerFenetreIdentifiant

```
void Ihm::allerFenetreIdentifiant ( )  [private], [slot]
```

Définition de la méthode allerFenetreIdentifiant.

Permet de se rendre à la fenêtre identifiant

Définition à la ligne 250 du fichier Ihm.cpp.

Références changerDeFenetre(), FENETRE_IDENTIFIANT, keyboard, Keyboard : :setLineEdit(), et ui.

Référencé par connecterSignauxEtSlots().

```
00251 {
00252     ui->lineMotDePasse->setEchoMode(QLineEdit::Password);
00253     keyboard->setLineEdit(ui->lineIdentifiant);
00254     changerDeFenetre(FENETRE_IDENTIFIANT);
00255     ui->lineIdentifiant->setFocus();
00256 }
```

8.7.3.13 allerFenetreMenu

```
void Ihm::allerFenetreMenu ( )  [private], [slot]
```

Définition de la méthode allerFenetreMenu.

Permet de se rendre à la fenêtre menu

Définition à la ligne 262 du fichier Ihm.cpp.

Références changerDeFenetre(), FENETRE_MENU, Supervision : :getCasiers(), keyboard, placerCasiers(), rechercheArticle(), Keyboard : :setLineEdit(), supervision, et ui.

Référencé par connecterSignauxEtSlots(), et traiterDemandeDeConnexion().

```
00263 {
00264     changerDeFenetre(FENETRE_MENU);
00265     // Initialisation widgets
00266     placerCasiers(supervision->getCasiers(),
      FENETRE_MENU);
00267     keyboard->setLineEdit(ui->lineRecherche);
00268     ui->pushRecherche->setFocus();
00269     ui->comboBoxArticle->clear();
00270     ui->comboBoxArticle->addItem("Sélectionner un article");
00271     ui->pushRecherche->setEnabled(false);
00272     ui->lineRecherche->setFocus();
00273     // Lance une recherche de tous les articles
00274     emit rechercheArticle("");
00275 }
```

8.7.3.14 allerFenetreScannerObjet

```
void Ihm::allerFenetreScannerObjet ( )  [private], [slot]
```

Définition de la méthode allerFenetreScannerObjet.

Permet de se rendre à la fenêtre scannerObjet

Définition à la ligne 281 du fichier Ihm.cpp.

Références changerDeFenetre(), FENETRE_SCANNER_OBJET, Supervision : :getCasiers(), placerCasiers(), supervision, et ui.

Référencé par connecterSignauxEtSlots().

```
00282 {
00283     placerCasiers(supervision->getCasiers(),
      FENETRE_SCANNER_OBJET);
00284     ui->pushScannerPrendre->setEnabled(false);
00285     ui->pushScannerAjouter->setEnabled(false);
00286     ui->spinBoxQuantiteScan->setValue(0);
00287     ui->spinBoxQuantiteScan->setEnabled(false);
00288     changerDeFenetre(FENETRE_SCANNER_OBJET);
00289 }
```

8.7.3.15 appuyerBoutonAjouter

```
void Ihm::appuyerBoutonAjouter ( )  [private], [slot]
```

Définition de la méthode appuyerBoutonAjouter.

slot pour quand l'on appuie sur le bouton ajouter

Définition à la ligne 486 du fichier Ihm.cpp.

Références boutonAjouter(), et ui.

Référencé par connecterSignauxEtSlots().

```
00487 {
00488     ui->labelScannerObjet->setText("Vous pouvez indiquer une quantité");
00489     ui->lineEditScanner->setFocus();
00490     emit boutonAjouter(true);
00491 }
```

8.7.3.16 appuyerBoutonPrendre

```
void Ihm::appuyerBoutonPrendre ( )  [private], [slot]
```

Définition de la méthode appuyerBoutonPrendre.

slot pour quand l'on appuie sur le bouton prendre

Définition à la ligne 475 du fichier Ihm.cpp.

Références boutonPrendre(), et ui.

Référencé par connecterSignauxEtSlots().

```
00476 {
00477     ui->labelScannerObjet->setText("Vous pouvez indiquer une quantité");
00478     ui->lineEditScanner->setFocus();
00479     emit boutonPrendre(false);
00480 }
```

8.7.3.17 articleSelectionne

```
void Ihm::articleSelectionne (
            QString  )  [signal]
```

8.7.3.18 authentifierParBadge

```
void Ihm::authentifierParBadge ( )  [private], [slot]
```

Définition de la méthode authentifierParBadge.

Récupère le badge et l'envoie à la méthode permettant de traiter le badge

Définition à la ligne 189 du fichier Ihm.cpp.

Références badgeDetecte(), et ui.

Référencé par connecterSignauxEtSlots().

```
00190 {
00191     ui->labelErreurBadge->clear();
00192 
00193     if(ui->lineBadge->text() != "")
00194     {
00195         #ifdef DEBUG_IHM
00196             qDebug() << Q_FUNC_INFO << "Contenu brut badge" << ui->lineBadge->text();
00197         #endif
00198 
00199         QString trameBadge = ui->lineBadge->text();
00200         ui->lineBadge->clear();
00201         emit badgeDetecte(trameBadge);
00202     }
00203 }
```

8.7.3.19 authentifierParIdentifiant

```
void Ihm::authentifierParIdentifiant ( )  [private], [slot]
```

Définition de la méthode authentifierParIdentifiant.

Récupère les identifiants et l'envoie à la méthode permettant de s'authentifier par identifiant

Définition à la ligne 209 du fichier Ihm.cpp.

Références identifiantDetecte(), et ui.

Référencé par connecterSignauxEtSlots().

```
00210 {
00211     if(ui->lineIdentifiant->text() != "")
00212     {
00213         #ifdef DEBUG_IHM
00214             qDebug() << Q_FUNC_INFO << "Identifiant" << ui->lineIdentifiant->text() << "MotDePasse" <<
      ui->lineMotDePasse->text();
00215         #endif
00216 
00217         QString identifiant = ui->lineIdentifiant->text();
00218         QString motDePasse = ui->lineMotDePasse->text();
00219         ui->lineIdentifiant->clear();
00220         ui->lineMotDePasse->clear();
00221         emit identifiantDetecte(identifiant, motDePasse);
00222     }
00223 }
```

8.7.3.20 badgeDetecte

```
void Ihm::badgeDetecte (
            QString  )  [signal]
```

Référencé par authentifierParBadge().

8.7.3.21 boutonAjouter

```
void Ihm::boutonAjouter (
            bool  )  [signal]
```

Référencé par appuyerBoutonAjouter().

8.7.3.22 boutonPrendre

```
void Ihm::boutonPrendre (
            bool  )  [signal]
```

Référencé par appuyerBoutonPrendre().

8.7.3.23 changerDeFenetre()

```
void Ihm::changerDeFenetre (
            int fenetre )
```

Définition de la méthode changerDeFenetre.

Permet de changer de fenêtre sur l'ihm

Paramètres

fenetre	

Définition à la ligne 106 du fichier Ihm.cpp.

Références ui.

Référencé par allerFenetreBadge(), allerFenetreIdentifiant(), allerFenetreMenu(), allerFenetreScannerObjet(), et deconnecter↩Utilisateur().

```
00107 {
00108     ui->stackedWidget->setCurrentIndex(fenetre);
00109 }
```

8.7.3.24 changerTexteAjouter

```
void Ihm::changerTexteAjouter ( )  [private], [slot]
```

définition de la méthode changerTexteAjouter

slot pour changer le texte afficher quand on appuie sur le bouton ajouter

Définition à la ligne 506 du fichier Ihm.cpp.

Références ui.

Référencé par connecterSignauxEtSlots().

```
00507 {
00508     ui->labelScannerObjet->setText("Veuillez scanner l'objet à ajouter");
00509 }
```

8.7.3.25 changerTextePrendre

```
void Ihm::changerTextePrendre ( )  [private], [slot]
```

définition de la méthode changerTextePrendre

slot pour changer le texte afficher quand on appuie sur le bouton prendre

Définition à la ligne 497 du fichier Ihm.cpp.

Références ui.

Référencé par connecterSignauxEtSlots().

```
00498 {
00499     ui->labelScannerObjet->setText("Veuillez scanner l'objet à prendre");
00500 }
```

8.7.3.26 codeBarreObjetScanner

```
void Ihm::codeBarreObjetScanner (
             QString  )  [signal]
```

Référencé par scannerObjet().

8.7.3.27 connecterClavier1

```
void Ihm::connecterClavier1 ( )  [private], [slot]
```

définition de la méthode connecterClavier1

methode permettant de connecter le clavier

Définition à la ligne 599 du fichier Ihm.cpp.

Références keyboard, Keyboard : :setLineEdit(), et ui.

Référencé par connecterSignauxEtSlots().

```
00600 {
00601     keyboard->setLineEdit(ui->lineMotDePasse);
00602 }
```

8.7.3.28 connecterClavier2

```
void Ihm::connecterClavier2 ( )  [private], [slot]
```

définition de la méthode connecterClavier2

methode permettant de connecter le clavier

Définition à la ligne 608 du fichier Ihm.cpp.

Références keyboard, Keyboard : :setLineEdit(), et ui.

Référencé par connecterSignauxEtSlots().

```
00609 {
00610     keyboard->setLineEdit(ui->lineIdentifiant);
00611 }
```

8.7.3.29 connecterSignauxEtSlots()

```
void Ihm::connecterSignauxEtSlots ( )  [private]
```

Définition de la méthode connecterSignauxEtSlots.

Etablie la connexion entre les diffrents signaux et slots

Définition à la ligne 70 du fichier Ihm.cpp.

Références activerRecherche(), afficherDemandeQuantite(), allerFenetreBadge(), allerFenetreIdentifiant(), allerFenetreMenu(), allerFenetreScannerObjet(), appuyerBoutonAjouter(), appuyerBoutonPrendre(), authentifierParBadge(), authentifierParIdentifiant(), changerTexteAjouter(), changerTextePrendre(), connecterClavier1(), connecterClavier2(), deconnecterUtilisateur(), rechercherArticle(), scannerObjet(), et ui.

Référencé par Ihm().

```
00071 {
00072     // Les deux types d'authentifiaction
00073     connect(ui->lineBadge, SIGNAL(editingFinished()), this, SLOT(
      authentifierParBadge()));
00074     connect(ui->pushSeConnecter, SIGNAL(clicked()), this, SLOT(
      authentifierParIdentifiant()));
00075     connect(ui->pushSeDeconnecter, SIGNAL(clicked()), this, SLOT(
      deconnecterUtilisateur()));
00076     connect(ui->lineIdentifiant, SIGNAL(editingFinished()), this, SLOT(
      connecterClavier1()));
00077     connect(ui->lineMotDePasse, SIGNAL(editingFinished()), this, SLOT(
      connecterClavier2()));
00078
00079     //fenêtre scanner un objet
00080     connect(ui->pushScannerObjet, SIGNAL(clicked()), this, SLOT(
      allerFenetreScannerObjet()));
00081     connect(ui->pushScannerSeDeconnecter, SIGNAL(clicked()), this, SLOT(
      deconnecterUtilisateur()));
00082     connect(ui->pushScannerPageStock, SIGNAL(clicked()), this, SLOT(
      allerFenetreMenu()));
00083
00084     // Les deux fenêtres d'authentifiaction
00085     connect(ui->pushParIdentifiant, SIGNAL(clicked()), this, SLOT(
      allerFenetreIdentifiant()));
00086     connect(ui->pushParBadge, SIGNAL(clicked()), this, SLOT(allerFenetreBadge()));
00087
00088     // Article
00089     connect(ui->lineRecherche, SIGNAL(textChanged(QString)), this, SLOT(
      activerRecherche()));
00090     connect(ui->pushRecherche, SIGNAL(clicked(bool)), this, SLOT(
      rechercherArticle()));
00091
00092     // CodeBarre
00093     connect(ui->pushScannerPrendre, SIGNAL(clicked()), this, SLOT(
      appuyerBoutonPrendre()));
00094     connect(ui->pushScannerPrendre, SIGNAL(clicked()), this, SLOT(
      changerTextePrendre()));
00095     connect(ui->pushScannerAjouter, SIGNAL(clicked()), this, SLOT(
      appuyerBoutonAjouter()));
00096     connect(ui->pushScannerAjouter, SIGNAL(clicked()), this, SLOT(
      changerTexteAjouter()));
00097     connect(ui->spinBoxQuantiteScan, SIGNAL(valueChanged(int)), this, SLOT(
      afficherDemandeQuantite(int)));
00098     connect(ui->lineEditScanner, SIGNAL(editingFinished()), this, SLOT(
      scannerObjet()));
00099 }
```

8.7.3.30 creerListeArticles()

```
void Ihm::creerListeArticles (
            const QVector< QStringList > & articlesTrouves )  [private]
```

Définition de la méthode creerListeArticles.

Crée la liste déroulante contenant les articles issus d'une recherche

Paramètres

articlesTrouves	

Définition à la ligne 455 du fichier Ihm.cpp.

Références selectionnerArticle(), et ui.

Référencé par mettreAJourListeArticles().

```
00456 {
00457     disconnect(ui->comboBoxArticle, SIGNAL(currentIndexChanged(int)), this, SLOT(
      selectionnerArticle(int)));
00458     ui->comboBoxArticle->clear();
00459
00460     ui->comboBoxArticle->addItem("Sélectionner un article");
```

```
00461     for(int i = 0 ; i < articlesTrouves.size() ; i++)
00462     {
00463         if(ui->comboBoxArticle->findText(articlesTrouves[i].at(2)) == -1)
00464         {
00465             ui->comboBoxArticle->addItem(articlesTrouves[i].at(2));
00466         }
00467     }
00468     connect(ui->comboBoxArticle, SIGNAL(currentIndexChanged(int)), this, SLOT(
      selectionnerArticle(int)));
00469 }
```

8.7.3.31 deconnecterUtilisateur

```
void Ihm::deconnecterUtilisateur ( )  [private], [slot]
```

Définition de la méthode deconnecterUtilisateur.

Permet de déconnecter l'utilisateur

Définition à la ligne 229 du fichier Ihm.cpp.

Références changerDeFenetre(), Supervision : :deconnecterUtilisateur(), FENETRE_BADGE, et supervision.

Référencé par connecterSignauxEtSlots().

```
00230 {
00231     supervision->deconnecterUtilisateur();
00232     changerDeFenetre(FENETRE_BADGE);
00233 }
```

8.7.3.32 effacerRechercheArticle

```
void Ihm::effacerRechercheArticle ( )  [private], [slot]
```

Définition de la méthode effacerRechercheArticle.

efface la recherche de l'article

Définition à la ligne 354 du fichier Ihm.cpp.

Références ui.

Référencé par mettreAJourListeArticles().

```
00355 {
00356     ui->lineRecherche->setText("");
00357 }
```

8.7.3.33 envoyerQuantite

```
void Ihm::envoyerQuantite (
            int  )  [signal]
```

Référencé par afficherDemandeQuantite().

8.7.3.34 gererPageScanObjet

```
void Ihm::gererPageScanObjet (
            int numeroCasier,
            bool etat )  [private], [slot]
```

Définition de la méthode gererPageScanObjet.

méthode qui permet de gérer la possibilité de scanner un objet

Paramètres

numeroCasier	
etat	

Définition à la ligne 572 du fichier Ihm.cpp.

Références ui.

```
00573 {
00574     if(etat)
00575     {
00576         ui->labelScannerObjet->setText("Veuillez prendre ou ajouter un objet");
00577         ui->pushScannerPrendre->setEnabled(true);
00578         ui->pushScannerAjouter->setEnabled(true);
00579         ui->spinBoxQuantiteScan->setEnabled(true);
00580         ui->spinBoxQuantiteScan->setValue(1); // par défaut
00581         ui->lineEditScanner->setEnabled(true);
00582         ui->lineEditScanner->setFocus();
00583     }
00584     else
00585     {
00586         ui->labelScannerObjet->setText("Veuillez ouvrir un casier");
00587         ui->pushScannerPrendre->setEnabled(false);
00588         ui->pushScannerAjouter->setEnabled(false);
00589         ui->spinBoxQuantiteScan->setValue(0);
00590         ui->spinBoxQuantiteScan->setEnabled(false);
00591         ui->lineEditScanner->setEnabled(false);
00592     }
00593 }
```

8.7.3.35 identifiantDetecte

```
void Ihm::identifiantDetecte (
            QString identifiant,
            QString motDePasse )  [signal]
```

Référencé par authentifierParIdentifiant().

8.7.3.36 mettreAJourListeArticles

```
void Ihm::mettreAJourListeArticles (
            QVector< QStringList > articlesTrouves )  [private], [slot]
```

Définition de la méthode mettreAJourListeArticles.

Mets à jour la liste des articles

Paramètres

articlesTrouves	

Définition à la ligne 364 du fichier Ihm.cpp.

Références creerListeArticles(), et effacerRechercheArticle().

```
00365 {
```

```
00366     #ifdef DEBUG_IHM
00367         qDebug() << Q_FUNC_INFO << "articlesTrouves" << articlesTrouves.size() << articlesTrouves;
00368     #endif
00369     creerListeArticles(articlesTrouves);
00370
00371     effacerRechercheArticle();
00372 }
```

8.7.3.37 placerCasier()

```
void Ihm::placerCasier (
            Casier * casier )
```

Définition de la méthode placerCasier.

gère l'affichage des casiers en fonction du nombre de ces derniers

Paramètres

casier	

Définition à la ligne 116 du fichier Ihm.cpp.

Références Casier : :getNumero(), et ui.

```
00117 {
00118     // pair/impair -> droite/gauche ?
00119     int numero = casier->getNumero() - 1;
00120     if((numero+1)%2)
00121     {
00122         ui->gridLayoutCasiers->addWidget(casier, numero/2, 0, 1, 1);
00123         ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 0, 1, 1);
00124     }
00125     else
00126     {
00127         ui->gridLayoutCasiers->addWidget(casier, numero/2, 1, 1, 1);
00128         ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 1, 1, 1);
00129     }
00130 }
```

8.7.3.38 placerCasiers()

```
void Ihm::placerCasiers (
            const QVector< Casier *> & casiers,
            int fenetre )
```

Définition à la ligne 132 du fichier Ihm.cpp.

Références FENETRE_MENU, FENETRE_SCANNER_OBJET, Casier : :getNumero(), et ui.

Référencé par allerFenetreMenu(), allerFenetreScannerObjet(), et Supervision : :creerCasiers().

```
00133 {
00134     for(int i=0; i < casiers.size(); i++)
00135     {
00136         Casier* casier = casiers[i];
00137
00138         // pair/impair -> droite/gauche ?
00139         int numero = casier->getNumero() - 1;
00140         if((numero+1)%2)
```

```
00141         {
00142             switch (fenetre)
00143             {
00144             case FENETRE_MENU:
00145                 ui->gridLayoutCasiers->addWidget(casier, numero/2, 0, 1, 1);
00146                 break;
00147             case FENETRE_SCANNER_OBJET:
00148                 ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 0, 1, 1);
00149                 break;
00150             default:
00151                 break;
00152             }
00153         }
00154         else
00155         {
00156             switch (fenetre)
00157             {
00158             case FENETRE_MENU:
00159                 ui->gridLayoutCasiers->addWidget(casier, numero/2, 1, 1, 1);
00160                 break;
00161             case FENETRE_SCANNER_OBJET:
00162                 ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 1, 1, 1);
00163                 break;
00164             default:
00165                 break;
00166             }
00167         }
00168     }
00169 }
```

8.7.3.39 rechercheArticle

```
void Ihm::rechercheArticle (
             QString  )  [signal]
```

Référencé par allerFenetreMenu().

8.7.3.40 rechercherArticle

```
void Ihm::rechercherArticle ( )  [private], [slot]
```

Définition de la méthode rechercherArticle.

récupère l'article à rechercher et l'envoie à la méthode qui effectue la recherche

Définition à la ligne 344 du fichier Ihm.cpp.

Références Supervision : :rechercherArticle(), supervision, et ui.

Référencé par connecterSignauxEtSlots().

```
00345 {
00346     if(!ui->lineRecherche->text().isEmpty())
00347         supervision->rechercherArticle(ui->lineRecherche->text());
00348 }
```

8.7.3.41 scannerObjet

```
void Ihm::scannerObjet ( )  [private], [slot]
```

Définition de la méthode objetScanner.

slot pour quand un code barre a était scanné

Définition à la ligne 515 du fichier Ihm.cpp.

Références codeBarreObjetScanner(), et ui.

Référencé par connecterSignauxEtSlots().

```
00516 {
00517     QString codeBarreObjet = ui->lineEditScanner->text();
00518     qDebug() << Q_FUNC_INFO << "codeBarreObjet" << codeBarreObjet;
00519
00520     if(!ui->lineEditScanner->text().isEmpty())
00521     {
00522         emit codeBarreObjetScanner(codeBarreObjet);
00523     }
00524
00525     ui->lineEditScanner->setText("");
00526 }
```

8.7.3.42 selectionnerArticle

```
void Ihm::selectionnerArticle (
            int index )  [private], [slot]
```

Définition de la méthode selectionnerArticle.

selectionne un Article

Paramètres

index	

Définition à la ligne 379 du fichier Ihm.cpp.

Références Supervision : :selectionnerArticle(), supervision, et ui.

Référencé par creerListeArticles().

```
00380 {
00381     #ifdef DEBUG_IHM
00382         qDebug() << Q_FUNC_INFO << "index" << index << ui->comboBoxArticle->currentText();
00383     #endif
00384
00385     supervision->selectionnerArticle(ui->comboBoxArticle->currentText());
00386 }
```

8.7.3.43 traiterDemandeDeConnexion

```
void Ihm::traiterDemandeDeConnexion (
            bool reponse,
            QString message )  [private], [slot]
```

Définition de la méthode traiterDemandeDeConnexion.

traite la demande de connexion

Paramètres

reponse	
message	

Définition à la ligne 316 du fichier Ihm.cpp.

Références allerFenetreMenu(), et APPLICATION.

```
00317 {
00318     if(reponse)
00319     {
00320         allerFenetreMenu();
00321     }
00322     else
00323     {
00324         QMessageBox::critical(nullptr, APPLICATION, message);
00325     }
00326 }
```

8.7.4 Documentation des données membres

8.7.4.1 keyboard

```
Keyboard* Ihm::keyboard  [private]
```

association vers le clavier virtuel

Définition à la ligne 125 du fichier Ihm.h.

Référencé par allerFenetreIdentifiant(), allerFenetreMenu(), connecterClavier1(), connecterClavier2(), et Ihm().

8.7.4.2 supervision

```
Supervision* Ihm::supervision  [private]
```

association vers supervision

Définition à la ligne 124 du fichier Ihm.h.

Référencé par allerFenetreMenu(), allerFenetreScannerObjet(), deconnecterUtilisateur(), Ihm(), rechercherArticle(), et selectionner↩Article().

8.7.4.3 ui

```
Ui::Ihm* Ihm::ui  [private]
```

contenu de l'interface utilisateur

Définition à la ligne 123 du fichier Ihm.h.

Référencé par activerRecherche(), afficherDemandeQuantite(), afficherDonneesArticleSelectionne(), afficherErreurBadge(), afficherInformationsArmoire(), allerFenetreBadge(), allerFenetreIdentifiant(), allerFenetreMenu(), allerFenetreScannerObjet(), appuyerBoutonAjouter(), appuyerBoutonPrendre(), authentifierParBadge(), authentifierParIdentifiant(), changerDeFenetre(), changerTexteAjouter(), changerTextePrendre(), connecterClavier1(), connecterClavier2(), connecterSignauxEtSlots(), creerListe↩Articles(), effacerRechercheArticle(), gererPageScanObjet(), Ihm(), placerCasier(), placerCasiers(), rechercherArticle(), scanner↩Objet(), selectionnerArticle(), et ∼Ihm().

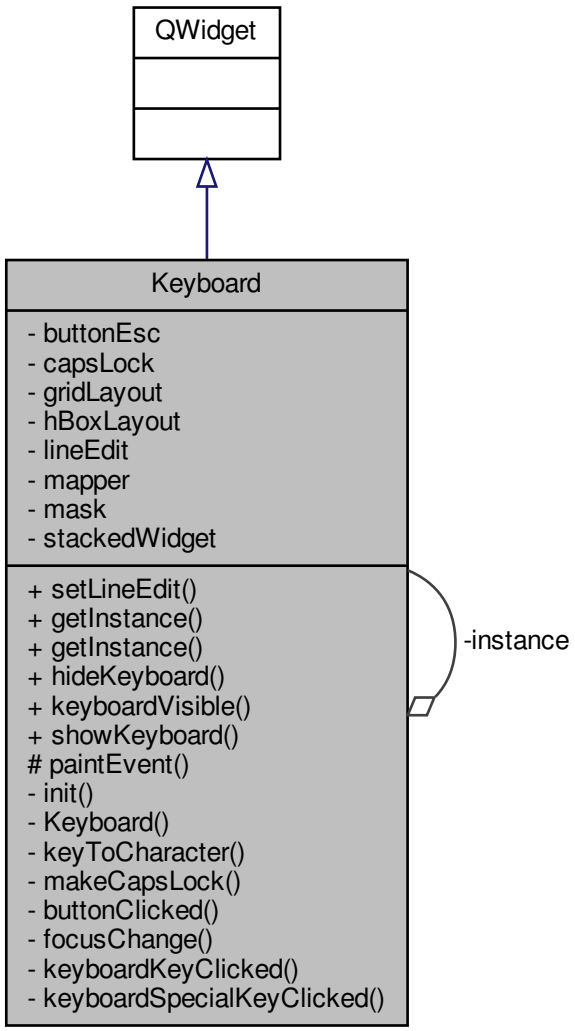
La documentation de cette classe a été générée à partir des fichiers suivants :
- Ihm.h
- Ihm.cpp

8.8 Référence de la classe Keyboard

Déclaration de la classe Keyboard.

```
#include <Keyboard.h>
```

Graphe de collaboration de Keyboard :



Connecteurs publics

- void hideKeyboard ()
- bool keyboardVisible () const
- void showKeyboard (int globalX, int globalY)

Signaux

- void keyClicked (const QString &text)
- void specialKeyClicked (int key)

Fonctions membres publiques

- void setLineEdit (QLineEdit *lineEdit)

Fonctions membres publiques statiques

- static Keyboard * getInstance (QWidget *parent=Q_NULLPTR)
- static Keyboard * getInstance (QLineEdit *lineEdit, QWidget *parent=Q_NULLPTR)

Fonctions membres protégées

- void paintEvent (QPaintEvent *e)

Connecteurs privés

- void buttonClicked (int key)
- void focusChange (QWidget *, QWidget *)
- void keyboardKeyClicked (const QString &characters)
- void keyboardSpecialKeyClicked (int key)

Fonctions membres privées

- void init ()
- Keyboard (QLineEdit *lineEdit=Q_NULLPTR, QWidget *parent=Q_NULLPTR)
- QString keyToCharacter (int key)
- void makeCapsLock ()

Attributs privés

- QPushButton * buttonEsc
- bool capsLock
- QGridLayout * gridLayout
- QHBoxLayout * hBoxLayout
- QLineEdit * lineEdit
- QSignalMapper * mapper
- bool mask
- QStackedWidget * stackedWidget

Attributs privés statiques

- static Keyboard * instance = Q_NULLPTR

8.8.1 Description détaillée

Déclaration de la classe Keyboard.

Cette classe permet de gérer un clavier virtuel

Définition à la ligne 22 du fichier Keyboard.h.

8.8.2 Documentation des constructeurs et destructeur

8.8.2.1 Keyboard()

```
Keyboard::Keyboard (
            QLineEdit * lineEdit = Q_NULLPTR,
            QWidget * parent = Q_NULLPTR )  [explicit], [private]
```

Définition à la ligne 100 du fichier Keyboard.cpp.

Références focusChange(), init(), keyboardKeyClicked(), keyboardSpecialKeyClicked(), keyClicked(), lineEdit, et specialKeyClicked().

```
00100                                                   : QWidget(parent),
      lineEdit(lineEdit), capsLock(false), mask(false)
00101 {
00102 #ifdef DEBUG_KEYBOARD
00103     qDebug() << Q_FUNC_INFO << this << lineEdit;
00104 #endif
00105     init();
00106     connect(qApp, SIGNAL(focusChanged(QWidget*,QWidget*)), this, SLOT(
      focusChange(QWidget*,QWidget*)));
00107     connect(this, SIGNAL(keyClicked(QString)), this, SLOT(
      keyboardKeyClicked(QString)));
00108     connect(this, SIGNAL(specialKeyClicked(int)), this, SLOT(
      keyboardSpecialKeyClicked(int)));
00109     setStyleSheet("/*QWidget{background-color:white;}*/QPushButton{font-family:\"Ubuntu Mono\"
      ;font:bold;fon
      t-size:16px;background-color:palegoldenrod;border-width:1px;border-color:darkkhaki;border-style:solid;border
      -radius:5;padding:1px;}QPushButton:hover{background-color:khaki;}QPushButton:pressed{background-color:#d0d67c;}");
00110     setAttribute(Qt::WA_TranslucentBackground, true);
00111 }
```

8.8.3 Documentation des fonctions membres

8.8.3.1 buttonClicked

```
void Keyboard::buttonClicked (
            int key )  [private], [slot]
```

Définition à la ligne 162 du fichier Keyboard.cpp.

Références capsLock, gridLayout, hBoxLayout, keyClicked(), keyToCharacter(), lineEdit, makeCapsLock(), mask, specialKeyClicked(), et stackedWidget.

Référencé par init().

```
00163 {
00164 #ifdef DEBUG_KEYBOARD
00165     qDebug() << Q_FUNC_INFO << key;
00166 #endif
00167     if ((key == Qt::Key_Escape))
00168     {
00169         mask = !mask;
00170         if(mask)
00171         {
00172             setFixedSize(hBoxLayout->sizeHint());
00173             stackedWidget->setCurrentIndex(1);
00174         }
00175         else
00176         {
00177             setFixedSize(gridLayout->sizeHint());
00178             stackedWidget->setCurrentIndex(0);
00179         }
00180         return;
00181     }
00182     if ((key == Qt::Key_CapsLock))
00183     {
00184         capsLock = !capsLock;
00185         makeCapsLock();
00186         return;
00187     }
00188     if ((key == Qt::Key_Delete))
```

```
00189     {
00190         if(lineEdit)
00191             lineEdit->clear();
00192         return;
00193     }
00194     if ((key == Qt::Key_Enter) || (key == Qt::Key_Backspace))
00195     {
00196         emit specialKeyClicked(key);
00197     }
00198     else
00199     {
00200         emit keyClicked(keyToCharacter(key));
00201     }
00202 }
```

8.8.3.2 focusChange

```
void Keyboard::focusChange (
             QWidget * oldWidget,
             QWidget * newWidget )  [private], [slot]
```

Définition à la ligne 204 du fichier Keyboard.cpp.

Références hideKeyboard(), lineEdit, et showKeyboard().

Référencé par Keyboard().

```
00205 {
00206     if(!lineEdit || !newWidget)
00207         return;
00208 #ifdef DEBUG_KEYBOARD
00209     qDebug() << Q_FUNC_INFO << oldWidget << newWidget;
00210 #endif
00211     if(newWidget == lineEdit)
00212     {
00213         QPoint globalPos(0, 0);
00214         globalPos = newWidget->mapToGlobal(QPoint(0, newWidget->height()));
00215         showKeyboard(globalPos.x(), globalPos.y());
00216     }
00217     else
00218     {
00219         hideKeyboard();
00220     }
00221 }
```

8.8.3.3 getInstance() [1/2]

```
static Keyboard* Keyboard::getInstance (
             QWidget * parent = Q_NULLPTR )  [static]
```

Référencé par Ihm : :Ihm().

8.8.3.4 getInstance() [2/2]

```
Keyboard * Keyboard::getInstance (
             QLineEdit * lineEdit,
             QWidget * parent = Q_NULLPTR )  [static]
```

Définition à la ligne 90 du fichier Keyboard.cpp.

```
00091 {
00092     if(instance == Q_NULLPTR)
00093     {
00094         instance = new Keyboard(lineEdit, parent);
00095     }
00096     return instance;
00097 }
```

8.8.3.5 hideKeyboard

```
void Keyboard::hideKeyboard ( )  [slot]
```

Définition à la ligne 152 du fichier Keyboard.cpp.

Référencé par focusChange().

```
00153 {
00154     QWidget::hide();
00155 }
```

8.8.3.6 init()

```
void Keyboard::init ( )  [private]
```

Définition à la ligne 263 du fichier Keyboard.cpp.

Références buttonClicked(), buttonEsc, gridLayout, hBoxLayout, layoutSize, mapper, NEXT_ROW_MARKER, et stackedWidget.

Référencé par Keyboard().

```
00264 {
00265     setWindowFlags(Qt::WindowDoesNotAcceptFocus | Qt::Tool | Qt::FramelessWindowHint |
      Qt::WindowStaysOnTopHint | Qt::BypassWindowManagerHint);
00266
00267     stackedWidget = new QStackedWidget(this);
00268     QWidget *widgetKeyboard = new QWidget(stackedWidget);
00269     QWidget *widgetMask = new QWidget(stackedWidget);
00270
00271     gridLayout = new QGridLayout(widgetKeyboard);
00272     hBoxLayout = new QHBoxLayout(widgetMask);
00273
00274     mapper = new QSignalMapper(this);
00275     connect(mapper, SIGNAL(mapped(int)), SLOT(buttonClicked(int)));
00276
00277     int row = 0;
00278     int column = 0;
00279
00280     for (int i = 0; i < layoutSize; ++i)
00281     {
00282         if (keyboardLayout[i].key == NEXT_ROW_MARKER)
00283         {
00284             row++;
00285             column = 0;
00286             continue;
00287         }
00288
00289         QPushButton *button = new QPushButton(this);
00290         if (keyboardLayout[i].key == Qt::Key_Enter)
00291             button->setFixedWidth(55);
00292         else
00293             button->setFixedWidth(35);
00294         button->setFixedHeight(25);
00295         button->setText(QString::fromUtf8(keyboardLayout[i].label));
00296
00297         mapper->setMapping(button, keyboardLayout[i].key);
00298         connect(button, SIGNAL(clicked()), mapper, SLOT(map()));
00299
00300         if (keyboardLayout[i].key == Qt::Key_Escape)
00301         {
00302             buttonEsc = new QPushButton(this);
00303             buttonEsc->setFixedWidth(35);
00304             buttonEsc->setText(QString::fromUtf8(keyboardLayout[i].label));
00305             mapper->setMapping(buttonEsc, keyboardLayout[i].key);
00306             connect(buttonEsc, SIGNAL(clicked()), mapper, SLOT(map()));
00307             hBoxLayout->addWidget(buttonEsc);
00308         }
00309         gridLayout->addWidget(button, row, column);
00310         column++;
00311     }
00312     widgetMask->setFixedSize(hBoxLayout->sizeHint());
00313     stackedWidget->addWidget(widgetKeyboard);
00314     stackedWidget->addWidget(widgetMask);
00315 }
```

8.8.3.7 keyboardKeyClicked

```
void Keyboard::keyboardKeyClicked (
            const QString & characters )  [private], [slot]
```

Définition à la ligne 249 du fichier Keyboard.cpp.

Références lineEdit.

Référencé par Keyboard().

```
00250 {
00251 #ifdef DEBUG_KEYBOARD
00252     qDebug() << Q_FUNC_INFO << lineEdit << characters;
00253 #endif
00254     if (!lineEdit)
00255         return;
00256
00257     QInputMethodEvent event;
00258     event.setCommitString(characters);
00259
00260     QGuiApplication::sendEvent(lineEdit, &event);
00261 }
```

8.8.3.8 keyboardSpecialKeyClicked

```
void Keyboard::keyboardSpecialKeyClicked (
            int key )  [private], [slot]
```

Définition à la ligne 223 du fichier Keyboard.cpp.

Références lineEdit.

Référencé par Keyboard().

```
00224 {
00225 #ifdef DEBUG_KEYBOARD
00226     qDebug() << Q_FUNC_INFO << this << key;
00227 #endif
00228     if (!lineEdit)
00229         return;
00230
00231     if (key == Qt::Key_Enter)
00232     {
00233         QKeyEvent *pressEvent = new QKeyEvent(QEvent::KeyPress, Qt::Key_Enter, Qt::NoModifier);
00234         QGuiApplication::postEvent(lineEdit, pressEvent);
00235
00236         QKeyEvent *releaseEvent = new QKeyEvent(QEvent::KeyRelease, Qt::Key_Enter, Qt::NoModifier);
00237         QGuiApplication::postEvent(lineEdit, releaseEvent);
00238     }
00239     else if (key == Qt::Key_Backspace)
00240     {
00241         QKeyEvent *pressEvent = new QKeyEvent(QEvent::KeyPress, Qt::Key_Backspace, Qt::NoModifier);
00242         QGuiApplication::postEvent(lineEdit, pressEvent);
00243
00244         QKeyEvent *releaseEvent = new QKeyEvent(QEvent::KeyRelease, Qt::Key_Backspace, Qt::NoModifier);
00245         QGuiApplication::postEvent(lineEdit, releaseEvent);
00246     }
00247 }
```

8.8.3.9 keyboardVisible

```
bool Keyboard::keyboardVisible ( ) const  [slot]
```

Définition à la ligne 157 du fichier Keyboard.cpp.

```
00158 {
00159     return QWidget::isVisible();
00160 }
```

8.8.3.10 keyClicked

```
void Keyboard::keyClicked (
             const QString & text )  [signal]
```

Référencé par buttonClicked(), et Keyboard().

8.8.3.11 keyToCharacter()

```
QString Keyboard::keyToCharacter (
             int key )  [private]
```

Définition à la ligne 345 du fichier Keyboard.cpp.

Références capsLock, et layoutSize.

Référencé par buttonClicked().

```
00346 {
00347     for (int i = 0; i < layoutSize; ++i)
00348     {
00349         if (keyboardLayout[i].key == key)
00350         {
00351             if(capsLock)
00352                 return QString::fromUtf8(keyboardLayout[i].labelCapsLock);
00353             else
00354                 return QString::fromUtf8(keyboardLayout[i].label);
00355         }
00356         else if (keyboardLayout[i].keyCapsLock == key)
00357             return QString::fromUtf8(keyboardLayout[i].labelCapsLock);
00358     }
00359
00360     return QString();
00361 }
```

8.8.3.12 makeCapsLock()

```
void Keyboard::makeCapsLock ( )  [private]
```

Définition à la ligne 317 du fichier Keyboard.cpp.

Références capsLock, gridLayout, layoutSize, mapper, et NEXT_ROW_MARKER.

Référencé par buttonClicked().

```
00318 {
00319     int row = 0;
00320     int column = 0;
00321     for (int i = 0; i < layoutSize; ++i)
00322     {
00323         if (keyboardLayout[i].key == NEXT_ROW_MARKER)
00324         {
00325             row++;
00326             column = 0;
00327             continue;
00328         }
00329
00330         QPushButton *button = dynamic_cast<QPushButton*>(
      gridLayout->itemAtPosition(row, column)->widget());
00331         if(capsLock)
00332         {
00333             mapper->setMapping(button, keyboardLayout[i].keyCapsLock);
00334             button->setText(QString::fromUtf8(keyboardLayout[i].labelCapsLock));
00335         }
00336         else
00337         {
00338             mapper->setMapping(button, keyboardLayout[i].key);
00339             button->setText(QString::fromUtf8(keyboardLayout[i].label));
00340         }
00341         column++;
00342     }
00343 }
```

8.8.3.13 paintEvent()

```
void Keyboard::paintEvent (
             QPaintEvent * e )  [protected]
```

Définition à la ligne 113 du fichier Keyboard.cpp.

```
00114 {
00115     QPainter painter(this);
00116     QPen pen;
00117     pen.setBrush(QBrush(QColor(128, 128, 128, 64)));
00118     painter.setPen(pen);
00119     painter.drawRoundedRect(0,0,width()-1, height()-1,5,5);
00120     QWidget::paintEvent(e);
00121 }
```

8.8.3.14 setLineEdit()

```
void Keyboard::setLineEdit (
             QLineEdit * lineEdit )
```

Définition à la ligne 123 du fichier Keyboard.cpp.

Références capsLock, lineEdit, et mask.

Référencé par Ihm : :allerFenetreIdentifiant(), Ihm : :allerFenetreMenu(), Ihm : :connecterClavier1(), et Ihm : :connecterClavier2().

```
00124 {
00125     if(lineEdit && this->lineEdit != lineEdit)
00126     {
00127 #ifdef DEBUG_KEYBOARD
00128     qDebug() << Q_FUNC_INFO << this << lineEdit;
00129 #endif
00130         this->lineEdit = lineEdit;
00131         capsLock = false;
00132         mask = false;
00133     }
00134 }
```

8.8.3.15 showKeyboard

```
void Keyboard::showKeyboard (
             int globalX,
             int globalY )  [slot]
```

Définition à la ligne 136 du fichier Keyboard.cpp.

Références gridLayout, hBoxLayout, mask, et stackedWidget.

Référencé par focusChange().

```
00137 {
00138     QWidget::move(globalX, globalY);
00139     if(mask)
00140     {
00141         setFixedSize(hBoxLayout->sizeHint());
00142         stackedWidget->setCurrentIndex(1);
00143     }
00144     else
00145     {
00146         setFixedSize(gridLayout->sizeHint());
00147         stackedWidget->setCurrentIndex(0);
00148     }
00149     QWidget::show();
00150 }
```

8.8.3.16 specialKeyClicked

```
void Keyboard::specialKeyClicked (
             int key )  [signal]
```

Référencé par buttonClicked(), et Keyboard().

8.8.4 Documentation des données membres

8.8.4.1 buttonEsc

```
QPushButton* Keyboard::buttonEsc  [private]
```

Définition à la ligne 35 du fichier Keyboard.h.

Référencé par init().

8.8.4.2 capsLock

```
bool Keyboard::capsLock  [private]
```

Définition à la ligne 36 du fichier Keyboard.h.

Référencé par buttonClicked(), keyToCharacter(), makeCapsLock(), et setLineEdit().

8.8.4.3 gridLayout

```
QGridLayout* Keyboard::gridLayout  [private]
```

Définition à la ligne 31 du fichier Keyboard.h.

Référencé par buttonClicked(), init(), makeCapsLock(), et showKeyboard().

8.8.4.4 hBoxLayout

```
QHBoxLayout* Keyboard::hBoxLayout  [private]
```

Définition à la ligne 32 du fichier Keyboard.h.

Référencé par buttonClicked(), init(), et showKeyboard().

8.8.4.5 instance

```
Keyboard * Keyboard::instance = Q_NULLPTR  [static], [private]
```

Définition à la ligne 28 du fichier Keyboard.h.

8.8.4.6 lineEdit

```
QLineEdit* Keyboard::lineEdit  [private]
```

Définition à la ligne 34 du fichier Keyboard.h.

Référencé par buttonClicked(), focusChange(), Keyboard(), keyboardKeyClicked(), keyboardSpecialKeyClicked(), et setLineEdit().

8.8.4.7 mapper

```
QSignalMapper* Keyboard::mapper  [private]
```

Définition à la ligne 33 du fichier Keyboard.h.

Référencé par init(), et makeCapsLock().

8.8.4.8 mask

```
bool Keyboard::mask  [private]
```

Définition à la ligne 37 du fichier Keyboard.h.

Référencé par buttonClicked(), setLineEdit(), et showKeyboard().

8.8.4.9 stackedWidget

```
QStackedWidget* Keyboard::stackedWidget  [private]
```

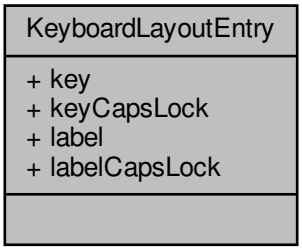
Définition à la ligne 30 du fichier Keyboard.h.

Référencé par buttonClicked(), init(), et showKeyboard().

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

8.9 Référence de la structure KeyboardLayoutEntry

Graphe de collaboration de KeyboardLayoutEntry :



Attributs publics

- int key
- int keyCapsLock
- const char * label
- const char * labelCapsLock

8.9.1 Description détaillée

Définition à la ligne 18 du fichier Keyboard.cpp.

8.9.2 Documentation des données membres

8.9.2.1 key

```
int KeyboardLayoutEntry::key
```

Définition à la ligne 20 du fichier Keyboard.cpp.

8.9.2.2 keyCapsLock

```
int KeyboardLayoutEntry::keyCapsLock
```

Définition à la ligne 22 du fichier Keyboard.cpp.

8.9.2.3 label

```
const char* KeyboardLayoutEntry::label
```

Définition à la ligne 21 du fichier Keyboard.cpp.

8.9.2.4 labelCapsLock

```
const char* KeyboardLayoutEntry::labelCapsLock
```

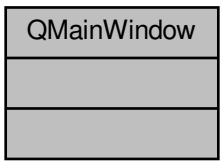
Définition à la ligne 23 du fichier Keyboard.cpp.

La documentation de cette structure a été générée à partir du fichier suivant :

— Keyboard.cpp

8.10 Référence de la classe QMainWindow

Graphe de collaboration de QMainWindow :

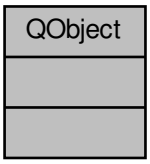


La documentation de cette classe a été générée à partir du fichier suivant :

— Ihm.h

8.11 Référence de la classe QObject

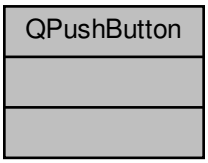
Graphe de collaboration de QObject :



La documentation de cette classe a été générée à partir du fichier suivant : Rfid.h

8.12 Référence de la classe QPushButton

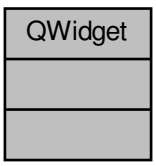
Graphe de collaboration de QPushButton :



La documentation de cette classe a été générée à partir du fichier suivant : Casier.h

8.13 Référence de la classe QWidget

Graphe de collaboration de QWidget :



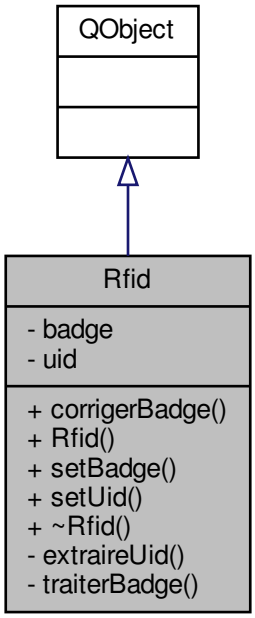
La documentation de cette classe a été générée à partir du fichier suivant : Keyboard.h

8.14 Référence de la classe Rfid

La classe Rfid traite la trame reçue d'un lecteur Rfid.

`#include <Rfid.h>`

Graphe de collaboration de Rfid :



Signaux

- void erreurBadgeInvalide (QString message)
- void nouveauUidBadge (QString badge)

Fonctions membres publiques

- QString corrigerBadge (QString badge)
  Définition de la méthode corrigerBadge(QString badge)
- Rfid (QObject *parent=nullptr)
  Définition du constructeur de la classe Rfid.
- void setBadge (QString badge)
  Définition de la méthode setBadge(QString badge)
- void setUid (QString uid)
  Définition de la méthode setUid(QString uid)
- ~Rfid ()
  Définition du destructeur de la classe Rfid.

Connecteurs privés

- void traiterBadge (QString trameBadge)
  Définition de la méthode Rfid : :traiterBadge(QString trameBadge)

Fonctions membres privées

- void extraireUid ()
  Définition de la méthode Rfid : :extraireUid()

Attributs privés

- QString badge
  trame reçue d'un badge
- QString uid
  l'UID extrait de la trame badge

8.14.1 Description détaillée

La classe Rfid traite la trame reçue d'un lecteur Rfid.

Auteur

Legger Pierre-Antoine

Version

1.0

Date

Mercredi 4 Mars 2020

Définition à la ligne 35 du fichier Rfid.h.

8.14.2 Documentation des constructeurs et destructeur

8.14.2.1 Rfid()

```
Rfid::Rfid (
            QObject * parent = nullptr )
```

Définition du constructeur de la classe Rfid.

Initialise un objet Rfid

Paramètres

parent	l'objet QObject parent

Définition à la ligne 22 du fichier Rfid.cpp.

```
00022                         : QObject(parent), badge("")
00023 {
00024 }
```

8.14.2.2 ~Rfid()

```
Rfid::~Rfid ( )
```

Définition du destructeur de la classe Rfid.

Détruitun objet Rfid

Définition à la ligne 31 du fichier Rfid.cpp.

```
00032 {
00033
00034 }
```

8.14.3 Documentation des fonctions membres

8.14.3.1 corrigerBadge()

```
QString Rfid::corrigerBadge (
            QString badge )
```

Définition de la méthode corrigerBadge(QString badge)

La trame reçue provient de l'émulation d'un clavier QWERTY qu'il faut traduire en AZERTY

Paramètres

badge	la trame badge en format QWERTY

Renvoie

la trame badge en format AZERTY

Définition à la ligne 43 du fichier Rfid.cpp.

Référencé par traiterBadge().

```
00044 {
00045     QString badgeCorrige = "";
00046
00047     if(!badge.isEmpty())
00048     {
00049         // effectue les remplacements des touches QWERTY en touches AZERTY
00050         badgeCorrige = badge.replace(QString::fromUtf8("Q"), "A");
00051         badgeCorrige = badge.replace(QString::fromUtf8("W"), "Z");
00052         badgeCorrige = badge.replace(QString::fromUtf8("q"), "q");
00053         badgeCorrige = badge.replace(QString::fromUtf8("w"), "z");
00054         badgeCorrige = badge.replace(QString::fromUtf8("M"), ":");
00055         badgeCorrige = badge.replace(QString::fromUtf8("à"), "0");
00056         badgeCorrige = badge.replace(QString::fromUtf8("&"), "1");
00057         badgeCorrige = badge.replace(QString::fromUtf8("é"), "2");
00058         badgeCorrige = badge.replace(QString::fromUtf8("\""), "3");
00059         badgeCorrige = badge.replace(QString::fromUtf8("'"), "4");
00060         badgeCorrige = badge.replace(QString::fromUtf8("("), "5");
00061         badgeCorrige = badge.replace(QString::fromUtf8("-"), "6");
00062         badgeCorrige = badge.replace(QString::fromUtf8("è"), "7");
00063         badgeCorrige = badge.replace(QString::fromUtf8("_"), "8");
00064         badgeCorrige = badge.replace(QString::fromUtf8("ç"), "9");
00065     }
00066     return badgeCorrige;
00067 }
```

8.14.3.2 erreurBadgeInvalide

```
void Rfid::erreurBadgeInvalide (
            QString message )  [signal]
```

Référencé par traiterBadge().

8.14.3.3 extraireUid()

```
void Rfid::extraireUid ( )  [private]
```

Définition de la méthode Rfid : :extraireUid()

Extrait l'UID de la trame badge

Définition à la ligne 130 du fichier Rfid.cpp.

Références badge, et setUid().

Référencé par traiterBadge().

```
00131 {
00132     setUid(badge.section(':',1,1));
00133 }
```

8.14.3.4 nouveauUidBadge

```
void Rfid::nouveauUidBadge (
            QString badge )  [signal]
```

Référencé par traiterBadge().

8.14.3.5 setBadge()

```
void Rfid::setBadge (
            QString badge )
```

Définition de la méthode setBadge(QString badge)

Accesseur set de l'attribut badge

Paramètres

badge	

Définition à la ligne 75 du fichier Rfid.cpp.

Références badge.

Référencé par traiterBadge().

```
00076 {
00077     this->badge = badge;
00078 }
```

8.14.3.6 setUid()

```
void Rfid::setUid (
            QString uid )
```

Définition de la méthode setUid(QString uid)

Accesseur set de l'attribut uid

Paramètres

uid	

Définition à la ligne 86 du fichier Rfid.cpp.

Références uid.

Référencé par extraireUid().

```
00087 {
00088     this->uid = uid;
00089 }
```

8.14.3.7 traiterBadge

```
void Rfid::traiterBadge (
            QString trameBadge )  [private], [slot]
```

Définition de la méthode Rfid : :traiterBadge(QString trameBadge)

Extrait l'UID d'une trame badge valide

Paramètres

trameBadge	la trame reçue du lecteur Rfid

Définition à la ligne 97 du fichier Rfid.cpp.

Références badge, corrigerBadge(), ERREUR_BADGE_INVALIDE, erreurBadgeInvalide(), extraireUid(), nouveauUidBadge(), setBadge(), et uid.

```
00098 {
00099     /*
00100      * Format trame reçue : RFID:xxxxxxxx
00101      * xxxxxxxx -> uid du badge
00102      */
00103     setBadge(corrigerBadge(trameBadge));
00104
00105     #ifdef DEBUG_RFID
```

```
00106         qDebug() << Q_FUNC_INFO << "Badge" << badge;
00107     #endif
00108
00109     // Vérifier si la trame est valide
00110     if(badge.startsWith("RFID:"))
00111     {
00112         extraireUid();
00113
00114         emit nouveauUidBadge(uid);
00115         #ifdef DEBUG_RFID
00116             qDebug() << Q_FUNC_INFO << "UID" << uid;
00117         #endif
00118     }
00119     else
00120     {
00121         emit erreurBadgeInvalide(ERREUR_BADGE_INVALIDE);
00122     }
00123 }
```

8.14.4 Documentation des données membres

8.14.4.1 badge

```
QString Rfid::badge  [private]
```

trame reçue d'un badge

Définition à la ligne 56 du fichier Rfid.h.

Référencé par extraireUid(), setBadge(), et traiterBadge().

8.14.4.2 uid

```
QString Rfid::uid  [private]
```

l'UID extrait de la trame badge

Définition à la ligne 57 du fichier Rfid.h.

Référencé par setUid(), et traiterBadge().

La documentation de cette classe a été générée à partir des fichiers suivants :

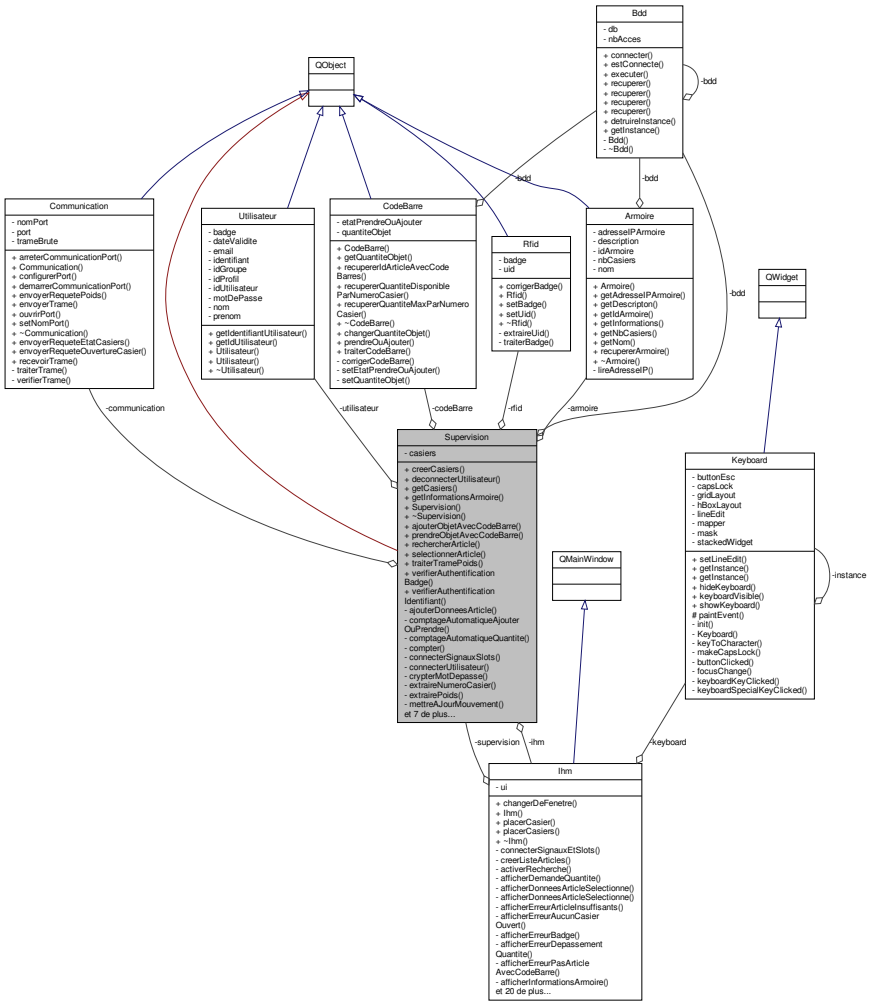
- Rfid.h
- Rfid.cpp

8.15 Référence de la classe Supervision

La classe Supervision permet de superviser l'ensemble de l'application.

`#include <Supervision.h>`

Graphe de collaboration de Supervision :



Connecteurs publics

- void ajouterObjetAvecCodeBarre (QString codeBarre)
  Définition de la méthode ajouterObjetAvecCodeBarre.
- void prendreObjetAvecCodeBarre (QString codeBarre)
  Définition de la méthode prendreObjetAvecCodeBarre.
- void rechercherArticle (QString recherche)
  Définition de la méthode rechercherArticle.
- void selectionnerArticle (QString nomArticle)
  Définition de la méthode selectionnerArticle.
- void traiterTramePoids (QString trame)
  Définition de la méthode traiterTramePoids.
- void verifierAuthentificationBadge (QString badge)
  Définition de la méthode verifierAuthentificationBadge.
- void verifierAuthentificationIdentifiant (QString identifiant, QString motDePasse)
  Définition de la méthode verifierAuthentificationIdentifiant.

Signaux

- void articlesTrouves (QVector< QStringList >)
- void donneesArticleSelectionne (QVector< QStringList >)
- void donneesArticleSelectionne (QStringList)
- void erreurArticleInsuffisants ()
- void erreurAucunArticleAvecCodeBarre ()
- void erreurAucunCasierOuvert ()
- void erreurDepassementQuantite ()
- void erreurQuantiteTropElevee ()
- void reponseDemandeDeConnexion (bool, QString)

Fonctions membres publiques

- void creerCasiers ()
  Définition de la méthode creerCasiers.
- void deconnecterUtilisateur ()
  Méthode qui permet la déconnexion de l'utilisateur.
- QVector< Casier * > getCasiers ()
  Définition de la méthode getCasiers.
- QStringList getInformationsArmoire ()
  Définition de la méthode getInformationsArmoire.
- Supervision (Ihm *parent=nullptr)
  Définition du constructeur de la classe Supervision.
- ~Supervision ()
  Définition du destructeur de Supervision.

Fonctions membres privées

- void ajouterDonneesArticle (Article *article, QVector< QStringList > &donneesArticle, QStringList &donnees)
  Définition de la méthode ajouterDonneesArticle.
- unsigned int comptageAutomatiqueAjouterOuPrendre (QString nbArticleAvant, int nbArticleApres)
  Définition de la méthode comptageAutomatiqueAjouterOuPrendre.
- unsigned int comptageAutomatiqueQuantite (QString nbArticleAvant, int nbArticleApres)
  Définition de la méthode comptageAutomatiqueQuantite.
- int compter (QString poidArticle, QString poidTotal, QString tare)
  Définition de la méthode compter.
- void connecterSignauxSlots ()
  Définition de la méthode connecterSignauxSlots.
- void connecterUtilisateur (QStringList &donnees)
  Définition de la méthode connecterUtilisateur.
- void crypterMotDepasse (QString &motDePasse)
  Définition de la méthode crypterMotDepasse.
- QString extraireNumeroCasier (QString trame)
  Définition de la méthode extraireNumeroCasier.
- QString extrairePoids (QString trame)
  Définition de la méthode extrairePoids.
- void mettreAJourMouvement (QString idUtilisateur, QString idStock, QString idAction, QString quantite)
  Définition de la méthode mettreAJourMouvement.
- int rechercherCasierOuvert ()
  Définition de la méthode rechercherCasierOuvert.
- QStringList recupererDonneesUtilisateur (QString requeteBDD)
  Définition de la méthode recupererDonneesUtilisateur.
- QString recupererHorodatage ()
  Définition de la méthode recupererHorodatage.
- QString recupererIdStockAvecNumeroCasier (int numeroCasier)
  Définition de la méthode recupererIdStockAvecNumeroCasier.
- bool verifierArticlePresentDansCasier (QString numCasier, QString idArticle)
  Définition de la méthode verifierArticlePresentDansCasier.
- bool verifierDateValidite (QString stringDateValidite)
  Définition de la méthode verifierDateValidite.
- bool verifierDonneesUtilisateur (QStringList &donnees)
  Définition de la méthode verifierDonneesUtilisateur.

Attributs privés

- Armoire * armoire
  association d'un objet Armoire
- Bdd * bdd
  association d'un objet Bdd (accès à la base de données)
- QVector< Casier * > casiers
  les casiers de l'armoire
- CodeBarre * codeBarre
  association d'un objet CodeBarre
- Communication * communication
  association d'un objet Communication
- Ihm * ihm
  association d'un objet Ihm (fenêtre principale de l'application)
- Rfid * rfid
  association d'un objet Rfid (le lecteur de badge)
- Utilisateur * utilisateur
  association d'un objet Utilisateur (l'utilisateur authentifié)

8.15.1 Description détaillée

La classe Supervision permet de superviser l'ensemble de l'application.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition à la ligne 52 du fichier Supervision.h.

8.15.2 Documentation des constructeurs et destructeur

8.15.2.1 Supervision()

```
Supervision::Supervision (
            Ihm * parent = nullptr )
```

Définition du constructeur de la classe Supervision.

Initialise la supervision

Paramètres

parent	l'objet QObject parent

Définition à la ligne 36 du fichier Supervision.cpp.

Références armoire, bdd, codeBarre, communication, Bdd : :connecter(), connecterSignauxSlots(), Communication : :demarrer↩ CommunicationPort(), Bdd : :getInstance(), rfid, et utilisateur.

```
00036                                   : QObject(parent), ihm(parent)
00037 {
00038     // Instancie les objets dont la classe Supervision coordonne les actions
00039     bdd = Bdd::getInstance();
00040     bdd->connecter();
00041     codeBarre = new CodeBarre(this);
00042     //portSerie = new Communication(this);
00043     rfid = new Rfid(this);
00044     utilisateur = nullptr;
00045     armoire = new Armoire(this);
00046     communication = new Communication(this);
00047
00048     connecterSignauxSlots();
00049
00050     communication->demarrerCommunicationPort();
00051
00052 }
```

8.15.2.2 ~Supervision()

```
Supervision::~Supervision ( )
```

Définition du destructeur de Supervision.

Détruit un objet Supervision

Définition à la ligne 59 du fichier Supervision.cpp.

```
00060 {
00061
00062 }
```

8.15.3 Documentation des fonctions membres

8.15.3.1 ajouterDonneesArticle()

```
void Supervision::ajouterDonneesArticle (
            Article * article,
            QVector< QStringList > & donneesArticle,
            QStringList & donnees )  [private]
```

Définition de la méthode ajouterDonneesArticle.

permet d'ajouter des données d'un article d'un casier

Paramètres

article	
donneesArticle	
donnees	

Définition à la ligne 579 du fichier Supervision.cpp.

Références Article : :get(), TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_NUMERO_CASIER, et TABLE_ARTICLE_QUANTITE.

Référencé par selectionnerArticle().

```
00580 {
00581     donnees << article->get(TABLE_ARTICLE_QUANTITE);
00582     donnees << article->get(TABLE_ARTICLE_DISPONIBLE);
00583     donnees << article->get(TABLE_ARTICLE_NUMERO_CASIER);
00584
00585     donneesArticle.push_back(donnees);
00586     donnees.clear();
00587 }
```

8.15.3.2 ajouterObjetAvecCodeBarre

```
void Supervision::ajouterObjetAvecCodeBarre (
             QString codeBarre )  [slot]
```

Définition de la méthode ajouterObjetAvecCodeBarre.

méthode qui permet d'ajouter un objet avec son code barres

Paramètres

QString	codeBarre

Définition à la ligne 669 du fichier Supervision.cpp.

Références bdd, casiers, erreurAucunArticleAvecCodeBarre(), erreurAucunCasierOuvert(), erreurQuantiteTropElevee(), Bdd : :executer(), Utilisateur : :getIdUtilisateur(), mettreAJourMouvement(), rechercherCasierOuvert(), recupererIdStockAvecNumeroCasier(), utilisateur, et verifierArticlePresentDansCasier().

Référencé par connecterSignauxSlots().

```
00670 {
00671     int numeroCasier = rechercherCasierOuvert();
00672
00673     if(numeroCasier == -1)
00674         return;
00675
00676     #ifdef DEBUG_SUPERVISION
00677         qDebug() << Q_FUNC_INFO << "codeBarre" << codeBarre << "casier" << numeroCasier;
00678     #endif
00679
00680     unsigned int quantiteMax = this->codeBarre->
      recupererQuantiteMaxParNumeroCasier(QString::number(
      casiers[numeroCasier]->getNumero()));
00681     unsigned int quantiteDisponible = this->codeBarre->
      recupererQuantiteDisponibleParNumeroCasier(QString::number(
      casiers[numeroCasier]->getNumero()));
00682     unsigned int quantite = quantiteDisponible + this->codeBarre->
      getQuantiteObjet();
00683     QString idArticle = QString::number(this->codeBarre->
      recupererIdArticleAvecCodeBarres(codeBarre));
00684
00685     if(casiers[numeroCasier]->estOuvert())
00686     {
00687         if(verifierArticlePresentDansCasier(QString::number(
      casiers[numeroCasier]->getNumero()), idArticle))
00688         {
00689             if(quantite <= quantiteMax)
00690             {
00691                 QString strQuantite = QString::number(quantite);
00692                 QString requete = "UPDATE Stock SET Disponible = '" +  strQuantite + "' WHERE
       Stock.idArticle = '" + idArticle + "'";
00693                 bdd->executer(requete);
```

```
00694                 QString idUtilisateur = utilisateur->getIdUtilisateur();
00695                 QString idStock = recupererIdStockAvecNumeroCasier(
      numeroCasier+1);
00696                 QString idAction = "2";
00697                 QString quantiteMouvement = QString::number(this->codeBarre->
      getQuantiteObjet());
00698                 mettreAJourMouvement(idUtilisateur, idStock, idAction,
      quantiteMouvement);
00699             }
00700             else
00701             {
00702                 emit erreurQuantiteTropElevee();
00703             }
00704         }
00705         else
00706         {
00707             emit erreurAucunArticleAvecCodeBarre();
00708         }
00709     }
00710     else
00711     {
00712         erreurAucunCasierOuvert();
00713     }
00714 }
```

8.15.3.3 articlesTrouves

```
void Supervision::articlesTrouves (
             QVector< QStringList >  )  [signal]
```

Référencé par connecterSignauxSlots(), et rechercherArticle().

8.15.3.4 comptageAutomatiqueAjouterOuPrendre()

```
unsigned int Supervision::comptageAutomatiqueAjouterOuPrendre (
             QString nbArticleAvant,
             int nbArticleApres )  [private]
```

Définition de la méthode comptageAutomatiqueAjouterOuPrendre.

permet de savoir si l'action effectuer est un retrait ou un ajout

Paramètres

QString	nbArticleAvant
int	nbArticleApres

Renvoie

unsigned int action

Définition à la ligne 443 du fichier Supervision.cpp.

Référencé par traiterTramePoids().

```
00444 {
00445     int intNbArticleAvant = nbArticleAvant.toInt();
00446
00447     if(intNbArticleAvant > nbArticleApres)
00448     {
00449         return 2;
```

```
00450     }
00451     else
00452     {
00453         return 1;
00454     }
00455 }
```

8.15.3.5 comptageAutomatiqueQuantite()

```
unsigned int Supervision::comptageAutomatiqueQuantite (
            QString nbArticleAvant,
            int nbArticleApres )  [private]
```

Définition de la méthode comptageAutomatiqueQuantite.

permet de savoir combien d'article il y à d'article de différence

Paramètres

QString	nbArticleAvant
int	nbArticleApres

Renvoie

unsigned int quantite

Définition à la ligne 464 du fichier Supervision.cpp.

Référencé par traiterTramePoids().

```
00465 {
00466     int intNbArticleAvant = nbArticleAvant.toInt();
00467
00468     if(intNbArticleAvant > nbArticleApres)
00469     {
00470         return intNbArticleAvant - nbArticleApres;
00471     }
00472     else
00473     {
00474         return nbArticleApres - intNbArticleAvant;
00475     }
00476 }
```

8.15.3.6 compter()

```
int Supervision::compter (
            QString poidsArticle,
            QString poidsTotal,
            QString tare )  [private]
```

Définition de la méthode compter.

assure le comptage automatique du nombre d'article présent dans le casier

Paramètres

poidsArticle	le poids total dans le casier
poidsTotal	le poids d'un article
tare	

Renvoie

la quantite d'article présent dans le casier sous forme d'un entier

Définition à la ligne 522 du fichier Supervision.cpp.

Références PRECISION.

Référencé par traiterTramePoids().

```
00523 {
00524     double doublePoidsArticle = poidsArticle.toDouble();
00525     double doublePoidsTotal = poidsTotal.toDouble();
00526     double doubleTare = tare.toDouble();
00527 
00528     //comptage du nombre d'articles
00529 
00530     double doubleNombreArticle = qRound((doublePoidsTotal - doubleTare) / doublePoidsArticle);
00531     QString strNombreArticle = QString::number(doubleNombreArticle, 'f',
      PRECISION);
00532     int nombreArticle = strNombreArticle.toInt();
00533 
00534     #ifdef DEBUG_SUPERVISION
00535         qDebug() << Q_FUNC_INFO << "nombreArticle:" << nombreArticle;
00536     #endif
00537 
00538     return nombreArticle;
00539 }
```

8.15.3.7 connecterSignauxSlots()

```
void Supervision::connecterSignauxSlots ( )  [private]
```

Définition de la méthode connecterSignauxSlots.

Etablie la connexion entre les diffrents signaux et slots

Définition à la ligne 282 du fichier Supervision.cpp.

Références ajouterObjetAvecCodeBarre(), armoire, articlesTrouves(), codeBarre, communication, donneesArticleSelectionne(), erreurArticleInsuffisants(), erreurAucunArticleAvecCodeBarre(), erreurAucunCasierOuvert(), erreurDepassementQuantite(), erreur↩QuantiteTropElevee(), ihm, prendreObjetAvecCodeBarre(), rechercherArticle(), reponseDemandeDeConnexion(), rfid, selectionner↩Article(), traiterTramePoids(), verifierAuthentificationBadge(), et verifierAuthentificationIdentifiant().

Référencé par Supervision().

```
00283 {
00284     // Armoire
00285     connect(armoire, SIGNAL(informationsArmoire(QStringList)), ihm, SLOT(
      afficherInformationsArmoire(QStringList)));
00286 
00287     // Authentification Badge
00288     connect(ihm, SIGNAL(badgeDetecte(QString)), rfid, SLOT(traiterBadge(QString)));
00289     connect(rfid, SIGNAL(erreurBadgeInvalide(QString)), ihm, SLOT(afficherErreurBadge(QString)));
00290     connect(rfid, SIGNAL(nouveauUidBadge(QString)), this, SLOT(
      verifierAuthentificationBadge(QString)));
00291 
00292     // Authentification Identifiant
00293     connect(ihm, SIGNAL(identifiantDetecte(QString, QString)), this, SLOT(
```

```
    verifierAuthentificationIdentifiant(QString, QString)));
00294
00295     // Authentification Utilisateur
00296     connect(this, SIGNAL(reponseDemandeDeConnexion(bool,QString)),
    ihm, SLOT(traiterDemandeDeConnexion(bool,QString)));
00297
00298     // Article
00299     connect(communication, SIGNAL(envoieTramePoids(QString)), this, SLOT(
    traiterTramePoids(QString)));
00300     connect(ihm, SIGNAL(rechercheArticle(QString)), this, SLOT(
    rechercherArticle(QString)));
00301     connect(this, SIGNAL(articlesTrouves(QVector<QStringList>)),
    ihm, SLOT(mettreAJourListeArticles(QVector<QStringList>)));
00302     connect(ihm, SIGNAL(articleSelectionne(QString)), this, SLOT(
    selectionnerArticle(QString)));
00303     connect(this, SIGNAL(donneesArticleSelectionne(QStringList)),
    ihm, SLOT(afficherDonneesArticleSelectionne(QStringList)));
00304     connect(this, SIGNAL(donneesArticleSelectionne(QVector<QStringList>)),
    ihm, SLOT(afficherDonneesArticleSelectionne(QVector<QStringList>)));
00305     connect(this, SIGNAL(erreurDepassementQuantite()),
    ihm, SLOT(afficherErreurDepassementQuantite()));
00306
00307     // CodeBarre
00308     connect(ihm, SIGNAL(boutonPrendre(bool)), codeBarre, SLOT(prendreOuAjouter(bool)));
00309     connect(ihm, SIGNAL(boutonAjouter(bool)), codeBarre, SLOT(prendreOuAjouter(bool)));
00310     connect(ihm, SIGNAL(codeBarreObjetScanner(QString)), codeBarre, SLOT(traiterCodeBarre(
    QString)));
00311     connect(this, SIGNAL(erreurAucunArticleAvecCodeBarre()),
    ihm, SLOT(afficherErreurPasArticleAvecCodeBarre()));
00312     connect(codeBarre, SIGNAL(prendreObjet(QString)), this, SLOT(
    prendreObjetAvecCodeBarre(QString)));
00313     connect(codeBarre, SIGNAL(ajouterObjet(QString)), this, SLOT(
    ajouterObjetAvecCodeBarre(QString)));
00314     connect(ihm, SIGNAL(envoyerQuantite(int)), codeBarre, SLOT(changerQuantiteObjet(int)));
00315     connect(this, SIGNAL(erreurArticleInsuffisants()),
    ihm, SLOT(afficherErreurArticleInsuffisants()));
00316     connect(this, SIGNAL(erreurQuantiteTropElevee()),
    ihm, SLOT(afficherErreurDepassementQuantite()));
00317     connect(this, SIGNAL(erreurAucunCasierOuvert()), ihm, SLOT(
    afficherErreurAucunCasierOuvert()));
00318 }
```

8.15.3.8 connecterUtilisateur()

```
void Supervision::connecterUtilisateur (
            QStringList & donnees )  [private]
```

Définition de la méthode connecterUtilisateur.

Connecte l'utilisateur et le supprime si il en existe un

Paramètres

donnees	Plusieurs chaînes de caractères des données utilisateur

Définition à la ligne 260 du fichier Supervision.cpp.

Références deconnecterUtilisateur(), Utilisateur : :getIdentifiantUtilisateur(), traiterTramePoids(), et utilisateur.

Référencé par verifierAuthentificationBadge(), et verifierAuthentificationIdentifiant().

```
00261 {
00262     if(utilisateur != nullptr)
00263     {
00264         deconnecterUtilisateur();
00265     }
00266     utilisateur = new Utilisateur(donnees, this);
00267     #ifdef DEBUG_SUPERVISION
00268         qDebug() << Q_FUNC_INFO << utilisateur->
    getIdentifiantUtilisateur() << "authentifié";
00269     #endif
00270
```

```
00271     #ifdef SUPERVISION_TEST_POIDS
00272         QString trameTest = "CASIERS;3;2;1745";
00273         traiterTramePoids(trameTest);
00274     #endif
00275 }
```

8.15.3.9 creerCasiers()

```
void Supervision::creerCasiers ( )
```

Définition de la méthode creerCasiers.

Méthode qui crée les casiers à gérer

Définition à la ligne 83 du fichier Supervision.cpp.

Références armoire, casiers, FENETRE_MENU, Armoire : :getNbCasiers(), ihm, et Ihm : :placerCasiers().

Référencé par Ihm : :Ihm().

```
00084 {
00085     QString nbCasiers = armoire->getNbCasiers();
00086     qDebug() << Q_FUNC_INFO << "nbCasiers" << nbCasiers;
00087     if(!nbCasiers.isEmpty())
00088     {
00089         for(int i=0; i < nbCasiers.toInt(); i++)
00090         {
00091             Casier* casier = new Casier(i+1, ihm);
00092             connect(casier, SIGNAL(estOuvert(int,bool)), ihm, SLOT(gererPageScanObjet(int,bool)));
00093             casiers.push_back(casier);
00094         }
00095     }
00096     ihm->placerCasiers(casiers, FENETRE_MENU);
00097 }
```

8.15.3.10 crypterMotDepasse()

```
void Supervision::crypterMotDepasse (
            QString & motDePasse )  [private]
```

Définition de la méthode crypterMotDepasse.

Crypte le mots de passe avec la méthode Md5 puis vers l'hexadécimal

Paramètres

motDePasse	Chaîne de caractères du mot de passe

Définition à la ligne 183 du fichier Supervision.cpp.

Référencé par verifierAuthentificationIdentifiant().

```
00184 {
00185     if(!motDePasse.isEmpty())
00186     {
00187     motDePasse = QString(QCryptographicHash::hash((motDePasse).toLatin1(), QCryptographicHash::Md5).toHex()
      );
00188     }
```

```
00189
00190     #ifdef DEBUG_SUPERVISION
00191         qDebug() << Q_FUNC_INFO << "Mot de passe crypte" << motDePasse;
00192     #endif
00193 }
```

8.15.3.11 deconnecterUtilisateur()

```
void Supervision::deconnecterUtilisateur ( )
```

Méthode qui permet la déconnexion de l'utilisateur.

Supprime les données de l'utilisateur

Définition à la ligne 69 du fichier Supervision.cpp.

Références utilisateur.

Référencé par connecterUtilisateur(), et Ihm : :deconnecterUtilisateur().

```
00070 {
00071     if(utilisateur != nullptr)
00072     {
00073         delete utilisateur;
00074         utilisateur = nullptr;
00075     }
00076 }
```

8.15.3.12 donneesArticleSelectionne [1/2]

```
void Supervision::donneesArticleSelectionne (
            QVector< QStringList >  )  [signal]
```

Référencé par connecterSignauxSlots(), et selectionnerArticle().

8.15.3.13 donneesArticleSelectionne [2/2]

```
void Supervision::donneesArticleSelectionne (
            QStringList  )  [signal]
```

8.15.3.14 erreurArticleInsuffisants

```
void Supervision::erreurArticleInsuffisants ( )  [signal]
```

Référencé par connecterSignauxSlots(), et prendreObjetAvecCodeBarre().

8.15.3.15 erreurAucunArticleAvecCodeBarre

```
void Supervision::erreurAucunArticleAvecCodeBarre ( )  [signal]
```

Référencé par ajouterObjetAvecCodeBarre(), connecterSignauxSlots(), et prendreObjetAvecCodeBarre().

8.15.3.16 erreurAucunCasierOuvert

```
void Supervision::erreurAucunCasierOuvert ( )  [signal]
```

Référencé par ajouterObjetAvecCodeBarre(), connecterSignauxSlots(), et prendreObjetAvecCodeBarre().

8.15.3.17 erreurDepassementQuantite

```
void Supervision::erreurDepassementQuantite ( )  [signal]
```

Référencé par connecterSignauxSlots(), et traiterTramePoids().

8.15.3.18 erreurQuantiteTropElevee

```
void Supervision::erreurQuantiteTropElevee ( )  [signal]
```

Référencé par ajouterObjetAvecCodeBarre(), et connecterSignauxSlots().

8.15.3.19 extraireNumeroCasier()

```
QString Supervision::extraireNumeroCasier (
            QString trame )  [private]
```

Définition de la méthode extraireNumeroCasier.

extrait le numéro de casier de la trame

Paramètres

trame	

Renvoie

le numéro du casier

Définition à la ligne 503 du fichier Supervision.cpp.

Référencé par traiterTramePoids().

```
00504 {
00505     QString numCasier = trame.section(';',2,2);
```

```
00506
00507     #ifdef DEBUG_SUPERVISION
00508         qDebug() << Q_FUNC_INFO << "numCasier:" << numCasier;
00509     #endif
00510
00511     return numCasier;
00512 }
```

8.15.3.20 extrairePoids()

```
QString Supervision::extrairePoids (
            QString trame )  [private]
```

Définition de la méthode extrairePoids.

extrait le poids de l'article de la trame

Paramètres

trame	

Renvoie

le poids de l'article sous forme d'un QString

Définition à la ligne 485 du fichier Supervision.cpp.

Référencé par traiterTramePoids().

```
00486 {
00487     QString poids = trame.section(';',3,3);
00488
00489     #ifdef DEBUG_SUPERVISION
00490         qDebug() << Q_FUNC_INFO << "poids:" << poids;
00491     #endif
00492
00493     return poids;
00494 }
```

8.15.3.21 getCasiers()

```
QVector< Casier * > Supervision::getCasiers ( )
```

Définition de la méthode getCasiers.

Renvoie les casiers

Renvoie

QVector<Casier*> casiers

Définition à la ligne 117 du fichier Supervision.cpp.

Références casiers.

Référencé par Ihm : :allerFenetreMenu(), et Ihm : :allerFenetreScannerObjet().

```
00118 {
00119     return casiers;
00120 }
```

8.15.3.22 getInformationsArmoire()

```
QStringList Supervision::getInformationsArmoire ( )
```

Définition de la méthode getInformationsArmoire.

Récupère les informations (nom, ...) sur l'armoire

Renvoie

les informations (nom, ...) sur l'armoire sous la forme d'un QStringList

Définition à la ligne 105 du fichier Supervision.cpp.

Références armoire, et Armoire : :getInformations().

Référencé par Ihm : :Ihm().

```
00106 {
00107     QStringList informationsArmoire = armoire->getInformations();
00108
00109     return informationsArmoire;
00110 }
```

8.15.3.23 mettreAJourMouvement()

```
void Supervision::mettreAJourMouvement (
            QString idUtilisateur,
            QString idStock,
            QString idAction,
            QString quantite )  [private]
```

Définition de la méthode mettreAJourMouvement.

permet de mettre à jour les mouvements

Paramètres

QString	idUtilisateur
QString	idStock
QString	idAction
QString	quantite

Définition à la ligne 565 du fichier Supervision.cpp.

Références bdd, Bdd : :executer(), et recupererHorodatage().

Référencé par ajouterObjetAvecCodeBarre(), prendreObjetAvecCodeBarre(), et traiterTramePoids().

```
00566 {
00567     QString horodatage = recupererHorodatage();
00568     QString requete ="INSERT INTO Mouvement(idUtilisateur, idStock, idAction, Quantite, Horodatage)
      VALUES('" + idUtilisateur + "', '" + idStock + "', '" + idAction + "', '" + quantite + "', '" + horodatage + "');";
00569     bdd->executer(requete);
00570 }
```

8.15.3.24 prendreObjetAvecCodeBarre

```
void Supervision::prendreObjetAvecCodeBarre (
            QString codeBarre )  [slot]
```

Définition de la méthode prendreObjetAvecCodeBarre.

méthode qui permet de prendre un objet avec son code barres

Paramètres

QString	codeBarre

Définition à la ligne 618 du fichier Supervision.cpp.

Références bdd, casiers, erreurArticleInsuffisants(), erreurAucunArticleAvecCodeBarre(), erreurAucunCasierOuvert(), Bdd : :executer(), Utilisateur : :getIdUtilisateur(), mettreAJourMouvement(), rechercherCasierOuvert(), recupererIdStockAvecNumeroCasier(), utilisateur, et verifierArticlePresentDansCasier().

Référencé par connecterSignauxSlots().

```
00619 {
00620     int numeroCasier = rechercherCasierOuvert();
00621
00622     if(numeroCasier == -1)
00623         return;
00624
00625     #ifdef DEBUG_SUPERVISION
00626         qDebug() << Q_FUNC_INFO << "codeBarre" << codeBarre << "casier" << numeroCasier;
00627     #endif
00628
00629     unsigned int quantiteDisponible = this->codeBarre->
      recupererQuantiteDisponibleParNumeroCasier(QString::number(
      casiers[numeroCasier]->getNumero()));
00630     int quantite = quantiteDisponible - this->codeBarre->
      getQuantiteObjet();
00631     QString idArticle = QString::number(this->codeBarre->
      recupererIdArticleAvecCodeBarres(codeBarre));
00632
00633     if(casiers[numeroCasier]->estOuvert())
00634     {
00635         if(verifierArticlePresentDansCasier(QString::number(
      casiers[numeroCasier]->getNumero()), idArticle))
00636         {
00637             if(quantite >= 0)
00638             {
00639                 QString strQuantite = QString::number(quantite);
00640                 QString requete = "UPDATE Stock SET Disponible = '" +  strQuantite + "' WHERE
       Stock.idArticle = '" + idArticle + "'";
00641                 bdd->executer(requete);
00642                 QString idUtilisateur = utilisateur->getIdUtilisateur();
00643                 QString idStock = recupererIdStockAvecNumeroCasier(
      numeroCasier+1 );
00644                 QString idAction = "1";
00645                 QString quantiteMouvement = QString::number(this->codeBarre->
      getQuantiteObjet());
00646                 mettreAJourMouvement(idUtilisateur, idStock, idAction,
      quantiteMouvement);
00647             }
00648             else
00649             {
00650                 erreurArticleInsuffisants();
00651             }
00652         }
00653         else
00654         {
00655             emit erreurAucunArticleAvecCodeBarre();
00656         }
00657     }
00658     else
00659     {
00660         erreurAucunCasierOuvert();
00661     }
00662 }
```

8.15.3.25 rechercherArticle

```
void Supervision::rechercherArticle (
            QString recherche )  [slot]
```

Définition de la méthode rechercherArticle.

Recherche un Article

Paramètres

recherche	

Définition à la ligne 326 du fichier Supervision.cpp.

Références articlesTrouves(), bdd, et Bdd : :recuperer().

Référencé par connecterSignauxSlots(), et Ihm : :rechercherArticle().

```
00327 {
00328     QString requete = "SELECT Stock.NumeroCasier, Article.idType, Article.Nom, Stock.Quantite,
      Stock.Disponible, Article.Designation FROM Stock INNER JOIN Article ON Stock.idArticle = Article.idArticle WHERE
      Article.Nom LIKE '%" + recherche + "%' OR Article.Code LIKE '%" + recherche + "%' OR Article.Designation LIKE '%" +
      recherche + "%' ORDER BY Stock.NumeroCasier ASC";
00329
00330     QVector<QStringList> listeArticlesTrouves;
00331     bdd->recuperer(requete, listeArticlesTrouves);
00332
00333     emit articlesTrouves(listeArticlesTrouves);
00334 }
```

8.15.3.26 rechercherCasierOuvert()

```
int Supervision::rechercherCasierOuvert ( )  [private]
```

Définition de la méthode rechercherCasierOuvert.

méthode pour rechercher le casier ouvert

Renvoie

int casierOuvert

Définition à la ligne 721 du fichier Supervision.cpp.

Références casiers.

Référencé par ajouterObjetAvecCodeBarre(), et prendreObjetAvecCodeBarre().

```
00722 {
00723     for(int i=0; i < casiers.size(); i++)
00724     {
00725         if(casiers[i]->estOuvert())
00726             return i;
00727     }
00728     return -1;
00729 }
```

8.15.3.27 recupererDonneesUtilisateur()

```
QStringList Supervision::recupererDonneesUtilisateur (
            QString requeteBDD )  [private]
```

Définition de la méthode recupererDonneesUtilisateur.

Récupére des données utilisateur dans la base de donnéess

Paramètres

requeteBDD	

Renvoie

La liste des données utilisateur

Définition à la ligne 169 du fichier Supervision.cpp.

Références bdd, et Bdd : :recuperer().

Référencé par verifierAuthentificationBadge(), et verifierAuthentificationIdentifiant().

```
00170 {
00171     QStringList donnees;
00172     bdd->recuperer(requeteBDD, donnees);
00173
00174     return donnees;
00175 }
```

8.15.3.28 recupererHorodatage()

```
QString Supervision::recupererHorodatage ( )  [private]
```

Définition de la méthode recupererHorodatage.

permet de récupérer la date est l'heure actuel

Renvoie

QString horodatage

Définition à la ligne 546 du fichier Supervision.cpp.

Référencé par mettreAJourMouvement().

```
00547 {
00548     QDate qDate(QDate::currentDate());
00549     QString date = qDate.toString("yyyy-MM-dd");
00550
00551     QTime time(QTime::currentTime());
00552     QString heure = time.toString("hh:mm:ss");
00553
00554     return date + " " + heure;
00555 }
```

8.15.3.29 recupererIdStockAvecNumeroCasier()

```
QString Supervision::recupererIdStockAvecNumeroCasier (
            int numeroCasier )  [private]
```

Définition de la méthode recupererIdStockAvecNumeroCasier.

méthode pour récupérer l'idStock avec un numéro de casier

Paramètres

int	numeroCasier

Renvoie

QString IdStock

Définition à la ligne 737 du fichier Supervision.cpp.

Références bdd, et Bdd : :recuperer().

Référencé par ajouterObjetAvecCodeBarre(), et prendreObjetAvecCodeBarre().

```
00738 {
00739     QString strNumeroCasier = QString::number(numeroCasier);
00740     QString requete = "SELECT idStock FROM Stock WHERE numeroCasier = " + strNumeroCasier + ";";
00741     QString donnees;
00742     bdd->recuperer(requete, donnees);
00743
00744     return donnees;
00745 }
```

8.15.3.30 reponseDemandeDeConnexion

```
void Supervision::reponseDemandeDeConnexion (
            bool ,
            QString  )  [signal]
```

Référencé par connecterSignauxSlots(), et verifierDonneesUtilisateur().

8.15.3.31 selectionnerArticle

```
void Supervision::selectionnerArticle (
            QString nomArticle )  [slot]
```

Définition de la méthode selectionnerArticle.

sélectionne un article

Paramètres

nomArticle	

Définition à la ligne 341 du fichier Supervision.cpp.

Références ajouterDonneesArticle(), donneesArticleSelectionne(), Article : :recupererDonneesArticleParNom(), Article : :recuperer↩ DonneesArticleParNumeroCasier(), Article : :recupererNombreCasiersPourNomArticle(), et Article : :recupererNumeroCasierPour↩ NomArticle().

Référencé par connecterSignauxSlots(), et Ihm : :selectionnerArticle().

```
00342 {
00343     #ifdef DEBUG_SUPERVISION
00344         qDebug() << Q_FUNC_INFO << "Nom article" << nomArticle;
00345     #endif
00346
00347     Article *article = new Article(this);
00348     QVector<QStringList> donneesArticle;
00349     QStringList donnees;
00350
00351     unsigned int nombreCasiers = Article::recupererNombreCasiersPourNomArticle
      (nomArticle);
00352     #ifdef DEBUG_SUPERVISION
00353         qDebug() << Q_FUNC_INFO << "nombreCasiers" << nombreCasiers;
00354     #endif
00355
00356     if(nombreCasiers > 1)
00357     {
00358         QVector<QString> numeroDesCasiers;
00359
00360         numeroDesCasiers = Article::recupererNumeroCasierPourNomArticle
      (nomArticle);
00361
00362         for(int i = 0; i < numeroDesCasiers.size(); i++)
00363         {
00364             article->recupererDonneesArticleParNumeroCasier(
      numeroDesCasiers[i]);
00365             ajouterDonneesArticle(article, donneesArticle, donnees);
00366         }
00367
00368         if(!donneesArticle.isEmpty())
00369         {
00370             emit donneesArticleSelectionne(donneesArticle);
00371         }
00372     }
00373     else
00374     {
00375         article->recupererDonneesArticleParNom(nomArticle);
00376         ajouterDonneesArticle(article, donneesArticle, donnees);
00377
00378         if(!donneesArticle.isEmpty())
00379         {
00380             emit donneesArticleSelectionne(donneesArticle.at(0));
00381         }
00382     }
00383 }
```

8.15.3.32 traiterTramePoids

```
void Supervision::traiterTramePoids (
            QString trame )  [slot]
```

Définition de la méthode traiterTramePoids.

traite la trame poids reçue

Paramètres

trame	

Définition à la ligne 391 du fichier Supervision.cpp.

Références comptageAutomatiqueAjouterOuPrendre(), comptageAutomatiqueQuantite(), compter(), erreurDepassementQuantite(), extraireNumeroCasier(), extrairePoids(), Article : :get(), Utilisateur : :getIdUtilisateur(), mettreAJourMouvement(), Article : :mettre↩ AJourQuantite(), Article : :recupererDonneesArticleParNumeroCasier(), TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_ID_S↩ TOCK, TABLE_ARTICLE_NOM_ARTICLE, TABLE_ARTICLE_POIDS, TABLE_ARTICLE_QUANTITE, TABLE_ARTICLE_TARE, et utilisateur.

Référencé par connecterSignauxSlots(), et connecterUtilisateur().

```
00392 {
```

```
00393     #ifdef DEBUG_SUPERVISION
00394         qDebug() << Q_FUNC_INFO << trame;
00395     #endif
00396
00397     QString numCasier = extraireNumeroCasier(trame);
00398
00399     Article *article = new Article(this);
00400     if(article->recupererDonneesArticleParNumeroCasier(numCasier))
00401     {
00402         #ifdef DEBUG_SUPERVISION
00403             qDebug() << Q_FUNC_INFO << "Article" << article->get(
      TABLE_ARTICLE_NOM_ARTICLE) << article->get(
      TABLE_ARTICLE_QUANTITE) << article->get(
      TABLE_ARTICLE_DISPONIBLE);
00404         #endif
00405
00406         int nombreArticle = compter(article->get(TABLE_ARTICLE_POIDS),
      extrairePoids(trame), article->get(TABLE_ARTICLE_TARE));
00407
00408         QString strArticleQuantite = article->get(TABLE_ARTICLE_QUANTITE);
00409
00410         int articleQuantite = strArticleQuantite.toInt();
00411
00412         if(nombreArticle > articleQuantite)
00413         {
00414             emit erreurDepassementQuantite();
00415         }
00416         else
00417         {
00418             article->mettreAJourQuantite(QString::number(nombreArticle));
00419
00420             QString idUtilisateur = utilisateur->getIdUtilisateur();
00421             QString idStock = article->get(TABLE_ARTICLE_ID_STOCK);
00422             QString idAction = QString::number(
      comptageAutomatiqueAjouterOuPrendre(article->
      get(TABLE_ARTICLE_DISPONIBLE), nombreArticle));
00423             QString quantite = QString::number(comptageAutomatiqueQuantite(
      article->get(TABLE_ARTICLE_DISPONIBLE), nombreArticle));
00424
00425             mettreAJourMouvement(idUtilisateur, idStock,  idAction, quantite);
00426         }
00427     }
00428     else
00429     {
00430         #ifdef DEBUG_SUPERVISION
00431             qDebug() << Q_FUNC_INFO << "Article introuvable !";
00432         #endif
00433     }
00434 }
```

8.15.3.33 verifierArticlePresentDansCasier()

```
bool Supervision::verifierArticlePresentDansCasier (
            QString numCasier,
            QString idArticle )  [private]
```

Définition de la méthode verifierArticlePresentDansCasier.

permet de vérifier si l'article est bien présent dans le casier

Paramètres

QString	numCasier
QString	idArticle

Renvoie

bool articlePresent

Définition à la ligne 596 du fichier Supervision.cpp.

Références bdd, et Bdd : :recuperer().

Référencé par ajouterObjetAvecCodeBarre(), et prendreObjetAvecCodeBarre().

```
00597 {
00598     QString requete = "SELECT idArticle FROM Stock WHERE numeroCasier = "+ numCasier + ";";
00599     QString donnees;
00600
00601     bdd->recuperer(requete, donnees);
00602
00603     if(donnees == idArticle)
00604     {
00605         return true;
00606     }
00607     else
00608     {
00609         return false;
00610     }
00611 }
```

8.15.3.34 verifierAuthentificationBadge

```
void Supervision::verifierAuthentificationBadge (
             QString badge )  [slot]
```

Définition de la méthode verifierAuthentificationBadge.

Permet la vérification des données utilisateur par badge

Paramètres

badge	Chaîne de caractères de l'uid du badge

Définition à la ligne 128 du fichier Supervision.cpp.

Références connecterUtilisateur(), recupererDonneesUtilisateur(), et verifierDonneesUtilisateur().

Référencé par connecterSignauxSlots().

```
00129 {
00130     QString requeteBDD = "SELECT * from Utilisateur where Badge = '" + badge + "';";
00131     QStringList donnees = recupererDonneesUtilisateur(requeteBDD);
00132     if(verifierDonneesUtilisateur(donnees))
00133     {
00134         connecterUtilisateur(donnees);
00135     }
00136 }
```

8.15.3.35 verifierAuthentificationIdentifiant

```
void Supervision::verifierAuthentificationIdentifiant (
             QString identifiant,
             QString motDePasse )  [slot]
```

Définition de la méthode verifierAuthentificationIdentifiant.

Permet la vérification des données utilisateur par champs

Paramètres

identifiant	Chaîne de caractères de l'identifiant
motDePasse	Chaîne de caractères du mot de passe

Définition à la ligne 145 du fichier Supervision.cpp.

Références bdd, connecterUtilisateur(), crypterMotDepasse(), Bdd : :executer(), recupererDonneesUtilisateur(), et verifierDonnees↩ Utilisateur().

Référencé par connecterSignauxSlots().

```
00146 {
00147     this->crypterMotDepasse(motDePasse);
00148
00149     #ifdef CHANGE_PASSWORD_BEFORE
00150     QString requete = QString("UPDATE Utilisateur SET MotDePasse='%1' WHERE Identifiant='%2'").arg(
      motDePasse).arg(identifiant);
00151     bdd->executer(requete);
00152     #endif
00153
00154     QString requeteBDD = "SELECT * from Utilisateur where Identifiant =  '" + identifiant + "' &&
       MotDePasse = '" + motDePasse + "';";
00155     QStringList donnees = recupererDonneesUtilisateur(requeteBDD);
00156     if(verifierDonneesUtilisateur(donnees))
00157     {
00158         connecterUtilisateur(donnees);
00159     }
00160 }
```

8.15.3.36 verifierDateValidite()

```
bool Supervision::verifierDateValidite (
            QString stringDateValidite )  [private]
```

Définition de la méthode verifierDateValidite.

Permet de vérifier la date de validité

Paramètres

stringDateValidite	Chaîne de caractères de la date de validité

Renvoie

Si la date de validité est valide

Définition à la ligne 202 du fichier Supervision.cpp.

Référencé par verifierDonneesUtilisateur().

```
00203 {
00204     // Verification de la date de validité
00205     QDate dateValidite = dateValidite.fromString(stringDateValidite,"yyyy-MM-dd");
00206     QDate dateActuelle = QDate::currentDate();
00207
00208     #ifdef DEBUG_SUPERVISION
00209         qDebug() << "Date actuelle" << dateActuelle;
00210         qDebug() << "Date validité" << dateValidite;
00211     #endif
00212
00213     if(dateActuelle <= dateValidite)
00214     {
00215         return true;
00216     }
00217
00218     return false;
00219 }
```

8.15.3.37 verifierDonneesUtilisateur()

```
bool Supervision::verifierDonneesUtilisateur (
            QStringList & donnees )  [private]
```

Définition de la méthode verifierDonneesUtilisateur.

Vérfie la date de validité et que les données ne sont pas vide sinon renvoie des erreurs

Paramètres

donnees	Chaîne de caractères de la date de validité

Renvoie

Si la demande de connexion est autoriser

Définition à la ligne 228 du fichier Supervision.cpp.

Références MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE, MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE, reponseDemandeDeConnexion(), TABLE_UTILISATEUR_DATE_VALIDITE, et verifierDateValidite().

Référencé par verifierAuthentificationBadge(), et verifierAuthentificationIdentifiant().

```
00229 {
00230     #ifdef DEBUG_SUPERVISON
00231         qDebug() << Q_FUNC_INFO << donnees;
00232     #endif
00233
00234     if(!donnees.isEmpty())
00235     {
00236         if(verifierDateValidite(donnees.at(
      TABLE_UTILISATEUR_DATE_VALIDITE)))
00237         {
00238             emit reponseDemandeDeConnexion(true, "");
00239             return true;
00240         }
00241         else
00242         {
00243             emit reponseDemandeDeConnexion(false,
      MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE);
00244             return false;
00245         }
00246     }
00247     else
00248     {
00249         emit reponseDemandeDeConnexion(false,
      MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE);
00250         return false;
00251     }
00252 }
```

8.15.4 Documentation des données membres

8.15.4.1 armoire

```
Armoire* Supervision::armoire  [private]
```

association d'un objet Armoire

Définition à la ligne 91 du fichier Supervision.h.

Référencé par connecterSignauxSlots(), creerCasiers(), getInformationsArmoire(), et Supervision().

8.15.4.2 bdd

```
Bdd* Supervision::bdd  [private]
```

association d'un objet Bdd (accès à la base de données)

Définition à la ligne 87 du fichier Supervision.h.

Référencé par ajouterObjetAvecCodeBarre(), mettreAJourMouvement(), prendreObjetAvecCodeBarre(), rechercherArticle(), recupererDonneesUtilisateur(), recupererIdStockAvecNumeroCasier(), Supervision(), verifierArticlePresentDansCasier(), et verifier↩AuthentificationIdentifiant().

8.15.4.3 casiers

```
QVector<Casier*> Supervision::casiers  [private]
```

les casiers de l'armoire

Définition à la ligne 93 du fichier Supervision.h.

Référencé par ajouterObjetAvecCodeBarre(), creerCasiers(), getCasiers(), prendreObjetAvecCodeBarre(), et rechercherCasier↩Ouvert().

8.15.4.4 codeBarre

```
CodeBarre* Supervision::codeBarre  [private]
```

association d'un objet CodeBarre

Définition à la ligne 90 du fichier Supervision.h.

Référencé par connecterSignauxSlots(), et Supervision().

8.15.4.5 communication

```
Communication* Supervision::communication  [private]
```

association d'un objet Communication

Définition à la ligne 92 du fichier Supervision.h.

Référencé par connecterSignauxSlots(), et Supervision().

8.15.4.6 ihm

```
Ihm* Supervision::ihm  [private]
```

association d'un objet Ihm (fenêtre principale de l'application)

Définition à la ligne 86 du fichier Supervision.h.

Référencé par connecterSignauxSlots(), et creerCasiers().

8.15.4.7 rfid

```
Rfid* Supervision::rfid  [private]
```

association d'un objet Rfid (le lecteur de badge)

Définition à la ligne 88 du fichier Supervision.h.

Référencé par connecterSignauxSlots(), et Supervision().

8.15.4.8 utilisateur

```
Utilisateur* Supervision::utilisateur  [private]
```

association d'un objet Utilisateur (l'utilisateur authentifié)

Définition à la ligne 89 du fichier Supervision.h.

Référencé par ajouterObjetAvecCodeBarre(), connecterUtilisateur(), deconnecterUtilisateur(), prendreObjetAvecCodeBarre(), Supervision(), et traiterTramePoids().

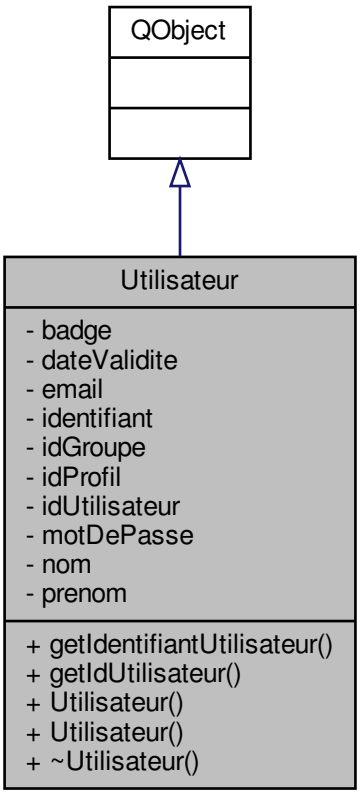
La documentation de cette classe a été générée à partir des fichiers suivants :
- Supervision.h
- Supervision.cpp

8.16 Référence de la classe Utilisateur

La classe Utilisateur gère les données relative à l'utilisateur.

```
#include <Utilisateur.h>
```

Graphe de collaboration de Utilisateur :



Fonctions membres publiques

- QString getIdentifiantUtilisateur ()
  Définition de la méthode getIdentifiantUtilisateur.
- QString getIdUtilisateur ()
  Définition de la méthode getIdUtilisateur.
- Utilisateur (QObject *parent=nullptr)
  Constructeur de la classe Utilisateur.
- Utilisateur (QStringList donnees, QObject *parent=nullptr)
  Constructeur de la classe Utilisateur.
- ~Utilisateur ()
  Destructeur de la classe Utilisateur.

Attributs privés

- QString badge
  badge de l'utilisateur connecté
- QString dateValidite
  dateValidite du compte
- QString email
  email de l'utilisateur connecté
- QString identifiant
  identifiant de l'utilisateur connecté
- QString idGroupe
  idGroupe de l'utilisateur connecté
- QString idProfil
  idProfil de l'utilisateur connecté
- QString idUtilisateur
  idUtilisateur de l'utilisateur connecté
- QString motDePasse
  motDePasse de l'utilisateur connecté
- QString nom
  nom de l'utilisateur connecté
- QString prenom
  prenom de l'utilisateur connecté

8.16.1 Description détaillée

La classe Utilisateur gère les données relative à l'utilisateur.

Auteur

Legger Pierre-Antoine

Version

1.0

Date

mercredi 04 Mars 2020

Définition à la ligne 52 du fichier Utilisateur.h.

8.16.2 Documentation des constructeurs et destructeur

8.16.2.1 Utilisateur() [1/2]

```
Utilisateur::Utilisateur (
            QObject * parent = nullptr )
```

Constructeur de la classe Utilisateur.

Initialise un objet Utilisateur

Paramètres

parent	

Définition à la ligne 22 du fichier Utilisateur.cpp.

Références badge, dateValidite, email, identifiant, idGroupe, idProfil, idUtilisateur, motDePasse, nom, et prenom.

```
00022                                           : QObject(parent)
00023 {
00024     #ifdef DEBUG_UTILISATEUR
00025         qDebug() << Q_FUNC_INFO;
00026     #endif
00027     idUtilisateur = "";
00028     idProfil = "";
00029     idGroupe = "";
00030     nom = "";
00031     prenom = "";
00032     dateValidite = "";
00033     identifiant = "";
00034     motDePasse = "";
00035     badge = "";
00036     email = "";
00037 }
```

8.16.2.2 Utilisateur() [2/2]

```
Utilisateur::Utilisateur (
            QStringList donnees,
            QObject * parent = nullptr )
```

Constructeur de la classe Utilisateur.

Initialise un objet Utilisateur

Paramètres

donnees	
parent	

Définition à la ligne 45 du fichier Utilisateur.cpp.

Références badge, dateValidite, email, identifiant, idGroupe, idProfil, idUtilisateur, motDePasse, nom, prenom, TABLE_UTILISAT↩EUR_BADGE, TABLE_UTILISATEUR_DATE_VALIDITE, TABLE_UTILISATEUR_EMAIL, TABLE_UTILISATEUR_ID_GROUPE, T↩ABLE_UTILISATEUR_ID_PROFIL, TABLE_UTILISATEUR_ID_UTILISATEUR, TABLE_UTILISATEUR_IDENTIFIANT, TABLE_UT↩ILISATEUR_MOT_DE_PASSE, TABLE_UTILISATEUR_NOM, et TABLE_UTILISATEUR_PRENOM.

```
00045                                                       : QObject(parent)
00046 {
00047     #ifdef DEBUG_UTILISATEUR
00048         qDebug() << Q_FUNC_INFO << donnees;
00049     #endif
00050     idUtilisateur = donnees.at(TABLE_UTILISATEUR_ID_UTILISATEUR
      );
00051     idProfil = donnees.at(TABLE_UTILISATEUR_ID_PROFIL);
00052     idGroupe = donnees.at(TABLE_UTILISATEUR_ID_GROUPE);
00053     nom = donnees.at(TABLE_UTILISATEUR_NOM);
00054     prenom = donnees.at(TABLE_UTILISATEUR_PRENOM);
00055     dateValidite = donnees.at(TABLE_UTILISATEUR_DATE_VALIDITE);
00056     identifiant = donnees.at(TABLE_UTILISATEUR_IDENTIFIANT);
00057     motDePasse = donnees.at(TABLE_UTILISATEUR_MOT_DE_PASSE);
00058     badge = donnees.at(TABLE_UTILISATEUR_BADGE);
00059     email = donnees.at(TABLE_UTILISATEUR_EMAIL);
00060 }
```

8.16.2.3 ~Utilisateur()

```
Utilisateur::~Utilisateur ( )
```

Destructeur de la classe Utilisateur.

Détruit un objet Utilisateur

Définition à la ligne 67 du fichier Utilisateur.cpp.

```
00068 {
00069
00070 }
```

8.16.3 Documentation des fonctions membres

8.16.3.1 getIdentifiantUtilisateur()

```
QString Utilisateur::getIdentifiantUtilisateur ( )
```

Définition de la méthode getIdentifiantUtilisateur.

retourne les identifiant de l'utilisateur

Renvoie

les identifiants

Définition à la ligne 78 du fichier Utilisateur.cpp.

Références nom, et prenom.

Référencé par Supervision : :connecterUtilisateur().

```
00079 {
00080     return nom + " " + prenom;
00081 }
```

8.16.3.2 getIdUtilisateur()

```
QString Utilisateur::getIdUtilisateur ( )
```

Définition de la méthode getIdUtilisateur.

retourne l'es identifiant de l'utilisateur'id de l'utilisateur

Renvoie

QString idUtilisateur

Définition à la ligne 88 du fichier Utilisateur.cpp.

Références idUtilisateur.

Référencé par Supervision : :ajouterObjetAvecCodeBarre(), Supervision : :prendreObjetAvecCodeBarre(), et Supervision : :traiter↩ TramePoids().

```
00089 {
00090     return idUtilisateur;
00091 }
```

8.16.4 Documentation des données membres

8.16.4.1 badge

```
QString Utilisateur::badge  [private]
```

badge de l'utilisateur connecté

Définition à la ligne 76 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.2 dateValidite

```
QString Utilisateur::dateValidite  [private]
```

dateValidite du compte

Définition à la ligne 73 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.3 email

```
QString Utilisateur::email  [private]
```

email de l'utilisateur connecté

Définition à la ligne 77 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.4 identifiant

```
QString Utilisateur::identifiant  [private]
```

identifiant de l'utilisateur connecté

Définition à la ligne 74 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.5 idGroupe

```
QString Utilisateur::idGroupe  [private]
```

idGroupe de l'utilisateur connecté

Définition à la ligne 70 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.6 idProfil

```
QString Utilisateur::idProfil  [private]
```

idProfil de l'utilisateur connecté

Définition à la ligne 69 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.7 idUtilisateur

```
QString Utilisateur::idUtilisateur  [private]
```

idUtilisateur de l'utilisateur connecté

Définition à la ligne 68 du fichier Utilisateur.h.

Référencé par getIdUtilisateur(), et Utilisateur().

8.16.4.8 motDePasse

```
QString Utilisateur::motDePasse  [private]
```

motDePasse de l'utilisateur connecté

Définition à la ligne 75 du fichier Utilisateur.h.

Référencé par Utilisateur().

8.16.4.9 nom

```
QString Utilisateur::nom  [private]
```

nom de l'utilisateur connecté

Définition à la ligne 71 du fichier Utilisateur.h.

Référencé par getIdentifiantUtilisateur(), et Utilisateur().

8.16.4.10 prenom

```
QString Utilisateur::prenom  [private]
```

prenom de l'utilisateur connecté

Définition à la ligne 72 du fichier Utilisateur.h.

Référencé par getIdentifiantUtilisateur(), et Utilisateur().

La documentation de cette classe a été générée à partir des fichiers suivants :

— Utilisateur.h
— Utilisateur.cpp

9 Documentation des fichiers

9.1 Référence du fichier Armoire.cpp

Définition de la classe Armoire.

```
#include "Armoire.h"
#include "Bdd.h"
#include <QNetworkInterface>
```

9.1.1 Description détaillée

Définition de la classe Armoire.

Auteur

Tranchat Joffrey

Version

1.0

Date

22 Mars 2020

Définition dans le fichier Armoire.cpp.

9.2 Armoire.cpp

```
00001 #include "Armoire.h"
00002 #include "Bdd.h"
00003 #include <QNetworkInterface>
00004
00022 Armoire::Armoire(QObject *parent) : QObject(parent)
00023 {
00024     #ifdef DEBUG_ARMOIRE
00025         qDebug() << Q_FUNC_INFO;
00026     #endif
00027     bdd = Bdd::getInstance();
00028     adresseIPArmoire = lireAdresseIP();
00029     recupererArmoire();
00030 }
00031
00036 Armoire::~Armoire()
00037 {
00038     Bdd::detruireInstance();
00039     #ifdef DEBUG_ARMOIRE
00040         qDebug() << Q_FUNC_INFO;
00041     #endif
00042 }
00043
00049 void Armoire::recupererArmoire(QString idArmoire)
00050 {
00051     QString requeteBDD;
00052
00053     if(!idArmoire.isEmpty()) // par id
00054     {
00055         requeteBDD = "SELECT idArmoire, Nom, Description, nbCasiers from Armoire where idArmoire = '" +
      idArmoire + "'";
00056         QStringList donnees;
00057         bdd->recuperer(requeteBDD, donnees);
00058
00059         #ifdef DEBUG_ARMOIRE
00060             qDebug() << Q_FUNC_INFO << donnees;
00061         #endif
00062
00063         if(donnees.size() > 0)
00064         {
00065             this->idArmoire = donnees.at(TABLE_ARMOIRE_ID_ARMOIRE);
00066             nom = donnees.at(TABLE_ARMOIRE_NOM);
00067             description = donnees.at(TABLE_ARMOIRE_DESCRIPTION);
00068             nbCasiers = donnees.at(TABLE_ARMOIRE_NB_CASIERS);
00069         }
00070     }
00071 }
00072
00078 QStringList Armoire::getInformations()
00079 {
00080     QStringList informations;
00081
00082     informations << idArmoire << nom << description <<
      nbCasiers << adresseIPArmoire;
00083
00084     emit informationsArmoire(informations);
00085
00086     return informations;
00087 }
00088
00094 QString Armoire::getIdArmoire() const
00095 {
00096     return idArmoire;
00097 }
00098
00105 QString Armoire::getNom() const
00106 {
00107     return nom;
00108 }
00109
00116 QString Armoire::getDescripton() const
00117 {
00118     return description;
00119 }
00120
00127 QString Armoire::getNbCasiers() const
00128 {
00129     return nbCasiers;
00130 }
00131
00138 QString Armoire::getAdresseIPArmoire() const
00139 {
00140     return adresseIPArmoire;
00141 }
00142
00149 QString Armoire::lireAdresseIP()
00150 {
00151     QStringList adresses;
00152     foreach(QHostAddress adresse, QNetworkInterface::allAddresses())
```

```
00153     {
00154         // Filtre les adresses localhost ...
00155         if(adresse != QHostAddress::LocalHostIPv6
00156            && adresse != QHostAddress::LocalHost
00157            // ... APIPA ...
00158            && !adresse.isInSubnet(QHostAddress::parseSubnet("169.254.0.0/16"))
00159            // ... Lien Local IPv6
00160            && !adresse.isInSubnet(QHostAddress::parseSubnet("FE80::/64")))
00161         {
00162             qDebug() << Q_FUNC_INFO << adresse.toString();
00163             adresses << adresse.toString();
00164         }
00165     }
00166
00167     foreach(QString adresse, adresses)
00168     {
00169         #ifdef DEBUG_ARMOIRE
00170             qDebug() << Q_FUNC_INFO << adresse;
00171         #endif
00172         if(adresse.contains("192."))
00173             return adresse;
00174     }
00175
00176     /*if(adresses.count() > 0)
00177     {
00178         return adresses.at(0);
00179     }*/
00180
00181     return QString("");
00182 }
```

9.3 Référence du fichier Armoire.h

Déclaration de la classe Armoire.

```
#include <QObject>
#include <QString>
#include <QDebug>
```

Classes

— class Armoire
La classe Armoire traite les articles.

Macros

— #define DEBUG_ARMOIRE

Énumérations

— enum ChampArmoire { TABLE_ARMOIRE_ID_ARMOIRE, TABLE_ARMOIRE_NOM, TABLE_ARMOIRE_DESCRIPTION, TABLE_ARMOIRE_NB_CASIERS }
Définit les différents champs de la table Armoire.

9.3.1 Description détaillée

Déclaration de la classe Armoire.

Auteur

Tranchat Joffrey

Version

1.0

Date

Dimanche 22 Mars 2020

Définition dans le fichier Armoire.h.

9.3.2 Documentation des macros

9.3.2.1 DEBUG_ARMOIRE

```
#define DEBUG_ARMOIRE
```

Définition à la ligne 21 du fichier Armoire.h.

9.3.3 Documentation du type de l'énumération

9.3.3.1 ChampArmoire

```
enum ChampArmoire
```

Définit les différents champs de la table Armoire.

Valeurs énumérées

TABLE_ARMOIRE_ID_ARMOIRE	
TABLE_ARMOIRE_NOM	
TABLE_ARMOIRE_DESCRIPTION	
TABLE_ARMOIRE_NB_CASIERS	

Définition à la ligne 27 du fichier Armoire.h.

```
00028 {
00029     TABLE_ARMOIRE_ID_ARMOIRE,
00030     TABLE_ARMOIRE_NOM,
00031     TABLE_ARMOIRE_DESCRIPTION,
00032     TABLE_ARMOIRE_NB_CASIERS
00033 };
```

9.4 Armoire.h

```
00001 #ifndef ARMOIRE_H
00002 #define ARMOIRE_H
00003
00017 #include <QObject>
00018 #include <QString>
00019 #include <QDebug>
00020
00021 #define DEBUG_ARMOIRE
00022
00027 enum ChampArmoire
00028 {
00029     TABLE_ARMOIRE_ID_ARMOIRE,
00030     TABLE_ARMOIRE_NOM,
00031     TABLE_ARMOIRE_DESCRIPTION,
00032     TABLE_ARMOIRE_NB_CASIERS
00033 };
00034
00035 class Bdd;
00036
00049 class Armoire : public QObject
00050 {
00051     Q_OBJECT
00052 public:
00053     Armoire(QObject *parent = nullptr);
```

```
00054     ~Armoire();
00055
00056     void recupererArmoire(QString idArmoire="1");
00057     QStringList getInformations();
00058     QString getIdArmoire() const;
00059     QString getNom() const;
00060     QString getDescripton() const;
00061     QString getNbCasiers() const;
00062     QString getAdresseIPArmoire() const;
00063
00064 private:
00065     Bdd *bdd;
00066     QString idArmoire;
00067     QString nom;
00068     QString description;
00069     QString nbCasiers;
00070     QString adresseIPArmoire;
00071
00072     QString lireAdresseIP();
00073
00074 signals:
00075     void informationsArmoire(QStringList);
00076 };
00077
00078 #endif // ARMOIRE_H
```

9.5 Référence du fichier Article.cpp

Définition de la classe Article.

```
#include "Article.h"
#include "Bdd.h"
#include <QtMath>
```

9.5.1 Description détaillée

Définition de la classe Article.

Auteur

Tranchat Joffrey

Version

1.0

Date

11 Mars 2020

Définition dans le fichier Article.cpp.

9.6 Article.cpp

```
00001 #include "Article.h"
00002 #include "Bdd.h"
00003 #include <QtMath>
00004
00017 Bdd* Article::bdd = Bdd::getInstance();
00018
00024 Article::Article(QObject *parent) : QObject(parent)
00025 {
00026     #ifdef DEBUG_ARTICLE
00027         qDebug() << Q_FUNC_INFO;
00028     #endif
00029     //bdd = Bdd::getInstance();
00030 }
00031
00036 Article::~Article()
00037 {
00038     //Bdd::detruireInstance();
00039     #ifdef DEBUG_ARTICLE
00040         qDebug() << Q_FUNC_INFO;
00041     #endif
00042 }
00043
00050 bool Article::recupererDonneesArticle(QString
      idArticle, int numCasier)
00051 {
00052     if(idArticle.isEmpty())
00053         return false;
00054
00055     QString requeteBDD;
00056
00057     if(numCasier == 0)
00058     {
00059         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
       Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
       Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
       JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
       Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
       Article.idArticle = '" + idArticle + "'";
00060     }
00061     else
00062     {
00063         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
       Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
       Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
       JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
       Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
       Article.idArticle = '" + idArticle + "' AND Stock.numeroCasier = '" + numCasier + "'";
00064     }
00065
00066     QStringList donnees; // un seul casier pour cet article
00067     bdd->recuperer(requeteBDD, donnees);
00068
00069     #ifdef DEBUG_ARTICLE
00070         qDebug() << Q_FUNC_INFO << donnees;
00071     #endif
00072
00073     if(donnees.size() > 0)
00074     {
00075         this->idStock = donnees.at(TABLE_ARTICLE_ID_STOCK);
00076         this->idArticle = donnees.at(TABLE_ARTICLE_ID_ARTICLE);
00077         this->nomArticle = donnees.at(TABLE_ARTICLE_NOM_ARTICLE);
00078         this->idType = donnees.at(TABLE_ARTICLE_ID_TYPE);
00079         this->nomType = donnees.at(TABLE_ARTICLE_NOM_TYPE);
00080         this->idComptage = donnees.at(TABLE_ARTICLE_ID_COMPTAGE);
00081         this->nomComptage = donnees.at(TABLE_ARTICLE_NOM_COMPTAGE);
00082         this->codeBarre = donnees.at(TABLE_ARTICLE_CODE_BARRE);
00083         this->designation = donnees.at(TABLE_ARTICLE_DESIGNATION);
00084         this->quantite = donnees.at(TABLE_ARTICLE_QUANTITE);
00085         this->disponible = donnees.at(TABLE_ARTICLE_DISPONIBLE);
00086         this->poidsArticle = donnees.at(TABLE_ARTICLE_POIDS);
00087         this->tare = donnees.at(TABLE_ARTICLE_TARE);
00088         this->idUnite = donnees.at(TABLE_ARTICLE_ID_UNITE);
00089         this->nomUnite = donnees.at(TABLE_ARTICLE_NOM_UNITE);
00090         this->numeroCasier = donnees.at(TABLE_ARTICLE_NUMERO_CASIER)
      ;
00091         return true;
00092     }
00093     return false;
00094 }
00095
00102 bool Article::recupererDonneesArticleParNom(QString
      nomArticle, int numCasier)
00103 {
00104     if(nomArticle.isEmpty())
00105         return false;
00106
00107     QString requeteBDD;
00108
```

```
00109     if(numCasier == 0)
00110     {
00111         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
      Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
      JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Article.Nom = '" + nomArticle + "'";
00112     }
00113     else
00114     {
00115         requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType,
      Type.nom AS Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER
      JOIN Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Article.Nom = '" + nomArticle + "' AND Stock.numeroCasier = '" + numCasier + "'";
00116     }
00117
00118     QStringList donnees; // un seul casier pour cet article
00119     bdd->recuperer(requeteBDD, donnees);
00120
00121     #ifdef DEBUG_ARTICLE
00122         qDebug() << Q_FUNC_INFO << donnees;
00123     #endif
00124
00125     if(donnees.size() > 0)
00126     {
00127         this->idStock = donnees.at(TABLE_ARTICLE_ID_STOCK);
00128         this->idArticle = donnees.at(TABLE_ARTICLE_ID_ARTICLE);
00129         this->nomArticle = donnees.at(TABLE_ARTICLE_NOM_ARTICLE);
00130         this->idType = donnees.at(TABLE_ARTICLE_ID_TYPE);
00131         this->nomType = donnees.at(TABLE_ARTICLE_NOM_TYPE);
00132         this->idComptage = donnees.at(TABLE_ARTICLE_ID_COMPTAGE);
00133         this->nomComptage = donnees.at(TABLE_ARTICLE_NOM_COMPTAGE);
00134         this->codeBarre = donnees.at(TABLE_ARTICLE_CODE_BARRE);
00135         this->designation = donnees.at(TABLE_ARTICLE_DESIGNATION);
00136         this->quantite = donnees.at(TABLE_ARTICLE_QUANTITE);
00137         this->disponible = donnees.at(TABLE_ARTICLE_DISPONIBLE);
00138         this->poidsArticle = donnees.at(TABLE_ARTICLE_POIDS);
00139         this->tare = donnees.at(TABLE_ARTICLE_TARE);
00140         this->idUnite = donnees.at(TABLE_ARTICLE_ID_UNITE);
00141         this->nomUnite = donnees.at(TABLE_ARTICLE_NOM_UNITE);
00142         this->numeroCasier = donnees.at(TABLE_ARTICLE_NUMERO_CASIER)
    ;
00143         return true;
00144     }
00145
00146     return false;
00147 }
00148
00155 bool Article::recupererDonneesArticleParNumeroCasier(QString
      numeroCasier)
00156 {
00157     if(numeroCasier.isEmpty())
00158         return false;
00159
00160     QString requeteBDD;
00161
00162     requeteBDD = "SELECT Stock.idStock, Article.idArticle, Article.Nom AS Article, Type.idType, Type.nom AS
      Type, Comptage.idComptage, Comptage.Nom AS Comptage, Article.Code, Article.Designation, Stock.Quantite,
      Stock.Disponible, Article.Poids, Stock.Tare, Unite.idUnite, Unite.Nom, Stock.numeroCasier FROM Stock INNER JOIN
      Article ON Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN
      Comptage ON Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE
      Stock.numeroCasier = '" + numeroCasier + "'";
00163
00164     QStringList donnees;
00165     bdd->recuperer(requeteBDD, donnees);
00166
00167     #ifdef DEBUG_ARTICLE
00168         qDebug() << Q_FUNC_INFO << donnees;
00169     #endif
00170
00171     if(donnees.size() > 0)
00172     {
00173         this->idStock = donnees.at(TABLE_ARTICLE_ID_STOCK);
00174         this->idArticle = donnees.at(TABLE_ARTICLE_ID_ARTICLE);
00175         this->nomArticle = donnees.at(TABLE_ARTICLE_NOM_ARTICLE);
00176         this->idType = donnees.at(TABLE_ARTICLE_ID_TYPE);
00177         this->nomType = donnees.at(TABLE_ARTICLE_NOM_TYPE);
00178         this->idComptage = donnees.at(TABLE_ARTICLE_ID_COMPTAGE);
00179         this->nomComptage = donnees.at(TABLE_ARTICLE_NOM_COMPTAGE);
00180         this->codeBarre = donnees.at(TABLE_ARTICLE_CODE_BARRE);
00181         this->designation = donnees.at(TABLE_ARTICLE_DESIGNATION);
00182         this->quantite = donnees.at(TABLE_ARTICLE_QUANTITE);
00183         this->disponible = donnees.at(TABLE_ARTICLE_DISPONIBLE);
00184         this->poidsArticle = donnees.at(TABLE_ARTICLE_POIDS);
00185         this->tare = donnees.at(TABLE_ARTICLE_TARE);
00186         this->idUnite = donnees.at(TABLE_ARTICLE_ID_UNITE);
00187         this->nomUnite = donnees.at(TABLE_ARTICLE_NOM_UNITE);
00188         this->numeroCasier = donnees.at(TABLE_ARTICLE_NUMERO_CASIER);
```

```
00189         return true;
00190     }
00191
00192     return false;
00193 }
00194
00201 unsigned int Article::recupererNombreCasiersPourIdArticle(
      QString idArticle)
00202 {
00203     QString requete = "SELECT COUNT(Stock.idArticle) FROM Stock INNER JOIN Article ON Stock.idArticle =
       Article.idArticle WHERE Article.idArticle = '" + idArticle + "'";
00204
00205     QString donnees;
00206     bdd->recuperer(requete, donnees);
00207
00208     return donnees.toUInt();
00209 }
00210
00217 unsigned int Article::recupererNombreCasiersPourNomArticle(
      QString nomArticle)
00218 {
00219     QString requete = "SELECT COUNT(Stock.idArticle) FROM Stock INNER JOIN Article ON Stock.idArticle =
       Article.idArticle WHERE Article.Nom = '" + nomArticle + "'";
00220
00221     QString donnees;
00222     bdd->recuperer(requete, donnees);
00223
00224     return donnees.toUInt();
00225 }
00226
00233 QVector<QString> Article::recupererNumeroCasierPourIdArticle(
      QString idArticle)
00234 {
00235     QString requete = "SELECT Stock.numeroCasier FROM Stock INNER JOIN Article ON
       Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN Comptage ON
       Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE Article.idArticle = '" + idArticle +
00236
00237     QVector<QString> donnees;
00238     bdd->recuperer(requete, donnees);
00239
00240     return donnees;
00241 }
00242
00249 QVector<QString> Article::recupererNumeroCasierPourNomArticle(
      QString nomArticle)
00250 {
00251     QString requete = "SELECT Stock.numeroCasier FROM Stock INNER JOIN Article ON
       Article.idArticle=Stock.idArticle INNER JOIN Type ON Type.idType=Article.idType INNER JOIN Comptage ON
       Comptage.idComptage=Stock.idComptage INNER JOIN Unite ON Unite.idUnite=Stock.idUnite WHERE Article.Nom = '" + nomArticle + "'";
00252
00253     QVector<QString> donnees;
00254     bdd->recuperer(requete, donnees);
00255
00256     return donnees;
00257 }
00258
00265 QString Article::get(ChampArticle champ)
00266 {
00267     switch(champ)
00268     {
00269         case TABLE_ARTICLE_ID_STOCK:
00270             return this->idStock;
00271             break;
00272         case TABLE_ARTICLE_ID_ARTICLE:
00273             return this->idArticle;
00274             break;
00275         case TABLE_ARTICLE_NOM_ARTICLE:
00276             return this->nomArticle;
00277             break;
00278         case TABLE_ARTICLE_ID_TYPE:
00279             return this->idType;
00280             break;
00281         case TABLE_ARTICLE_NOM_TYPE:
00282             return this->nomType;
00283             break;
00284         case TABLE_ARTICLE_ID_COMPTAGE:
00285             return this->idComptage;
00286             break;
00287         case TABLE_ARTICLE_NOM_COMPTAGE:
00288             return this->nomComptage;
00289             break;
00290         case TABLE_ARTICLE_CODE_BARRE:
00291             return this->codeBarre;
00292             break;
00293         case TABLE_ARTICLE_DESIGNATION:
00294             return this->designation;
00295             break;
00296         case TABLE_ARTICLE_QUANTITE:
00297             return this->quantite;
00298             break;
00299         case TABLE_ARTICLE_DISPONIBLE:
```

```
00300             return this->disponible;
00301             break;
00302         case TABLE_ARTICLE_POIDS:
00303             return this->poidsArticle;
00304             break;
00305         case TABLE_ARTICLE_TARE:
00306             return this->tare;
00307             break;
00308         case TABLE_ARTICLE_ID_UNITE:
00309             return this->idUnite;
00310             break;
00311         case TABLE_ARTICLE_NOM_UNITE:
00312             return this->nomUnite;
00313             break;
00314         case TABLE_ARTICLE_NUMERO_CASIER:
00315             return this->numeroCasier;
00316             break;
00317         default:
00318             qDebug() << Q_FUNC_INFO << champ << "champ inconnu";
00319     }
00320     return QString("");
00321 }
00322
00328 void Article::mettreAJourQuantite(QString quantite)
00329 {
00330     if(idArticle.isEmpty())
00331         return;
00332     if(this->quantite != quantite)
00333     {
00334         #ifdef DEBUG_ARTICLE
00335             qDebug() << Q_FUNC_INFO << "quantite" << quantite;
00336         #endif
00337         this->quantite = quantite;
00338         QString requete = "UPDATE Stock SET Disponible =" + quantite + " WHERE idArticle =" +
      idArticle + ";";
00339         bdd->executer(requete);
00340     }
00341 }
```

9.7 Référence du fichier Article.h

Déclaration de la classe Article.

```
#include <QObject>
#include <QString>
#include <QDebug>
```

Classes

— class Article
La classe Article traite les articles.

Macros

— #define DEBUG_ARTICLE

Énumérations

— enum ChampArticle {
TABLE_ARTICLE_ID_STOCK, TABLE_ARTICLE_ID_ARTICLE, TABLE_ARTICLE_NOM_ARTICLE, TABLE_ARTICLE_ID↩
_TYPE,
TABLE_ARTICLE_NOM_TYPE, TABLE_ARTICLE_ID_COMPTAGE, TABLE_ARTICLE_NOM_COMPTAGE, TABLE_ARTI↩
CLE_CODE_BARRE,
TABLE_ARTICLE_DESIGNATION, TABLE_ARTICLE_QUANTITE, TABLE_ARTICLE_DISPONIBLE, TABLE_ARTICLE_P↩
OIDS,
TABLE_ARTICLE_TARE, TABLE_ARTICLE_ID_UNITE, TABLE_ARTICLE_NOM_UNITE, TABLE_ARTICLE_NUMERO_C↩
ASIER }
Définit les différents champs pour une requête d'un article dans le stock.

9.7.1 Description détaillée

Déclaration de la classe Article.

Auteur

Tranchat Joffrey

Version

1.0

Date

Mercredi 11 Mars 2020

Définition dans le fichier Article.h.

9.7.2 Documentation des macros

9.7.2.1 DEBUG_ARTICLE

```
#define DEBUG_ARTICLE
```

Définition à la ligne 22 du fichier Article.h.

9.7.3 Documentation du type de l'énumération

9.7.3.1 ChampArticle

```
enum ChampArticle
```

Définit les différents champs pour une requête d'un article dans le stock.

Valeurs énumérées

TABLE_ARTICLE_ID_STOCK	
TABLE_ARTICLE_ID_ARTICLE	
TABLE_ARTICLE_NOM_ARTICLE	
TABLE_ARTICLE_ID_TYPE	
TABLE_ARTICLE_NOM_TYPE	
TABLE_ARTICLE_ID_COMPTAGE	
TABLE_ARTICLE_NOM_COMPTAGE	
TABLE_ARTICLE_CODE_BARRE	
TABLE_ARTICLE_DESIGNATION	
TABLE_ARTICLE_QUANTITE	
TABLE_ARTICLE_DISPONIBLE	
TABLE_ARTICLE_POIDS	
TABLE_ARTICLE_TARE	
TABLE_ARTICLE_ID_UNITE	
TABLE_ARTICLE_NOM_UNITE	
TABLE_ARTICLE_NUMERO_CASIER	

Définition à la ligne 28 du fichier Article.h.

```
00029 {
00030     TABLE_ARTICLE_ID_STOCK,
00031     TABLE_ARTICLE_ID_ARTICLE,
00032     TABLE_ARTICLE_NOM_ARTICLE,
00033     TABLE_ARTICLE_ID_TYPE,
00034     TABLE_ARTICLE_NOM_TYPE,
00035     TABLE_ARTICLE_ID_COMPTAGE,
00036     TABLE_ARTICLE_NOM_COMPTAGE,
00037     TABLE_ARTICLE_CODE_BARRE,
00038     TABLE_ARTICLE_DESIGNATION,
00039     TABLE_ARTICLE_QUANTITE,
00040     TABLE_ARTICLE_DISPONIBLE,
00041     TABLE_ARTICLE_POIDS,
00042     TABLE_ARTICLE_TARE,
00043     TABLE_ARTICLE_ID_UNITE,
00044     TABLE_ARTICLE_NOM_UNITE,
00045     TABLE_ARTICLE_NUMERO_CASIER
00046 };
```

9.8 Article.h

```
00001 #ifndef ARTICLE_H
00002 #define ARTICLE_H
00003
00004
00018 #include <QObject>
00019 #include <QString>
00020 #include <QDebug>
00021
00022 #define DEBUG_ARTICLE
00023
00028 enum ChampArticle
00029 {
00030     TABLE_ARTICLE_ID_STOCK,
00031     TABLE_ARTICLE_ID_ARTICLE,
00032     TABLE_ARTICLE_NOM_ARTICLE,
00033     TABLE_ARTICLE_ID_TYPE,
00034     TABLE_ARTICLE_NOM_TYPE,
00035     TABLE_ARTICLE_ID_COMPTAGE,
00036     TABLE_ARTICLE_NOM_COMPTAGE,
00037     TABLE_ARTICLE_CODE_BARRE,
00038     TABLE_ARTICLE_DESIGNATION,
00039     TABLE_ARTICLE_QUANTITE,
00040     TABLE_ARTICLE_DISPONIBLE,
00041     TABLE_ARTICLE_POIDS,
00042     TABLE_ARTICLE_TARE,
00043     TABLE_ARTICLE_ID_UNITE,
00044     TABLE_ARTICLE_NOM_UNITE,
00045     TABLE_ARTICLE_NUMERO_CASIER
00046 };
00047
00048 class Bdd;
00049
00062 class Article : public QObject
00063 {
00064     Q_OBJECT
00065 public:
00066     Article(QObject *parent = nullptr);
00067     ~Article();
00068
00069     bool recupererDonneesArticle(QString idArticle, int numCasier=0);
00070     bool recupererDonneesArticleParNom(QString
      nomArticle, int numCasier=0);
00071     bool recupererDonneesArticleParNumeroCasier(QString
      numeroCasier);
00072     static unsigned int recupererNombreCasiersPourIdArticle(QString
      idArticle);
00073     static unsigned int recupererNombreCasiersPourNomArticle(QString
      nomArticle);
00074     static QVector<QString> recupererNumeroCasierPourIdArticle(QString
      idArticle);
00075     static QVector<QString> recupererNumeroCasierPourNomArticle(QString
      nomArticle);
00076
00077     QString get(ChampArticle champ);
00078     void mettreAJourQuantite(QString quantite);
00079
00080 private:
00081     static Bdd *bdd;
00082     QString idStock;
00083     QString idArticle;
00084     QString nomArticle;
00085     QString idType;
```

```
00086     QString nomType;
00087     QString idComptage;
00088     QString nomComptage;
00089     QString codeBarre;
00090     QString designation;
00091     QString quantite;
00092     QString disponible;
00093     QString poidsArticle;
00094     QString tare;
00095     QString idUnite;
00096     QString nomUnite;
00097     QString numeroCasier;
00098
00099 private slots:
00100
00101 signals:
00102
00103 public slots:
00104 };
00105
00106 #endif // ARTICLE_H
```

9.9 Référence du fichier Bdd.cpp

Définition de la classe Bdd.

```
#include "Bdd.h"
#include <QDebug>
#include <QMessageBox>
```

9.9.1 Description détaillée

Définition de la classe Bdd.

Auteur

Tranchat Joffrey
Legger Pierre-Antoine
Vaira Thierry

Version

1.0

Date

14 Février 2020

Définition dans le fichier Bdd.cpp.

9.10 Bdd.cpp

```
00001 #include "Bdd.h"
00002 #include <QDebug>
00003 #include <QMessageBox>
00004
00019 Bdd* Bdd::bdd = NULL;
00020 int Bdd::nbAcces = 0;
00021
00027 Bdd::Bdd()
00028 {
00029     #ifdef DEBUG_BDD
00030     qDebug() << Q_FUNC_INFO;
00031     #endif
00032     db = QSqlDatabase::addDatabase("QMYSQL");
```

```
00033 }
00034
00040 Bdd::~Bdd()
00041 {
00042     #ifdef DEBUG_BDD
00043     qDebug() << Q_FUNC_INFO;
00044     #endif
00045 }
00046
00053 Bdd* Bdd::getInstance()
00054 {
00055     if(bdd == NULL)
00056         bdd = new Bdd();
00057
00058     nbAcces++;
00059
00060     #ifdef DEBUG_BDD
00061     qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00062     #endif
00063
00064     return bdd;
00065
00066 }
00067
00073 void Bdd::detruireInstance()
00074 {
00075     if(bdd != NULL)
00076     {
00077         nbAcces--;
00078         #ifdef DEBUG_BASEDEDONNEES
00079         qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00080         #endif
00081
00082         if(nbAcces == 0)
00083             delete bdd;
00084     }
00085 }
00086
00093 bool Bdd::connecter()
00094 {
00095     if(!db.isOpen())
00096     {
00097         db.setHostName(HOSTNAME);
00098         db.setUserName(USERNAME);
00099         db.setPassword(PASSWORD);
00100         db.setDatabaseName(DATABASENAME);
00101
00102         #ifdef DEBUG_BDD
00103         qDebug() << Q_FUNC_INFO << "HostName" << db.hostName();
00104         qDebug() << Q_FUNC_INFO << "UserName" << db.userName();
00105         qDebug() << Q_FUNC_INFO << "DatabaseName" << db.databaseName();
00106         #endif
00107         if(db.open())
00108         {
00109             #ifdef DEBUG_BDD
00110             qDebug() << Q_FUNC_INFO << QString::fromUtf8("connexion réussie à %1").arg(
      db.hostName());
00111             #endif
00112
00113             return true;
00114         }
00115         else
00116         {
00117             qDebug() << Q_FUNC_INFO << QString::fromUtf8("erreur : impossible de se connecter à la base de
       données !");
00118
00119             QMessageBox::critical(0, QString::fromUtf8("e-stock"), QString::fromUtf8("Impossible de se
      connecter à la base de données !"));
00120             return false;
00121         }
00122
00123     }
00124     else
00125         return true;
00126 }
00127
00134 bool Bdd::estConnecte()
00135 {
00136     return db.isOpen();
00137 }
00138
00146 bool Bdd::executer(QString requete)
00147 {
00148     QSqlQuery r;
00149     bool retour;
00150
00151     if(db.isOpen())
00152     {
00153         if(requete.contains("UPDATE") || requete.contains("INSERT") || requete.contains("DELETE"))
00154         {
00155             retour = r.exec(requete);
```

```
00156             #ifdef DEBUG_BASEDEDONNEES
00157             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00158             #endif
00159             if(retour)
00160             {
00161                 return true;
00162             }
00163             else
00164             {
00165                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00166                 return false;
00167             }
00168         }
00169         else
00170         {
00171             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00172             return false;
00173         }
00174     }
00175     else
00176         return false;
00177
00178 }
00179
00187 bool Bdd::recuperer(QString requete, QString &donnees)
00188 {
00189     QSqlQuery r;
00190     bool retour;
00191
00192     if(db.isOpen())
00193     {
00194         if(requete.contains("SELECT"))
00195         {
00196             retour = r.exec(requete);
00197             #ifdef DEBUG_BASEDEDONNEES
00198             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00199             #endif
00200             if(retour)
00201             {
00202                 // on se positionne sur l'enregistrement
00203                 r.first();
00204
00205                 // on vérifie l'état de l'enregistrement retourné
00206                 if(!r.isValid())
00207                 {
00208                     #ifdef DEBUG_BASEDEDONNEES
00209                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00210                     #endif
00211                     return false;
00212                 }
00213
00214                 // on récupère sous forme de QString la valeur du champ
00215                 if(r.isNull(0))
00216                 {
00217                     #ifdef DEBUG_BASEDEDONNEES
00218                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucun résultat !");
00219                     #endif
00220                     return false;
00221                 }
00222                 donnees = r.value(0).toString();
00223                 #ifdef DEBUG_BASEDEDONNEES
00224                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00225                 #endif
00226                 return true;
00227             }
00228             else
00229             {
00230                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00231                 return false;
00232             }
00233         }
00234         else
00235         {
00236             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00237             return false;
00238         }
00239     }
00240     else
00241         return false;
00242 }
00243
00251 bool Bdd::recuperer(QString requete, QStringList &donnees)
00252 {
00253     QSqlQuery r;
00254     bool retour;
```

```
00255
00256     if(db.isOpen())
00257     {
00258         if(requete.contains("SELECT"))
00259         {
00260             retour = r.exec(requete);
00261             #ifdef DEBUG_BASEDEDONNEES
00262             qDebug() << QString::fromUtf8("Retour %1 pour la requete : %2").arg(QString::number(retour)).
      arg(requete);
00263             #endif
00264             if(retour)
00265             {
00266                 // on se positionne sur l'enregistrement
00267                 r.first();
00268
00269                 // on vérifie l'état de l'enregistrement retourné
00270                 if(!r.isValid())
00271                 {
00272                     #ifdef DEBUG_BASEDEDONNEES
00273                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00274                     #endif
00275                     return false;
00276                 }
00277
00278                 // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00279                 // et on les stocke dans une liste de QString
00280                 for(int i=0;i<r.record().count();i++)
00281                     if(!r.isNull(i))
00282                         donnees << r.value(i).toString();
00283                 #ifdef DEBUG_BASEDEDONNEES
00284                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00285                 #endif
00286                 return true;
00287             }
00288             else
00289             {
00290                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00291                 return false;
00292             }
00293         }
00294         else
00295         {
00296             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00297             return false;
00298         }
00299     }
00300     else
00301         return false;
00302 }
00303
00311 bool Bdd::recuperer(QString requete, QVector<QString> &donnees)
00312 {
00313     QSqlQuery r;
00314     bool retour;
00315     QString data;
00316
00317     if(db.isOpen())
00318     {
00319         if(requete.contains("SELECT"))
00320         {
00321             retour = r.exec(requete);
00322             #ifdef DEBUG_BASEDEDONNEES
00323             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00324             #endif
00325             if(retour)
00326             {
00327                 // pour chaque enregistrement
00328                 while ( r.next() )
00329                 {
00330                     // on récupère sous forme de QString la valeur du champs sélectionné
00331                     data = r.value(0).toString();
00332
00333                     #ifdef DEBUG_BASEDEDONNEES
00334                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00335                     #endif
00336
00337                     // on stocke l'enregistrement dans le QVector
00338                     donnees.push_back(data);
00339                 }
00340                 #ifdef DEBUG_BASEDEDONNEES
00341                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00342                 #endif
00343                 return true;
00344             }
00345             else
00346             {
00347                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
```

```
00348                 return false;
00349             }
00350         }
00351         else
00352         {
00353             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00354             return false;
00355         }
00356     }
00357     else
00358         return false;
00359 }
00360
00367 bool Bdd::recuperer(QString requete, QVector<QStringList> &donnees)
00368 {
00369     QSqlQuery r;
00370     bool retour;
00371     QStringList data;
00372
00373     if(db.isOpen())
00374     {
00375         if(requete.contains("SELECT"))
00376         {
00377             retour = r.exec(requete);
00378             #ifdef DEBUG_BASEDEDONNEES
00379             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00380             #endif
00381             if(retour)
00382             {
00383                 // pour chaque enregistrement
00384                 while ( r.next() )
00385                 {
00386                     // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00387                     // et on les stocke dans une liste de QString
00388                     for(int i=0;i<r.record().count();i++)
00389                         data << r.value(i).toString();
00390
00391                     #ifdef DEBUG_BASEDEDONNEES
00392                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00393                     /*for(int i=0;i<r.record().count();i++)
00394                         qDebug() << r.value(i).toString();*/
00395                     #endif
00396
00397                     // on stocke l'enregistrement dans le QVector
00398                     donnees.push_back(data);
00399
00400                     // on efface la liste de QString pour le prochain enregistrement
00401                     data.clear();
00402                 }
00403                 #ifdef DEBUG_BASEDEDONNEES
00404                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00405                 #endif
00406                 return true;
00407             }
00408             else
00409             {
00410                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00411                 return false;
00412             }
00413         }
00414         else
00415         {
00416             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00417             return false;
00418         }
00419     }
00420     else
00421         return false;
00422 }
```

9.11 Référence du fichier Bdd.h

Déclaration de la classe Bdd.

```
#include <QtSql/QtSql>
#include <QtSql/QSqlDatabase>
```

Classes

— class Bdd
Déclaration de la classe utilisant la base de données.

Macros

— #define DATABASENAME "e-stock"
— #define DEBUG_BASEDEDONNEES
— #define DEBUG_BDD
— #define HOSTNAME "localhost"
— #define PASSWORD "password"
— #define USERNAME "estock"

9.11.1 Description détaillée

Déclaration de la classe Bdd.

Auteur

Tranchat Joffrey
Legger Pierre-Antoine

Version

1.0

Date

Vendredi 14 Février

Définition dans le fichier Bdd.h.

9.11.2 Documentation des macros

9.11.2.1 DATABASENAME

```
#define DATABASENAME "e-stock"
```

Définition à la ligne 13 du fichier Bdd.h.

Référencé par Bdd : :connecter().

9.11.2.2 DEBUG_BASEDEDONNEES

```
#define DEBUG_BASEDEDONNEES
```

Définition à la ligne 8 du fichier Bdd.h.

9.11.2.3 DEBUG_BDD

```
#define DEBUG_BDD
```

Définition à la ligne 7 du fichier Bdd.h.

9.11.2.4 HOSTNAME

```
#define HOSTNAME "localhost"
```

Définition à la ligne 10 du fichier Bdd.h.

Référencé par Bdd : :connecter().

9.11.2.5 PASSWORD

```
#define PASSWORD "password"
```

Définition à la ligne 12 du fichier Bdd.h.

Référencé par Bdd : :connecter().

9.11.2.6 USERNAME

```
#define USERNAME "estock"
```

Définition à la ligne 11 du fichier Bdd.h.

Référencé par Bdd : :connecter().

9.12 Bdd.h

```
00001 #ifndef BDD_H
00002 #define BDD_H
00003
00004 #include <QtSql/QtSql>
00005 #include <QtSql/QSqlDatabase>
00006
00007 #define DEBUG_BDD
00008 #define DEBUG_BASEDEDONNEES
00009
00010 #define HOSTNAME       "localhost"
00011 #define USERNAME       "estock"
00012 #define PASSWORD       "password"
00013 #define DATABASENAME   "e-stock"
00014
00042 class Bdd
00043 {
00044 public:
00045     static Bdd* getInstance();
00046     static void detruireInstance();
00047
00048     bool connecter();
00049     bool estConnecte();
00050
00051     /* uniquement pour : UPDATE, INSERT et DELETE */
00052     bool executer(QString requete);
00053
00054     /* uniquement pour : SELECT */
00055     bool recuperer(QString requete, QString &donnees); // 1 -> 1
00056     bool recuperer(QString requete, QStringList &donnees); // 1 -> 1..*
00057     bool recuperer(QString requete, QVector<QString> &donnees); // 1..* -> 1
00058     bool recuperer(QString requete, QVector<QStringList> &donnees); // 1..* -> 1..*
00059
00060 private:
00061     Bdd();
00062     ~Bdd();
00063     QSqlDatabase db;
00064     static Bdd* bdd;
00065     static int nbAcces;
00066 };
00067
00068 #endif // BDD_H
```

9.13 Référence du fichier Casier.cpp

Définition de la classe Casier.

```
#include "Casier.h"
```

9.13.1 Description détaillée

Définition de la classe Casier.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

samedi 28 Mars 2020

Définition dans le fichier Casier.cpp.

9.14 Casier.cpp

```
00001 #include "Casier.h"
00002
00023 Casier::Casier(int numero, QWidget *parent) : QPushButton(parent), numero(
      numero), ouvert(false)
00024 {
00025     qDebug() << Q_FUNC_INFO << numero << this;
00026     setText("Casier " + QString::number(numero));
00027
00031     setMaximumHeight(100);
00032     setContentsMargins(10, 0, 10, 0); // Marges : Gauche Haut Droite Bas
00036     //setStyleSheet("background-color: rgb(85, 85, 85);font-size: 18px;"); // inconnu
00037     setStyleSheet("background-color: rgb(239, 41, 41);font-size: 18px;"); // fermé
00038
00042     connect(this, SIGNAL(clicked(bool)), this, SLOT(gererEtat()));
00043 }
00044
00049 Casier::~Casier()
00050 {
00051     qDebug() << Q_FUNC_INFO << numero << this;
00052 }
00053
00059 int Casier::getNumero() const
00060 {
00061     return numero;
00062 }
00063
00069 bool Casier::estOuvert() const
00070 {
00071     return ouvert;
00072 }
00073
00079 void Casier::setOuvert(bool ouvert)
00080 {
00081     if(this->ouvert != ouvert)
00082     {
00083         this->ouvert = ouvert;
00084         if(ouvert)
00085             setStyleSheet("background-color: rgb(115, 210, 22);font-size: 18px;"); // ouvert
00086         else
00087             setStyleSheet("background-color: rgb(239, 41, 41);font-size: 18px;"); // fermé
00088         emit estOuvert(numero, ouvert);
00089     }
00090 }
```

```
00091
00096 void Casier::ouvrir()
00097 {
00102     // simule le casier ouvert
00103     setOuvert(true);
00104 }
00105
00114 void Casier::gererEtat()
00115 {
00116     qDebug() << Q_FUNC_INFO << numero << this;
00117     if(!ouvert)
00118     {
00119         ouvrir();
00120     }
00121     #ifdef SIMULATION_CASIER
00122     else
00123     {
00124         // simule le casier fermé
00125         setOuvert(false);
00126     }
00127     #endif
00128 }
```

9.15 Référence du fichier Casier.h

Déclaration de la classe Casier.

```
#include <QtWidgets>
```

Classes

— class Casier
La classe Casier gère le casier contenant des articles.

Macros

— #define SIMULATION_CASIER

9.15.1 Description détaillée

Déclaration de la classe Casier.

Auteur

Version

1.0

Date

Définition dans le fichier Casier.h.

9.15.2 Documentation des macros

9.15.2.1 SIMULATION_CASIER

```
#define SIMULATION_CASIER
```

Définition à la ligne 19 du fichier Casier.h.

9.16 Casier.h

```
00001 #ifndef CASIER_H
00002 #define CASIER_H
00003
00017 #include <QtWidgets>
00018
00019 #define SIMULATION_CASIER
00020
00034 class Casier : public QPushButton
00035 {
00036     Q_OBJECT
00037 public:
00038     Casier(int numero, QWidget *parent=0);
00039     ~Casier();
00040
00041     int getNumero() const;
00042     bool estOuvert() const;
00043     void setOuvert(bool ouvert);
00044     void ouvrir();
00045
00046 private:
00047     int numero;
00048     bool ouvert;
00049
00050 public slots:
00051     void gererEtat();
00052
00053 signals:
00054     void estOuvert(int numero, bool etat);
00055
00056 };
00057
00058 #endif // CASIER_H
```

9.17 Référence du fichier Changelog.md

9.18 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003 r82 | palegger | 2020-04-02 05:30:03 +0200 (jeu. 02 avril 2020) | 1 ligne
00004
00005 correction diagramme
00006
00007 r81 | jtranchat | 2020-04-01 23:57:17 +0200 (mer. 01 avril 2020) | 1 ligne
00008
00009 ajout diagramme de classe scénario mettre à jour le stock
00010
00011 r80 | jtranchat | 2020-04-01 21:47:00 +0200 (mer. 01 avril 2020) | 1 ligne
00012
00013 ajout de la documentation doxygen pour la version 0.1
00014
00015 r79 | jtranchat | 2020-04-01 21:16:21 +0200 (mer. 01 avril 2020) | 1 ligne
00016
00017 création du tag 0.1
00018
00019 r78 | palegger | 2020-04-01 20:13:44 +0200 (mer. 01 avril 2020) | 1 ligne
00020
00021 ajout diagramme de sequence et de classe
00022
00023 r77 | palegger | 2020-03-31 19:27:41 +0200 (mar. 31 mars 2020) | 1 ligne
00024
00025 Amelioration de la gestion de plusieurs casiers
00026
00027 r76 | jtranchat | 2020-03-31 12:04:31 +0200 (mar. 31 mars 2020) | 1 ligne
00028
00029 ajout fonction envoyerRequetePoid
00030
00031 r75 | jtranchat | 2020-03-29 18:40:55 +0200 (dim. 29 mars 2020) | 1 ligne
00032
```

```
00033 modification diagramme mettre à jour le stock
00034
00035 r74 | jtranchat | 2020-03-28 17:06:48 +0100 (sam. 28 mars 2020) | 1 ligne
00036
00037 Ajout/modification de commentaire dans le projet
00038
00039 r73 | jtranchat | 2020-03-28 15:18:53 +0100 (sam. 28 mars 2020) | 1 ligne
00040
00041 ajout de commentaire dans la classe article
00042
00043 r72 | jtranchat | 2020-03-28 15:03:00 +0100 (sam. 28 mars 2020) | 1 ligne
00044
00045 ajout envoie requete trame poid au démarrage
00046
00047 r71 | jtranchat | 2020-03-28 14:48:25 +0100 (sam. 28 mars 2020) | 1 ligne
00048
00049 realisation todo dans traiterTramePoids()
00050
00051 r70 | tvaira | 2020-03-28 09:39:43 +0100 (sam. 28 mars 2020) | 1 ligne
00052
00053 Mise a jour Bouml
00054
00055 r69 | tvaira | 2020-03-28 09:22:52 +0100 (sam. 28 mars 2020) | 2 lignes
00056
00057 Ajout TODO pour l'itération 2
00058
00059 r68 | tvaira | 2020-03-28 08:45:30 +0100 (sam. 28 mars 2020) | 2 lignes
00060
00061 Révision de code
00062
00063 r67 | tvaira | 2020-03-28 08:37:15 +0100 (sam. 28 mars 2020) | 1 ligne
00064
00065 Modification BD
00066
00067 r66 | palegger | 2020-03-28 04:48:31 +0100 (sam. 28 mars 2020) | 1 ligne
00068
00069 Ajout fonctionnalite prise en charge un article dans plusieurs casiers
00070
00071 r65 | palegger | 2020-03-27 19:17:54 +0100 (ven. 27 mars 2020) | 1 ligne
00072
00073 resolution erreur liée a l'affichage
00074
00075 r64 | jtranchat | 2020-03-26 23:56:51 +0100 (jeu. 26 mars 2020) | 1 ligne
00076
00077 fonction traiter trame poids dans supervision + fonction compter() + fonction arrondie()
00078
00079 r63 | jtranchat | 2020-03-24 18:09:49 +0100 (mar. 24 mars 2020) | 1 ligne
00080
00081 definition fonction mettreAJourQuantite dans article
00082
00083 r62 | tvaira | 2020-03-22 16:29:34 +0100 (dim. 22 mars 2020) | 1 ligne
00084
00085 Modification exemples requêtes SQL
00086
00087 r61 | tvaira | 2020-03-22 16:29:03 +0100 (dim. 22 mars 2020) | 2 lignes
00088
00089 Révision du code pour Article et Supervision
00090
00091 r60 | tvaira | 2020-03-22 15:34:38 +0100 (dim. 22 mars 2020) | 1 ligne
00092
00093 Modification v0.3 SQL de la base de données
00094
00095 r59 | tvaira | 2020-03-22 15:33:48 +0100 (dim. 22 mars 2020) | 2 lignes
00096
00097 Ajout de la classe Armoire
00098
00099 r58 | tvaira | 2020-03-22 09:11:24 +0100 (dim. 22 mars 2020) | 2 lignes
00100
00101 Révision de code de la classe Communication
00102
00103 r57 | jtranchat | 2020-03-22 00:36:09 +0100 (dim. 22 mars 2020) | 1 ligne
00104
00105 ajout diagramme de mise à jour du stock
00106
00107 r56 | palegger | 2020-03-21 20:50:21 +0100 (sam. 21 mars 2020) | 1 ligne
00108
00109 Ajout Diagrame de séquence pour authentification par champs
00110
00111 r55 | palegger | 2020-03-21 20:37:57 +0100 (sam. 21 mars 2020) | 1 ligne
00112
00113 Ajout Diagrame de séquence pour authentification par champs
00114
00115 r54 | tvaira | 2020-03-21 18:07:02 +0100 (sam. 21 mars 2020) | 1 ligne
00116
00117 Validation des diagrammes de sequence
00118
00119 r53 | palegger | 2020-03-21 04:52:49 +0100 (sam. 21 mars 2020) | 1 ligne
00120
00121 Modification diagrame de séquence connexion par badge et ajout diagrame connexion par champs
00122
00123 r52 | jtranchat | 2020-03-20 06:00:57 +0100 (ven. 20 mars 2020) | 1 ligne
```

```
00124
00125 ajout de commentaire de la classe article plus ajout de commentaire dans Supervision
00126
00127 r51 | tvaira | 2020-03-19 16:10:41 +0100 (jeu. 19 mars 2020) | 1 ligne
00128
00129 Ajout du fichier SQL de base
00130
00131 r50 | palegger | 2020-03-19 03:18:49 +0100 (jeu. 19 mars 2020) | 1 ligne
00132
00133 Ajout de commentaires et de laisser vide si pas de mots de passe
00134
00135 r49 | tvaira | 2020-03-17 12:06:14 +0100 (mar. 17 mars 2020) | 2 lignes
00136
00137 Validation bouton Se déconnecter
00138
00139 r48 | tvaira | 2020-03-17 11:54:05 +0100 (mar. 17 mars 2020) | 2 lignes
00140
00141 Validation de crypterMotDepasse() (cf. define CHANGE_PASSWORD_BEFORE)
00142
00143 r47 | tvaira | 2020-03-17 11:24:31 +0100 (mar. 17 mars 2020) | 2 lignes
00144
00145 Exemple recherche articles pour FenetreMenu
00146
00147 r46 | tvaira | 2020-03-17 10:36:28 +0100 (mar. 17 mars 2020) | 2 lignes
00148
00149 Révision de code
00150
00151 r45 | jtranchat | 2020-03-13 16:01:20 +0100 (ven. 13 mars 2020) | 1 ligne
00152
00153 supression ROADBOOK et mise à jour du todo
00154
00155 r44 | palegger | 2020-03-13 15:38:50 +0100 (ven. 13 mars 2020) | 1 ligne
00156
00157 Liaison avec Esp effectuer
00158
00159 r43 | palegger | 2020-03-13 14:00:32 +0100 (ven. 13 mars 2020) | 1 ligne
00160
00161 Ajout page stock Ihm
00162
00163 r42 | jtranchat | 2020-03-13 10:30:28 +0100 (ven. 13 mars 2020) | 1 ligne
00164
00165 ajout fonction berifierTypeTrame
00166
00167 r41 | jtranchat | 2020-03-13 10:27:47 +0100 (ven. 13 mars 2020) | 1 ligne
00168
00169 correction bug comptage automatique
00170
00171 r40 | jtranchat | 2020-03-12 15:33:28 +0100 (jeu. 12 mars 2020) | 1 ligne
00172
00173 msie en place traiter trame
00174
00175 r39 | jtranchat | 2020-03-12 12:32:06 +0100 (jeu. 12 mars 2020) | 1 ligne
00176
00177 mise en place prendre et rapporter article automatique
00178
00179 r38 | jtranchat | 2020-03-12 11:47:15 +0100 (jeu. 12 mars 2020) | 1 ligne
00180
00181 mise en place du comptage automatique
00182
00183 r37 | palegger | 2020-03-12 10:48:23 +0100 (jeu. 12 mars 2020) | 1 ligne
00184
00185 Creation méthode de la classe Communication
00186
00187 r36 | palegger | 2020-03-12 10:40:05 +0100 (jeu. 12 mars 2020) | 1 ligne
00188
00189 Mise a jour Ihm
00190
00191 r35 | palegger | 2020-03-12 10:01:13 +0100 (jeu. 12 mars 2020) | 2 lignes
00192
00193 Mise a jour ROADBOOK
00194
00195 r34 | palegger | 2020-03-11 11:57:51 +0100 (mer. 11 mars 2020) | 1 ligne
00196
00197 Ajout crytage mot de passe
00198
00199 r33 | palegger | 2020-03-11 10:47:57 +0100 (mer. 11 mars 2020) | 1 ligne
00200
00201 Ajout verification de la verification de la date de validite
00202
00203 r32 | jtranchat | 2020-03-11 10:44:39 +0100 (mer. 11 mars 2020) | 1 ligne
00204
00205 ajout de la classe Article
00206
00207 r31 | tvaira | 2020-03-07 09:47:19 +0100 (sam. 07 mars 2020) | 2 lignes
00208
00209 Révision du code
00210
00211 r30 | jtranchat | 2020-03-06 17:01:26 +0100 (ven. 06 mars 2020) | 1 ligne
00212
00213 mise à jour ROADBOOK
00214
```

```
00215 r29 | jtranchat | 2020-03-06 16:36:20 +0100 (ven. 06 mars 2020) | 1 ligne
00216
00217 ajout de la fonction ajouter article
00218
00219 r28 | palegger | 2020-03-06 16:10:24 +0100 (ven. 06 mars 2020) | 1 ligne
00220
00221 Correction orthographe, Ajout fonction, Ajout fichier bouml avec diagramme de classe
00222
00223 r27 | jtranchat | 2020-03-06 11:37:55 +0100 (ven. 06 mars 2020) | 1 ligne
00224
00225
00226 r26 | jtranchat | 2020-03-06 10:35:58 +0100 (ven. 06 mars 2020) | 1 ligne
00227
00228 ajout de m'essage d'erreur dans ajouter article
00229
00230 r25 | palegger | 2020-03-06 10:22:50 +0100 (ven. 06 mars 2020) | 1 ligne
00231
00232 Ajout connexion RFID
00233
00234 r24 | jtranchat | 2020-03-05 15:54:22 +0100 (jeu. 05 mars 2020) | 1 ligne
00235
00236 ajout page prendre et rapporter artiv
00237
00238 r23 | palegger | 2020-03-05 12:37:45 +0100 (jeu. 05 mars 2020) | 1 ligne
00239
00240 Ajustement pour connexion identifiant
00241
00242 r22 | palegger | 2020-03-05 12:29:45 +0100 (jeu. 05 mars 2020) | 1 ligne
00243
00244 Connexion par identifiant fonctionnel
00245
00246 r21 | palegger | 2020-03-05 10:47:21 +0100 (jeu. 05 mars 2020) | 1 ligne
00247
00248 Connection bouton Se Connecter
00249
00250 r20 | jtranchat | 2020-03-05 10:45:31 +0100 (jeu. 05 mars 2020) | 1 ligne
00251
00252 correction bug
00253
00254 r19 | jtranchat | 2020-03-05 10:28:50 +0100 (jeu. 05 mars 2020) | 1 ligne
00255
00256 sa marche
00257
00258 r18 | jtranchat | 2020-03-05 00:23:18 +0100 (jeu. 05 mars 2020) | 1 ligne
00259
00260 ajout page menu principal et prendre ou rajouter un objet
00261
00262 r17 | jtranchat | 2020-03-05 00:03:05 +0100 (jeu. 05 mars 2020) | 1 ligne
00263
00264 mise a jour roadbook
00265
00266 r16 | palegger | 2020-03-04 16:09:22 +0100 (mer. 04 mars 2020) | 1 ligne
00267
00268 correction classe Utilisateur
00269
00270 r15 | palegger | 2020-03-04 13:50:55 +0100 (mer. 04 mars 2020) | 1 ligne
00271
00272 Ajout classe utilisateur
00273
00274 r14 | jtranchat | 2020-03-04 12:32:39 +0100 (mer. 04 mars 2020) | 1 ligne
00275
00276 modification pageAjouter
00277
00278 r13 | jtranchat | 2020-03-04 12:16:06 +0100 (mer. 04 mars 2020) | 1 ligne
00279
00280 ajout page pour ajuter un article
00281
00282 r12 | palegger | 2020-03-04 01:34:59 +0100 (mer. 04 mars 2020) | 1 ligne
00283
00284 Mise en place Ihm identifiant
00285
00286 r11 | palegger | 2020-03-03 22:00:30 +0100 (mar. 03 mars 2020) | 1 ligne
00287
00288 verification identifiant badge et affichage message d'erreur et passage a la fenetre identifiant
00289
00290 r10 | tvaira | 2020-02-15 14:05:26 +0100 (sam. 15 févr. 2020) | 3 lignes
00291
00292 Ajout des méthodes pour effecteur des requêtes SQL dans la classe Bdd
00293 Ajout des mécanismes Qt aux classes de l'application
00294
00295 r9 | jtranchat | 2020-02-14 11:36:16 +0100 (ven. 14 févr. 2020) | 1 ligne
00296
00297 ajout de commentaire
00298
00299 r8 | jtranchat | 2020-02-14 10:31:04 +0100 (ven. 14 févr. 2020) | 1 ligne
00300
00301 mise en place d'un singleton pour la classe Bdd
00302
00303 r7 | palegger | 2020-02-14 10:23:07 +0100 (ven. 14 févr. 2020) | 1 ligne
00304
00305 Lecture badge Rfid
```

```
00306
00307 r6 | jtranchat | 2020-02-14 09:54:13 +0100 (ven. 14 févr. 2020) | 1 ligne
00308
00309 ajout de la connection avec la Bdd
00310
00311 r5 | palegger | 2020-02-14 09:01:45 +0100 (ven. 14 févr. 2020) | 1 ligne
00312
00313 Ajout relation entre classes
00314
00315 r4 | jtranchat | 2020-02-13 15:57:31 +0100 (jeu. 13 févr. 2020) | 1 ligne
00316
00317 mise à jour ROADBOOK
00318
00319 r3 | jtranchat | 2020-02-13 12:29:48 +0100 (jeu. 13 févr. 2020) | 1 ligne
00320
00321 ajout des fichiers sources du projet
00322
00323 r2 | jtranchat | 2020-02-13 10:34:06 +0100 (jeu. 13 févr. 2020) | 1 ligne
00324
00325 ajout fichier ROADBOOK + TODO
00326
00327 r1 | www-data | 2020-02-01 15:03:10 +0100 (sam. 01 févr. 2020) | 1 ligne
00328
00329 Creating initial repository structure
```

9.19 Référence du fichier CodeBarre.cpp

Définition de la classe CodeBare.

```
#include "CodeBarre.h"
#include "Bdd.h"
#include <QDebug>
```

9.19.1 Description détaillée

Définition de la classe CodeBare.

Auteur

Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition dans le fichier CodeBarre.cpp.

9.20 CodeBarre.cpp

```
00001 #include "CodeBarre.h"
00002 #include "Bdd.h"
00003 #include <QDebug>
00004
00023 CodeBarre::CodeBarre(QObject *parent) : QObject(parent),
      etatPrendreOuAjouter(false), quantiteObjet(0)
00024 {
00025     #ifdef DEBUG_CODE_BARRE
00026         qDebug() << Q_FUNC_INFO;
00027     #endif
00028     bdd = Bdd::getInstance();
00029 }
00030
```

```
00035 CodeBarre::~CodeBarre()
00036 {
00037     Bdd::detruireInstance();
00038     #ifdef DEBUG_CODE_BARRE
00039         qDebug() << Q_FUNC_INFO;
00040     #endif
00041 }
00042
00048 void CodeBarre::setEtatPrendreOuAjouter(bool etat)
00049 {
00050     this->etatPrendreOuAjouter = etat;
00051
00052     #ifdef DEBUG_CODE_BARRE
00053         qDebug() << Q_FUNC_INFO << "etatPrendreOuAjouter" << this->
      etatPrendreOuAjouter ;
00054     #endif
00055 }
00056
00062 void CodeBarre::setQuantiteObjet(unsigned int quantite)
00063 {
00064     quantiteObjet = quantite;
00065 }
00066
00072 void CodeBarre::prendreOuAjouter(bool etat)
00073 {
00074     setEtatPrendreOuAjouter(etat);
00075 }
00076
00082 void CodeBarre::changerQuantiteObjet(int quantite)
00083 {
00084     setQuantiteObjet(quantite);
00085
00086     #ifdef DEBUG_CODE_BARRE
00087         qDebug() << Q_FUNC_INFO << "quantiteObjet" << quantiteObjet;
00088     #endif
00089 }
00090
00096 void CodeBarre::traiterCodeBarre(QString codeBarre)
00097 {
00098     #ifdef DEBUG_CODE_BARRE
00099         qDebug() << Q_FUNC_INFO << "codeBarre" << codeBarre;
00100     #endif
00101     QString codeBarreCorriger = corrigerCodeBarre(codeBarre);
00102
00103     if(!etatPrendreOuAjouter)
00104     {
00105         // prendre objet
00106         emit prendreObjet(codeBarreCorriger);
00107     }
00108     else
00109     {
00110         // ajouter objet
00111         emit ajouterObjet(codeBarreCorriger);
00112     }
00113 }
00114
00120 unsigned int CodeBarre::getQuantiteObjet()
00121 {
00122     return quantiteObjet;
00123 }
00124
00130 QString CodeBarre::corrigerCodeBarre(QString codeBarre)
00131 {
00132     QString codeBarreCorrige = "";
00133
00134     if(!codeBarre.isEmpty())
00135     {
00136         // effectue les remplacements des touches QWERTY en touches AZERTY
00137         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("à"), "0");
00138         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("&"), "1");
00139         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("é"), "2");
00140         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("\""), "3");
00141         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("'"), "4");
00142         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("("), "5");
00143         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("-"), "6");
00144         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("è"), "7");
00145         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("_"), "8");
00146         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("ç"), "9");
00147         codeBarreCorrige = codeBarre.replace(QString::fromUtf8("\n"), "");
00148     }
00149     #ifdef DEBUG_CODE_BARRE
00150         qDebug() << Q_FUNC_INFO << "codeBarreCorrige" << codeBarreCorrige;
00151     #endif
00152     return codeBarreCorrige;
00153 }
00154
00161 unsigned int CodeBarre::recupererQuantiteMaxParNumeroCasier(
      QString numeroCasier)
00162 {
00163     QString requete = "SELECT Stock.Quantite FROM Stock WHERE Stock.numeroCasier = '" + numeroCasier + "'";
00164
```

```
00165     QString donnees;
00166     bdd->recuperer(requete, donnees);
00167
00168     return donnees.toUInt();
00169 }
00170
00177 unsigned int CodeBarre::recupererQuantiteDisponibleParNumeroCasier
      (QString numeroCasier)
00178 {
00179     QString requete = "SELECT Stock.Disponible FROM Stock WHERE Stock.numeroCasier = '" + numeroCasier + "'
      ";
00180
00181     QString donnees;
00182     bdd->recuperer(requete, donnees);
00183
00184     return donnees.toUInt();
00185 }
00186
00193 unsigned int CodeBarre::recupererIdArticleAvecCodeBarres(QString
       codeBarre)
00194 {
00195     QString requete = "SELECT idArticle FROM Article WHERE Code = '" + codeBarre + "'";
00196
00197     QString donnees;
00198     bdd->recuperer(requete, donnees);
00199
00200     return donnees.toUInt();
00201 }
```

9.21 Référence du fichier CodeBarre.h

Déclaration de la classe CodeBare.

```
#include <QObject>
```

Classes

— class CodeBarre
Déclaration de la classe CodeBarre.

Macros

— #define DEBUG_CODE_BARRE

9.21.1 Description détaillée

Déclaration de la classe CodeBare.

Auteur

Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition dans le fichier CodeBarre.h.

9.21.2 Documentation des macros

9.21.2.1 DEBUG_CODE_BARRE

```
#define DEBUG_CODE_BARRE
```

Définition à la ligne 19 du fichier CodeBarre.h.

9.22 CodeBarre.h

```
00001 #ifndef CODEBARRE_H
00002 #define CODEBARRE_H
00003
00017 #include <QObject>
00018
00019 #define DEBUG_CODE_BARRE
00020
00021 class Bdd;
00022
00035 class CodeBarre : public QObject
00036 {
00037     Q_OBJECT
00038
00039 public:
00040     CodeBarre(QObject *parent = nullptr);
00041     ~CodeBarre();
00042
00043     unsigned int recupererQuantiteMaxParNumeroCasier(QString
      numeroCasier);
00044     unsigned int recupererQuantiteDisponibleParNumeroCasier(
      QString numeroCasier);
00045     unsigned int recupererIdArticleAvecCodeBarres(QString codeBarre);
00046     unsigned int getQuantiteObjet();
00047
00048 private:
00049     Bdd *bdd;
00050     bool etatPrendreOuAjouter;
00051     unsigned int quantiteObjet;
00052
00053     void setEtatPrendreOuAjouter(bool etat);
00054     void setQuantiteObjet(unsigned int quantite);
00055     QString corrigerCodeBarre(QString codeBarre);
00056
00057 public slots:
00058     void prendreOuAjouter(bool etat);
00059     void traiterCodeBarre(QString codeBarre);
00060     void changerQuantiteObjet(int quantite);
00061
00062 signals:
00063     void ajouterObjet(QString);
00064     void prendreObjet(QString);
00065 };
00066
00067 #endif // CODEBARRE_H
```

9.23 Référence du fichier Communication.cpp

Déclaration de la classe Communication.

```
#include "Communication.h"
#include <QDebug>
#include <QObject>
```

9.23.1 Description détaillée

Déclaration de la classe Communication.

Auteur

TRANCHAT Joffrey
LEGGER Pierre-Antoine

Version

1.0

Date

jeudi 12 Mars2020

Définition dans le fichier Communication.cpp.

9.24 Communication.cpp

```
00001 #include "Communication.h"
00002 #include <QDebug>
00003 #include <QObject>
00004
00023 Communication::Communication(QObject *parent) :
      QObject(parent), port(new QSerialPort(this)), trameBrute("\0"), nomPort(
      SERIAL_PORT_NAME)
00024 {
00025     #ifdef DEBUG_COMMUNICATION
00026         qDebug() << Q_FUNC_INFO;
00027     #endif
00028 }
00029
00034 Communication::~Communication()
00035 {
00036     port->close();
00037     #ifdef DEBUG_COMMUNICATION
00038         qDebug() << Q_FUNC_INFO;
00039     #endif
00040 }
00041
00046 void Communication::demarrerCommunicationPort()
00047 {
00048     #ifdef DEBUG_COMMUNICATION
00049         qDebug() << Q_FUNC_INFO;
00050     #endif
00051     configurerPort();
00052     ouvrirPort();
00053 }
00054
00059 void Communication::arreterCommunicationPort()
00060 {
00061     #ifdef DEBUG_COMMUNICATION
00062         qDebug() << Q_FUNC_INFO;
00063     #endif
00064     port->close();
00065 }
00066
00071 void Communication::configurerPort()
00072 {
00073     #ifdef DEBUG_COMMUNICATION
00074         qDebug() << Q_FUNC_INFO;
00075     #endif
00076     port->setPortName(nomPort);
00077     port->setBaudRate(QSerialPort::Baud9600);
00078     port->setDataBits(QSerialPort::Data8);
00079     port->setParity(QSerialPort::NoParity);
00080     port->setStopBits(QSerialPort::OneStop);
00081     port->setFlowControl(QSerialPort::NoFlowControl);
00082 }
00083
00088 void Communication::ouvrirPort()
00089 {
00090     if (port->open(QIODevice::ReadWrite))
00091     {
00092         #ifdef DEBUG_COMMUNICATION
```

```
00093             qDebug() << Q_FUNC_INFO << "connecté au port" << nomPort;
00094         #endif
00095         connect(port, SIGNAL(readyRead()), this, SLOT(recevoirTrame()));
00096     }
00097     else
00098     {
00099         #ifdef DEBUG_COMMUNICATION
00100             qDebug() << Q_FUNC_INFO << "erreur ouverture du port" << port->error();
00101         #endif
00102     }
00103 }
00104
00110 void Communication::setNomPort(QString nouveauPortSerie)
00111 {
00112     nomPort = nouveauPortSerie;
00113     #ifdef DEBUG_COMMUNICATION
00114         qDebug() << Q_FUNC_INFO << nomPort;
00115     #endif
00116 }
00117
00123 void Communication::envoyerTrame(QString trame)
00124 {
00125     if (port->isOpen())
00126     {
00127        port->write(trame.toLatin1());
00128     }
00129 }
00130
00135 void Communication::recevoirTrame()
00136 {
00137     trameBrute = "\0";
00138
00139     while (port->waitForReadyRead(500))
00140     {
00141         trameBrute.append(port->readAll());
00142     }
00143
00144     if(verifierTrame(trameBrute))
00145         traiterTrame(trameBrute);
00146 }
00147
00154 bool Communication::verifierTrame(QString trame)
00155 {
00156     #ifdef DEBUG_COMMUNICATION
00157         qDebug() << Q_FUNC_INFO << trame;
00158     #endif
00159     if(!trame.startsWith(EN_TETE))
00160     {
00161         return false;
00162     }
00163     if(!trame.endsWith(DELIMITEUR_FIN))
00164     {
00165         return false;
00166     }
00167
00168     return true;
00169 }
00170
00176 void Communication::traiterTrame(QString trame)
00177 {
00178     if(trame.startsWith(EN_TETE + DELIMITEUR_CHAMP +
      TRAME_OUVERTURE + DELIMITEUR_CHAMP))
00179     {
00180         emit envoieTrameOuverture(trame);
00181     }
00182     else if(trame.startsWith(EN_TETE + DELIMITEUR_CHAMP +
      TRAME_ETAT + DELIMITEUR_CHAMP))
00183     {
00184         emit envoieTrameEtat(trame);
00185     }
00186     else if(trame.startsWith(EN_TETE + DELIMITEUR_CHAMP +
      TRAME_POIDS + DELIMITEUR_CHAMP))
00187     {
00188         emit envoieTramePoids(trame);
00189     }
00190 }
00191
00197 void Communication::envoyerRequetePoids(QString numeroCasier)
00198 {
00199     QString trame = "CASIERS;3;" + numeroCasier + ";\r\n";
00200     envoyerTrame(trame);
00201 }
00202
00208 void Communication::envoyerRequeteOuvertureCasier(QString
      numeroCasier)
00209 {
00210     QString trame = "CASIERS;1;" + numeroCasier + ";\r\n";
00211     envoyerTrame(trame);
00212 }
00213
00219 void Communication::envoyerRequeteEtatCasiers(QString numeroCasier)
```

```
00220 {
00221     QString trame = "CASIERS;2;" + numeroCasier + ";\r\n";
00222     envoyerTrame(trame);
00223 }
```

9.25 Référence du fichier Communication.h

Définition de la classe Communication.

```
#include <QObject>
#include <QString>
#include <QSerialPort>
```

Classes

— class Communication
La classe Communication permet de communiquer avec le port série.

Macros

— #define DELIMITEUR_CHAMP QString(";")
— #define DELIMITEUR_FIN QString("\r\n")
— #define EN_TETE QString("CASIERS")
— #define SERIAL_PORT_NAME "/dev/ttyUSB0"
Définit le nom du port série associé au SE.
— #define TRAME_ETAT QString("2")
— #define TRAME_OUVERTURE QString("1")
— #define TRAME_POIDS QString("3")

9.25.1 Description détaillée

Définition de la classe Communication.

La classe Communication permet de communiquer avec le port série

Auteur

TRANCHAT Joffrey
LEGGER Pierre-Antoine

Version

1.0

Définition dans le fichier Communication.h.

9.25.2 Documentation des macros

9.25.2.1 DELIMITEUR_CHAMP

```
#define DELIMITEUR_CHAMP QString(";")
```

Définition à la ligne 28 du fichier Communication.h.

Référencé par Communication : :traiterTrame().

9.25.2.2 DELIMITEUR_FIN

```
#define DELIMITEUR_FIN QString("\r\n")
```

Définition à la ligne 29 du fichier Communication.h.

Référencé par Communication : :verifierTrame().

9.25.2.3 EN_TETE

```
#define EN_TETE QString("CASIERS")
```

Définition à la ligne 27 du fichier Communication.h.

Référencé par Communication : :traiterTrame(), et Communication : :verifierTrame().

9.25.2.4 SERIAL_PORT_NAME

```
#define SERIAL_PORT_NAME "/dev/ttyUSB0"
```

Définit le nom du port série associé au SE.

Définition à la ligne 24 du fichier Communication.h.

9.25.2.5 TRAME_ETAT

```
#define TRAME_ETAT QString("2")
```

Définition à la ligne 33 du fichier Communication.h.

Référencé par Communication : :traiterTrame().

9.25.2.6 TRAME_OUVERTURE

```
#define TRAME_OUVERTURE QString("1")
```

Définition à la ligne 32 du fichier Communication.h.

Référencé par Communication : :traiterTrame().

9.25.2.7 TRAME_POIDS

```
#define TRAME_POIDS QString("3")
```

Définition à la ligne 34 du fichier Communication.h.

Référencé par Communication : :traiterTrame().

9.26 Communication.h

```
00001 #ifndef COMMUNICATION_H
00002 #define COMMUNICATION_H
00003
00013 #include <QObject>
00014 #include <QString>
00015 #include <QSerialPort>
00016
00017 //#define DEBUG_COMMUNICATION
00018
00023 //#define SERIAL_PORT_NAME "/dev/se"
00024 #define SERIAL_PORT_NAME "/dev/ttyUSB0"
00025
00026 // Protocole e-stock
00027 #define EN_TETE             QString("CASIERS")
00028 #define DELIMITEUR_CHAMP    QString(";")
00029 #define DELIMITEUR_FIN      QString("\r\n")
00030
00031 // Types de trames
00032 #define TRAME_OUVERTURE     QString("1")
00033 #define TRAME_ETAT          QString("2")
00034 #define TRAME_POIDS         QString("3")
00035
00048 class Communication : public QObject
00049 {
00050     Q_OBJECT
00051
00052 public:
00053
00054     Communication(QObject *parent = nullptr);
00055     ~Communication();
00056
00057     void demarrerCommunicationPort();
00058     void arreterCommunicationPort();
00059     void configurerPort();
00060     void ouvrirPort();
00061     void setNomPort(QString nouveauPortSerie);
00062     void envoyerTrame(QString trame);
00063     void envoyerRequetePoids(QString numeroCasier = 0);
00064
00065 public slots:
00066     void recevoirTrame();
00067     void envoyerRequeteOuvertureCasier(QString numeroCasier);
00068     void envoyerRequeteEtatCasiers(QString numeroCasier);
00069
00070 private:
00071      QSerialPort *port;
00072      QString trameBrute;
00073      QString nomPort;
00074
00075      bool verifierTrame(QString trame);
00076      void traiterTrame(QString trame);
00077
00078 signals:
00079      void envoieTrameOuverture(QString trame);
00080      void envoieTrameEtat(QString trame);
00081      void envoieTramePoids(QString trame);
00082 };
00083
00084 #endif // COMMUNICATION_H
```

9.27 Référence du fichier Ihm.cpp

Définition de la classe Ihm.

```
#include "Ihm.h"
#include "ui_Ihm.h"
```

```
#include "Supervision.h"
#include "Armoire.h"
#include "Casier.h"
#include "Keyboard.h"
#include <QMessageBox>
#include <QInputDialog>
#include <QDebug>
#include <QString>
```

9.27.1 Description détaillée

Définition de la classe Ihm.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition dans le fichier Ihm.cpp.

9.28 Ihm.cpp

```
00001 #include "Ihm.h"
00002 #include "ui_Ihm.h"
00003 #include "Supervision.h"
00004 #include "Armoire.h"
00005 #include "Casier.h"
00006 #include "Keyboard.h"
00007 #include <QMessageBox>
00008 #include <QInputDialog>
00009 #include <QDebug>
00010 #include <QString>
00011
00031 Ihm::Ihm(QWidget *parent) : QMainWindow(parent), ui(new
      Ui::Ihm), supervision(new Supervision(this))
00032 {
00033     ui->setupUi(this);
00034     // Suppression des parties inutile du QMainWindow
00035     delete ui->menuBar;
00036     delete ui->mainToolBar;
00037     delete ui->statusBar;
00038
00039     // Récupère le clavier virtuelle
00040     keyboard = Keyboard::getInstance(nullptr, this);
00041
00042     // Affiche la fenêtre par défaut en plein écran
00043     allerFenetreBadge();
00044
00045     // Met la fenêtre en plein écran fenêtrer
00046     setWindowFlags(Qt::WindowStaysOnTopHint);
00047     setWindowFlags(Qt::FramelessWindowHint);
00048     // Pour la Raspberry Pi
00049     //showMaximized();
00050
00051     connecterSignauxEtSlots();
00052
00053     supervision->getInformationsArmoire();
00054     supervision->creerCasiers();
00055 }
00056
00061 Ihm::~Ihm()
```

```
00062 {
00063     delete ui;
00064 }
00065
00070 void Ihm::connecterSignauxEtSlots()
00071 {
00072     // Les deux types d'authentifiaction
00073     connect(ui->lineBadge, SIGNAL(editingFinished()), this, SLOT(
      authentifierParBadge()));
00074     connect(ui->pushSeConnecter, SIGNAL(clicked()), this, SLOT(
      authentifierParIdentifiant()));
00075     connect(ui->pushSeDeconnecter, SIGNAL(clicked()), this, SLOT(
      deconnecterUtilisateur()));
00076     connect(ui->lineIdentifiant, SIGNAL(editingFinished()), this, SLOT(
      connecterClavier1()));
00077     connect(ui->lineMotDePasse, SIGNAL(editingFinished()), this, SLOT(
      connecterClavier2()));
00078
00079     //fenêtre scanner un objet
00080     connect(ui->pushScannerObjet, SIGNAL(clicked()), this, SLOT(
      allerFenetreScannerObjet()));
00081     connect(ui->pushScannerSeDeconnecter, SIGNAL(clicked()), this, SLOT(
      deconnecterUtilisateur()));
00082     connect(ui->pushScannerPageStock, SIGNAL(clicked()), this, SLOT(
      allerFenetreMenu()));
00083
00084     // Les deux fenêtres d'authentifiaction
00085     connect(ui->pushParIdentifiant, SIGNAL(clicked()), this, SLOT(
      allerFenetreIdentifiant()));
00086     connect(ui->pushParBadge, SIGNAL(clicked()), this, SLOT(allerFenetreBadge()));
00087
00088     // Article
00089     connect(ui->lineRecherche, SIGNAL(textChanged(QString)), this, SLOT(
      activerRecherche()));
00090     connect(ui->pushRecherche, SIGNAL(clicked(bool)), this, SLOT(
      rechercherArticle()));
00091
00092     // CodeBarre
00093     connect(ui->pushScannerPrendre, SIGNAL(clicked()), this, SLOT(
      appuyerBoutonPrendre()));
00094     connect(ui->pushScannerPrendre, SIGNAL(clicked()), this, SLOT(
      changerTextePrendre()));
00095     connect(ui->pushScannerAjouter, SIGNAL(clicked()), this, SLOT(
      appuyerBoutonAjouter()));
00096     connect(ui->pushScannerAjouter, SIGNAL(clicked()), this, SLOT(
      changerTexteAjouter()));
00097     connect(ui->spinBoxQuantiteScan, SIGNAL(valueChanged(int)), this, SLOT(
      afficherDemandeQuantite(int)));
00098     connect(ui->lineEditScanner, SIGNAL(editingFinished()), this, SLOT(
      scannerObjet()));
00099 }
00100
00106 void Ihm::changerDeFenetre(int fenetre)
00107 {
00108     ui->stackedWidget->setCurrentIndex(fenetre);
00109 }
00110
00116 void Ihm::placerCasier(Casier *casier)
00117 {
00118     // pair/impair -> droite/gauche ?
00119     int numero = casier->getNumero() - 1;
00120     if((numero+1)%2)
00121     {
00122         ui->gridLayoutCasiers->addWidget(casier, numero/2, 0, 1, 1);
00123         ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 0, 1, 1);
00124     }
00125     else
00126     {
00127         ui->gridLayoutCasiers->addWidget(casier, numero/2, 1, 1, 1);
00128         ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 1, 1, 1);
00129     }
00130 }
00131
00132 void Ihm::placerCasiers(const QVector<Casier*> &casiers, int fenetre)
00133 {
00134     for(int i=0; i < casiers.size(); i++)
00135     {
00136         Casier* casier = casiers[i];
00137
00138         // pair/impair -> droite/gauche ?
00139         int numero = casier->getNumero() - 1;
00140         if((numero+1)%2)
00141         {
00142             switch (fenetre)
00143             {
00144             case FENETRE_MENU:
00145                 ui->gridLayoutCasiers->addWidget(casier, numero/2, 0, 1, 1);
00146                 break;
00147             case FENETRE_SCANNER_OBJET:
00148                 ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 0, 1, 1);
00149                 break;
```

```
00150             default:
00151                 break;
00152             }
00153         }
00154         else
00155         {
00156             switch (fenetre)
00157             {
00158             case FENETRE_MENU:
00159                 ui->gridLayoutCasiers->addWidget(casier, numero/2, 1, 1, 1);
00160                 break;
00161             case FENETRE_SCANNER_OBJET:
00162                 ui->gridLayoutCasiersScan->addWidget(casier, numero/2, 1, 1, 1);
00163                 break;
00164             default:
00165                 break;
00166             }
00167         }
00168     }
00169 }
00170
00176 void Ihm::afficherInformationsArmoire(QStringList informationsArmoire)
00177 {
00178     #ifdef DEBUG_IHM
00179         qDebug() << Q_FUNC_INFO << "informationsArmoire" << informationsArmoire;
00180     #endif
00181     ui->labelNomArmoire->setText(informationsArmoire.at(TABLE_ARMOIRE_NOM) + " " +
      informationsArmoire.at(TABLE_ARMOIRE_DESCRIPTION) + " (" + informationsArmoire.at(
      TABLE_ARMOIRE_NB_CASIERS+1) + ")");
00182     ui->labelNbCasiers->setText(informationsArmoire.at(
      TABLE_ARMOIRE_NB_CASIERS));
00183 }
00184
00189 void Ihm::authentifierParBadge()
00190 {
00191     ui->labelErreurBadge->clear();
00192
00193     if(ui->lineBadge->text() != "")
00194     {
00195         #ifdef DEBUG_IHM
00196             qDebug() << Q_FUNC_INFO << "Contenu brut badge" << ui->lineBadge->text();
00197         #endif
00198
00199         QString trameBadge = ui->lineBadge->text();
00200         ui->lineBadge->clear();
00201         emit badgeDetecte(trameBadge);
00202     }
00203 }
00204
00209 void Ihm::authentifierParIdentifiant()
00210 {
00211     if(ui->lineIdentifiant->text() != "")
00212     {
00213         #ifdef DEBUG_IHM
00214             qDebug() << Q_FUNC_INFO << "Identifiant" << ui->lineIdentifiant->text() << "MotDePasse" <<
      ui->lineMotDePasse->text();
00215         #endif
00216
00217         QString identifiant = ui->lineIdentifiant->text();
00218         QString motDePasse = ui->lineMotDePasse->text();
00219         ui->lineIdentifiant->clear();
00220         ui->lineMotDePasse->clear();
00221         emit identifiantDetecte(identifiant, motDePasse);
00222     }
00223 }
00224
00229 void Ihm::deconnecterUtilisateur()
00230 {
00231     supervision->deconnecterUtilisateur();
00232     changerDeFenetre(FENETRE_BADGE);
00233 }
00234
00239 void Ihm::allerFenetreBadge()
00240 {
00241     changerDeFenetre(FENETRE_BADGE);
00242     ui->lineBadge->setFocus();
00243 }
00244
00250 void Ihm::allerFenetreIdentifiant()
00251 {
00252     ui->lineMotDePasse->setEchoMode(QLineEdit::Password);
00253     keyboard->setLineEdit(ui->lineIdentifiant);
00254     changerDeFenetre(FENETRE_IDENTIFIANT);
00255     ui->lineIdentifiant->setFocus();
00256 }
00257
00262 void Ihm::allerFenetreMenu()
00263 {
00264     changerDeFenetre(FENETRE_MENU);
00265     // Initialisation widgets
00266     placerCasiers(supervision->getCasiers(),
```

```
      FENETRE_MENU);
00267     keyboard->setLineEdit(ui->lineRecherche);
00268     ui->pushRecherche->setFocus();
00269     ui->comboBoxArticle->clear();
00270     ui->comboBoxArticle->addItem("Sélectionner un article");
00271     ui->pushRecherche->setEnabled(false);
00272     ui->lineRecherche->setFocus();
00273     // Lance une recherche de tous les articles
00274     emit rechercheArticle("");
00275 }
00276
00281 void Ihm::allerFenetreScannerObjet()
00282 {
00283     placerCasiers(supervision->getCasiers(),
      FENETRE_SCANNER_OBJET);
00284     ui->pushScannerPrendre->setEnabled(false);
00285     ui->pushScannerAjouter->setEnabled(false);
00286     ui->spinBoxQuantiteScan->setValue(0);
00287     ui->spinBoxQuantiteScan->setEnabled(false);
00288     changerDeFenetre(FENETRE_SCANNER_OBJET);
00289 }
00290
00296 void Ihm::afficherErreurBadge(QString message)
00297 {
00298     ui->labelErreurBadge->setText(message);
00299 }
00300
00305 void Ihm::afficherErreurDepassementQuantite()
00306 {
00307     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_DEPASSEMENT_QUANTITE));
00308 }
00309
00316 void Ihm::traiterDemandeDeConnexion(bool reponse, QString message)
00317 {
00318     if(reponse)
00319     {
00320         allerFenetreMenu();
00321     }
00322     else
00323     {
00324         QMessageBox::critical(nullptr, APPLICATION, message);
00325     }
00326 }
00327
00332 void Ihm::activerRecherche()
00333 {
00334     if(!ui->lineRecherche->text().isEmpty())
00335         ui->pushRecherche->setEnabled(true);
00336     else
00337         ui->pushRecherche->setEnabled(false);
00338 }
00339
00344 void Ihm::rechercherArticle()
00345 {
00346     if(!ui->lineRecherche->text().isEmpty())
00347         supervision->rechercherArticle(ui->lineRecherche->text());
00348 }
00349
00354 void Ihm::effacerRechercheArticle()
00355 {
00356     ui->lineRecherche->setText("");
00357 }
00358
00364 void Ihm::mettreAJourListeArticles(QVector<QStringList> articlesTrouves)
00365 {
00366     #ifdef DEBUG_IHM
00367         qDebug() << Q_FUNC_INFO << "articlesTrouves" << articlesTrouves.size() << articlesTrouves;
00368     #endif
00369     creerListeArticles(articlesTrouves);
00370
00371     effacerRechercheArticle();
00372 }
00373
00379 void Ihm::selectionnerArticle(int index)
00380 {
00381     #ifdef DEBUG_IHM
00382         qDebug() << Q_FUNC_INFO << "index" << index << ui->comboBoxArticle->currentText();
00383     #endif
00384
00385     supervision->selectionnerArticle(ui->comboBoxArticle->currentText());
00386 }
00387
00393 void Ihm::afficherDonneesArticleSelectionne(QStringList
      donneesArticle)
00394 {
00395     ui->labelCasier->setText("Casier :");
00396     ui->labelQuantiteNombre->setText(donneesArticle.at(ARTICLE_QUANTITE));
00397     ui->labelDisponibleNombre->setText(donneesArticle.at(ARTICLE_DISPONIBLE));
00398     ui->labelCasierNombre->setText(donneesArticle.at(NUMERO_CASIERS));
00399 }
```

```
00400
00406 void Ihm::afficherDonneesArticleSelectionne(QVector<QStringList>
      donneesArticle)
00407 {
00408     if(donneesArticle.size() <= 0)
00409         return;
00410     unsigned int articleQuantite = 0;
00411     unsigned int articleDisponible = 0;
00412     QString casiersQuantite;
00413     QString casiersDisponible;
00414     QString casiers;
00415     int nombreCasiers = donneesArticle.size();
00416
00417     for(int i = 0; i < nombreCasiers; i++)
00418     {
00419         #ifdef DEBUG_IHM
00420             qDebug() << Q_FUNC_INFO << "disponible" << (donneesArticle[i].at(
      ARTICLE_DISPONIBLE)).toUInt();
00421             qDebug() << Q_FUNC_INFO << "articleDisponible" << articleDisponible;
00422             qDebug() << Q_FUNC_INFO << "quantite" << (donneesArticle[i].at(
      ARTICLE_QUANTITE)).toUInt();
00423             qDebug() << Q_FUNC_INFO << "articleQuantite" << articleQuantite;
00424         #endif
00425         articleDisponible += (donneesArticle[i].at(ARTICLE_DISPONIBLE)).toUInt();
00426         articleQuantite += (donneesArticle[i].at(ARTICLE_QUANTITE)).toUInt();
00427
00428         if(i == 0)
00429         {
00430             casiers = donneesArticle[i].at(NUMERO_CASIERS);
00431             casiersDisponible = QString(" (") + donneesArticle[i].at(
      ARTICLE_DISPONIBLE);
00432             casiersQuantite = QString(" (") + donneesArticle[i].at(
      ARTICLE_QUANTITE);
00433         }
00434         else
00435         {
00436             casiers += " et " + donneesArticle[i].at(NUMERO_CASIERS);
00437             casiersDisponible += QString(" et ") + donneesArticle[i].at(
      ARTICLE_DISPONIBLE);
00438             casiersQuantite += QString(" et ") + donneesArticle[i].at(
      ARTICLE_QUANTITE);
00439         }
00440     }
00441     casiersDisponible += QString(")");
00442     casiersQuantite += QString(")");
00443
00444     ui->labelCasier->setText("Casiers :");
00445     ui->labelQuantiteNombre->setText(QString::number(articleQuantite) + casiersQuantite);
00446     ui->labelDisponibleNombre->setText(QString::number(articleDisponible) + casiersDisponible);
00447     ui->labelCasierNombre->setText(casiers);
00448 }
00449
00455 void Ihm::creerListeArticles(const QVector<QStringList> &articlesTrouves)
00456 {
00457     disconnect(ui->comboBoxArticle, SIGNAL(currentIndexChanged(int)), this, SLOT(
      selectionnerArticle(int)));
00458     ui->comboBoxArticle->clear();
00459
00460     ui->comboBoxArticle->addItem("Sélectionner un article");
00461     for(int i = 0 ; i < articlesTrouves.size() ; i++)
00462     {
00463         if(ui->comboBoxArticle->findText(articlesTrouves[i].at(2)) == -1)
00464         {
00465             ui->comboBoxArticle->addItem(articlesTrouves[i].at(2));
00466         }
00467     }
00468     connect(ui->comboBoxArticle, SIGNAL(currentIndexChanged(int)), this, SLOT(
      selectionnerArticle(int)));
00469 }
00470
00475 void Ihm::appuyerBoutonPrendre()
00476 {
00477     ui->labelScannerObjet->setText("Vous pouvez indiquer une quantité");
00478     ui->lineEditScanner->setFocus();
00479     emit boutonPrendre(false);
00480 }
00481
00486 void Ihm::appuyerBoutonAjouter()
00487 {
00488     ui->labelScannerObjet->setText("Vous pouvez indiquer une quantité");
00489     ui->lineEditScanner->setFocus();
00490     emit boutonAjouter(true);
00491 }
00492
00497 void Ihm::changerTextePrendre()
00498 {
00499     ui->labelScannerObjet->setText("Veuillez scanner l'objet à prendre");
00500 }
00501
00506 void Ihm::changerTexteAjouter()
00507 {
```

```
00508     ui->labelScannerObjet->setText("Veuillez scanner l'objet à ajouter");
00509 }
00510
00515 void Ihm::scannerObjet()
00516 {
00517     QString codeBarreObjet = ui->lineEditScanner->text();
00518     qDebug() << Q_FUNC_INFO << "codeBarreObjet" << codeBarreObjet;
00519
00520     if(!ui->lineEditScanner->text().isEmpty())
00521     {
00522         emit codeBarreObjetScanner(codeBarreObjet);
00523     }
00524
00525     ui->lineEditScanner->setText("");
00526 }
00527
00533 void Ihm::afficherDemandeQuantite(int quantite)
00534 {
00535     emit envoyerQuantite(quantite);
00536     ui->lineEditScanner->setFocus();
00537 }
00538
00543 void Ihm::afficherErreurArticleInsuffisants()
00544 {
00545     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_QUANTITE_INSUFFISANTE));
00546 }
00547
00552 void Ihm::afficherErreurAucunCasierOuvert()
00553 {
00554     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_AUCUN_CASIER_OUVERT));
00555 }
00556
00561 void Ihm::afficherErreurPasArticleAvecCodeBarre()
00562 {
00563     QMessageBox::critical(nullptr, APPLICATION, QString::fromUtf8(
      MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE));
00564 }
00565
00572 void Ihm::gererPageScanObjet(int numeroCasier, bool etat)
00573 {
00574     if(etat)
00575     {
00576         ui->labelScannerObjet->setText("Veuillez prendre ou ajouter un objet");
00577         ui->pushScannerPrendre->setEnabled(true);
00578         ui->pushScannerAjouter->setEnabled(true);
00579         ui->spinBoxQuantiteScan->setEnabled(true);
00580         ui->spinBoxQuantiteScan->setValue(1); // par défaut
00581         ui->lineEditScanner->setEnabled(true);
00582         ui->lineEditScanner->setFocus();
00583     }
00584     else
00585     {
00586         ui->labelScannerObjet->setText("Veuillez ouvrir un casier");
00587         ui->pushScannerPrendre->setEnabled(false);
00588         ui->pushScannerAjouter->setEnabled(false);
00589         ui->spinBoxQuantiteScan->setValue(0);
00590         ui->spinBoxQuantiteScan->setEnabled(false);
00591         ui->lineEditScanner->setEnabled(false);
00592     }
00593 }
00594
00599 void Ihm::connecterClavier1()
00600 {
00601     keyboard->setLineEdit(ui->lineMotDePasse);
00602 }
00603
00608 void Ihm::connecterClavier2()
00609 {
00610     keyboard->setLineEdit(ui->lineIdentifiant);
00611 }
```

9.29 Référence du fichier Ihm.h

Déclaration de la classe Ihm.

```
#include <QtWidgets>
```

Classes

— class Ihm
Déclaration de la classe Ihm.

Espaces de nommage

— Ui

Macros

— #define APPLICATION "e-stock"
— #define ARTICLE_DISPONIBLE 1
— #define ARTICLE_QUANTITE 0
— #define DEBUG_IHM
— #define MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE "Erreur, aucun article ne correspond à ce code barre dans le casier"
— #define MESSAGE_ERREUR_AUCUN_CASIER_OUVERT "Erreur, vous devez d'avord ouvrir le casier dans lequelle vous souhaiteé effectuer vos actions !"
— #define MESSAGE_ERREUR_DEPASSEMENT_QUANTITE "Opération impossible, le nombre d'articles maximum serait dépassé"
— #define MESSAGE_ERREUR_QUANTITE "Erreur, quantite non valide"
— #define MESSAGE_ERREUR_QUANTITE_INSUFFISANTE "Erreur, pas assez d'articles disponible"
— #define MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE "Erreur : le compte n'est plaus valide !"
— #define MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE "Erreur : utilisateur non valide !"
— #define NUMERO_CASIERS 2

Énumérations

— enum FenetresIHM { FENETRE_BADGE, FENETRE_IDENTIFIANT, FENETRE_MENU, FENETRE_SCANNER_OBJET }
Définit les différents types de fenêtres de l'application.

9.29.1 Description détaillée

Déclaration de la classe Ihm.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition dans le fichier Ihm.h.

9.29.2 Documentation des macros

9.29.2.1 APPLICATION

```
#define APPLICATION "e-stock"
```

Définition à la ligne 32 du fichier Ihm.h.

Référencé par Ihm : :afficherErreurArticleInsuffisants(), Ihm : :afficherErreurAucunCasierOuvert(), Ihm : :afficherErreurDepassement↩Quantite(), Ihm : :afficherErreurPasArticleAvecCodeBarre(), et Ihm : :traiterDemandeDeConnexion().

9.29.2.2 ARTICLE_DISPONIBLE

```
#define ARTICLE_DISPONIBLE 1
```

Définition à la ligne 42 du fichier Ihm.h.

Référencé par Ihm : :afficherDonneesArticleSelectionne().

9.29.2.3 ARTICLE_QUANTITE

```
#define ARTICLE_QUANTITE 0
```

Définition à la ligne 41 du fichier Ihm.h.

Référencé par Ihm : :afficherDonneesArticleSelectionne().

9.29.2.4 DEBUG_IHM

```
#define DEBUG_IHM
```

Définition à la ligne 4 du fichier Ihm.h.

9.29.2.5 MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE

```
#define MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE "Erreur, aucun article ne correspond à ce code barre dans le
casier"
```

Définition à la ligne 36 du fichier Ihm.h.

Référencé par Ihm : :afficherErreurPasArticleAvecCodeBarre().

9.29.2.6 MESSAGE_ERREUR_AUCUN_CASIER_OUVERT

```
#define MESSAGE_ERREUR_AUCUN_CASIER_OUVERT "Erreur, vous devez d'avord ouvrir le casier dans lequelle vous
souhaiteé effectuer vos actions!"
```

Définition à la ligne 39 du fichier Ihm.h.

Référencé par Ihm : :afficherErreurAucunCasierOuvert().

9.29.2.7 MESSAGE_ERREUR_DEPASSEMENT_QUANTITE

```
#define MESSAGE_ERREUR_DEPASSEMENT_QUANTITE "Opération impossible, le nombre d'articles maximum serait dépassé"
```

Définition à la ligne 35 du fichier Ihm.h.

Référencé par Ihm : :afficherErreurDepassementQuantite().

9.29.2.8 MESSAGE_ERREUR_QUANTITE

```
#define MESSAGE_ERREUR_QUANTITE "Erreur, quantite non valide"
```

Définition à la ligne 37 du fichier Ihm.h.

9.29.2.9 MESSAGE_ERREUR_QUANTITE_INSUFFISANTE

```
#define MESSAGE_ERREUR_QUANTITE_INSUFFISANTE "Erreur, pas assez d'articles disponible"
```

Définition à la ligne 38 du fichier Ihm.h.

Référencé par Ihm : :afficherErreurArticleInsuffisants().

9.29.2.10 MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE

```
#define MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE "Erreur : le compte n'est plaus valide !"
```

Définition à la ligne 34 du fichier Ihm.h.

Référencé par Supervision : :verifierDonneesUtilisateur().

9.29.2.11 MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE

```
#define MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE "Erreur : utilisateur non valide !"
```

Définition à la ligne 33 du fichier Ihm.h.

Référencé par Supervision : :verifierDonneesUtilisateur().

9.29.2.12 NUMERO_CASIERS

```
#define NUMERO_CASIERS 2
```

Définition à la ligne 43 du fichier Ihm.h.

Référencé par Ihm : :afficherDonneesArticleSelectionne().

9.29.3 Documentation du type de l'énumération

9.29.3.1 FenetresIHM

```
enum FenetresIHM
```

Définit les différents types de fenêtres de l'application.

Valeurs énumérées

FENETRE_BADGE	Fenêtre d'authentifiaction par badge.
FENETRE_IDENTIFIANT	Fenêtre d'authentifiaction par identifiant.
FENETRE_MENU	Fenêtre ? ? ?
FENETRE_SCANNER_OBJET	Fenêtre pour scan d'un objet.

Définition à la ligne 24 du fichier Ihm.h.

```
00025 {
00026     FENETRE_BADGE,
00027     FENETRE_IDENTIFIANT,
00028     FENETRE_MENU,
00029     FENETRE_SCANNER_OBJET
00030 };
```

9.30 Ihm.h

```
00001 #ifndef IHM_H
00002 #define IHM_H
00003
00004 #define DEBUG_IHM
00005
00024 enum FenetresIHM
00025 {
00026     FENETRE_BADGE,
00027     FENETRE_IDENTIFIANT,
00028     FENETRE_MENU,
00029     FENETRE_SCANNER_OBJET
00030 };
00031
00032 #define APPLICATION "e-stock"
00033 #define MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE "Erreur : utilisateur non valide !"
00034 #define MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE "Erreur : le compte n'est plaus valide !"
00035 #define MESSAGE_ERREUR_DEPASSEMENT_QUANTITE "Opération impossible, le nombre d'articles maximum serait
       dépassé"
00036 #define MESSAGE_ERREUR_AUCUN_ARTICLE_CODE_BARRE "Erreur, aucun article ne correspond à ce code barre dans
       le casier"
00037 #define MESSAGE_ERREUR_QUANTITE "Erreur, quantite non valide"
00038 #define MESSAGE_ERREUR_QUANTITE_INSUFFISANTE "Erreur, pas assez d'articles disponible"
00039 #define MESSAGE_ERREUR_AUCUN_CASIER_OUVERT "Erreur, vous devez d'avord ouvrir le casier dans lequelle vous
       souhaiteé effectuer vos actions!"
00040
00041 #define ARTICLE_QUANTITE 0
00042 #define ARTICLE_DISPONIBLE 1
00043 #define NUMERO_CASIERS 2
00044
00045 #include <QtWidgets>
00046
00047 class Supervision;
00048 class Casier;
00049 class Keyboard;
00050
00051 namespace Ui {
00052 class Ihm;
00053 }
00054
00068 class Ihm : public QMainWindow
00069 {
00070     Q_OBJECT
00071
00072 public:
00073     explicit Ihm(QWidget *parent = nullptr);
00074     ~Ihm();
00075
00076     void changerDeFenetre(int fenetre);
00077     void placerCasier(Casier *casier);
00078     void placerCasiers(const QVector<Casier*> &casiers, int fenetre);
00079
00080 private slots:
00081     void afficherInformationsArmoire(QStringList informationsArmoire);
00082     void authentifierParBadge();
00083     void authentifierParIdentifiant();
00084     void deconnecterUtilisateur();
00085     void allerFenetreBadge();
00086     void allerFenetreIdentifiant();
00087     void allerFenetreMenu();
00088     void allerFenetreScannerObjet();
```

```
00089     void afficherErreurBadge(QString message);
00090     void afficherErreurDepassementQuantite();
00091     void traiterDemandeDeConnexion(bool reponse, QString message);
00092     void activerRecherche();
00093     void rechercherArticle();
00094     void effacerRechercheArticle();
00095     void mettreAJourListeArticles(QVector<QStringList> articlesTrouves);
00096     void selectionnerArticle(int index);
00097     void afficherDonneesArticleSelectionne(QStringList donneesArticle);
00098     void afficherDonneesArticleSelectionne(QVector<QStringList> donneesArticle);
00099     void appuyerBoutonPrendre();
00100     void appuyerBoutonAjouter();
00101     void changerTextePrendre();
00102     void changerTexteAjouter();
00103     void scannerObjet();
00104     void afficherDemandeQuantite(int);
00105     void afficherErreurArticleInsuffisants();
00106     void afficherErreurAucunCasierOuvert();
00107     void afficherErreurPasArticleAvecCodeBarre();
00108     void gererPageScanObjet(int numeroCasier, bool etat);
00109     void connecterClavier1();
00110     void connecterClavier2();
00111
00112 signals:
00113     void badgeDetecte(QString);
00114     void identifiantDetecte(QString identifiant, QString motDePasse);
00115     void rechercheArticle(QString);
00116     void articleSelectionne(QString);
00117     void boutonPrendre(bool);
00118     void boutonAjouter(bool);
00119     void codeBarreObjetScanner(QString);
00120     void envoyerQuantite(int);
00121
00122 private:
00123     Ui::Ihm *ui;
00124     Supervision *supervision;
00125     Keyboard *keyboard;
00126
00127     void creerListeArticles(const QVector<QStringList> &articlesTrouves);
00128     void connecterSignauxEtSlots();
00129 };
00130
00131 #endif // IHM_H
```

9.31 Référence du fichier Keyboard.cpp

Fichier qui contient la définition de la classe Keyboard.

```
#include "Keyboard.h"
#include <QGridLayout>
#include <QSignalMapper>
#include <QPushButton>
#include <QDebug>
```

Classes

— struct KeyboardLayoutEntry

Macros

— #define NEXT_ROW_MARKER 0

Variables

— KeyboardLayoutEntry keyboardLayout [ ]
— static const int layoutSize = (sizeof(keyboardLayout) / sizeof(KeyboardLayoutEntry))

9.31.1 Description détaillée

Fichier qui contient la définition de la classe Keyboard.

Auteur

Thierry Vaira

based on The virtual keyboard demo software is Copyright (C) 2015 Klaralvdalens Datakonsult AB.

Définition dans le fichier Keyboard.cpp.

9.31.2 Documentation des macros

9.31.2.1 NEXT_ROW_MARKER

```
#define NEXT_ROW_MARKER 0
```

Définition à la ligne 16 du fichier Keyboard.cpp.

Référencé par Keyboard : :init(), et Keyboard : :makeCapsLock().

9.31.3 Documentation des variables

9.31.3.1 keyboardLayout

```
KeyboardLayoutEntry keyboardLayout[]
```

Définition à la ligne 26 du fichier Keyboard.cpp.

9.31.3.2 layoutSize

```
const int layoutSize = (sizeof(keyboardLayout) / sizeof(KeyboardLayoutEntry))  [static]
```

Définition à la ligne 85 du fichier Keyboard.cpp.

Référencé par Keyboard : :init(), Keyboard : :keyToCharacter(), et Keyboard : :makeCapsLock().

9.32 Keyboard.cpp

```
00001
00009 #include "Keyboard.h"
00010
00011 #include <QGridLayout>
00012 #include <QSignalMapper>
00013 #include <QPushButton>
00014 #include <QDebug>
00015
00016 #define NEXT_ROW_MARKER 0
00017
00018 struct KeyboardLayoutEntry
00019 {
00020     int key;
00021     const char *label;
00022     int keyCapsLock;
00023     const char *labelCapsLock;
00024 };
00025
00026 KeyboardLayoutEntry keyboardLayout[] =
00027 {
00028     { Qt::Key_1, "1", Qt::Key_Ampersand, "&" },
00029     { Qt::Key_2, "2", Qt::Key_Eacute, "é" },
00030     { Qt::Key_3, "3", Qt::Key_QuoteDbl, "\"" },
00031     { Qt::Key_4, "4", Qt::Key_Apostrophe, "'" },
00032     { Qt::Key_5, "5", Qt::Key_ParenRight, "(" },
00033     { Qt::Key_6, "6", Qt::Key_hyphen, "-" },
00034     { Qt::Key_7, "7", Qt::Key_Egrave, "è" },
00035     { Qt::Key_8, "8", Qt::Key_Underscore, "_" },
00036     { Qt::Key_9, "9", Qt::Key_cedilla, "ç" },
00037     { Qt::Key_0, "0", Qt::Key_Agrave, "à" },
00038     { Qt::Key_Percent, "%", Qt::Key_ParenLeft, ")" },
00039     { Qt::Key_Plus, "+", Qt::Key_Equal, "=" },
00040     { Qt::Key_Backspace, "<-",  Qt::Key_Backspace, "<-"},
00041     { NEXT_ROW_MARKER, 0, 0, 0 },
00042     { Qt::Key_A, "a", Qt::Key_A, "A" },
00043     { Qt::Key_Z, "z", Qt::Key_Z, "Z" },
00044     { Qt::Key_E, "e", Qt::Key_E, "E" },
00045     { Qt::Key_R, "r", Qt::Key_R, "R" },
00046     { Qt::Key_T, "t", Qt::Key_T, "T" },
00047     { Qt::Key_Y, "y", Qt::Key_Y, "Y" },
00048     { Qt::Key_U, "u", Qt::Key_U, "U" },
00049     { Qt::Key_I, "i", Qt::Key_I, "I" },
00050     { Qt::Key_O, "o", Qt::Key_O, "O" },
00051     { Qt::Key_P, "p", Qt::Key_P, "P" },
00052     { Qt::Key_At, "@", Qt::Key_Dollar, "$" },
00053     { Qt::Key_Escape, "Esc", Qt::Key_Escape, "Esc" },
00054     { Qt::Key_Delete, "X",  Qt::Key_Delete, "X"},
00055     { NEXT_ROW_MARKER, 0, 0, 0 },
00056     { Qt::Key_Q, "q", Qt::Key_Q, "Q" },
00057     { Qt::Key_S, "s", Qt::Key_S, "S" },
00058     { Qt::Key_D, "d", Qt::Key_D, "D" },
00059     { Qt::Key_F, "f", Qt::Key_F, "F" },
00060     { Qt::Key_G, "g", Qt::Key_G, "G" },
00061     { Qt::Key_H, "h", Qt::Key_H, "H" },
00062     { Qt::Key_J, "j", Qt::Key_J, "J" },
00063     { Qt::Key_K, "k", Qt::Key_K, "K" },
00064     { Qt::Key_L, "l", Qt::Key_L, "L" },
00065     { Qt::Key_M, "m", Qt::Key_M, "M" },
00066     { Qt::Key_Ooblique, "/", Qt::Key_Backslash, "\\" },
00067     { Qt::Key_multiply, "*", Qt::Key_ssharp, "#" },
00068     { Qt::Key_Space, " ", Qt::Key_Space, " " },
00069     { NEXT_ROW_MARKER, 0, 0, 0 },
00070     { Qt::Key_CapsLock, "Maj", Qt::Key_CapsLock, "Maj" },
00071     { Qt::Key_Less, "<", Qt::Key_Greater, ">" },
00072     { Qt::Key_W, "w", Qt::Key_W, "W"},
00073     { Qt::Key_X, "x", Qt::Key_X, "X"},
00074     { Qt::Key_C, "c", Qt::Key_C, "C"},
00075     { Qt::Key_V, "v", Qt::Key_V, "V"},
00076     { Qt::Key_B, "b", Qt::Key_B, "B"},
00077     { Qt::Key_N, "n", Qt::Key_N, "N"},
00078     { Qt::Key_Comma, ",", Qt::Key_Question, "?"},
00079     { Qt::Key_Semicolon, ";", Qt::Key_Period, "."},
00080     { Qt::Key_Colon, ":", Qt::Key_AsciiTilde, "~"},
00081     { Qt::Key_exclamdown, "!", Qt::Key_section, "§"},
00082     { Qt::Key_Enter, "Entrée", Qt::Key_Enter, "Entrée"}
00083 };
00084
00085 const static int layoutSize = (sizeof(keyboardLayout) / sizeof(
      KeyboardLayoutEntry));
00086
00087 #ifdef KEYBOARD_SINGLETON
00088 Keyboard* Keyboard::instance = Q_NULLPTR;
00089
00090 Keyboard* Keyboard::getInstance(QLineEdit *lineEdit,
      QWidget *parent)
00091 {
00092     if(instance == Q_NULLPTR)
00093     {
```

```
00094         instance = new Keyboard(lineEdit, parent);
00095     }
00096     return instance;
00097 }
00098 #endif
00099
00100 Keyboard::Keyboard(QLineEdit *lineEdit, QWidget *parent) :
      QWidget(parent), lineEdit(lineEdit), capsLock(false), mask(false)
00101 {
00102 #ifdef DEBUG_KEYBOARD
00103     qDebug() << Q_FUNC_INFO << this << lineEdit;
00104 #endif
00105     init();
00106     connect(qApp, SIGNAL(focusChanged(QWidget*,QWidget*)), this, SLOT(
      focusChange(QWidget*,QWidget*)));
00107     connect(this, SIGNAL(keyClicked(QString)), this, SLOT(
      keyboardKeyClicked(QString)));
00108     connect(this, SIGNAL(specialKeyClicked(int)), this, SLOT(
      keyboardSpecialKeyClicked(int)));
00109     setStyleSheet("/*QWidget{background-color:white;}*/QPushButton{font-family:\"Ubuntu Mono\"
      ;font:bold;fon
      t-size:16px;background-color:palegoldenrod;border-width:1px;border-color:darkkhaki;border-style:solid;border
      -radius:5;padding:1px;}QPushButton:hover{background-color:khaki;}QPushButton:pressed{background-color:#d0d67c;}");
00110     setAttribute(Qt::WA_TranslucentBackground, true);
00111 }
00112
00113 void Keyboard::paintEvent(QPaintEvent *e)
00114 {
00115     QPainter painter(this);
00116     QPen pen;
00117     pen.setBrush(QBrush(QColor(128, 128, 128, 64)));
00118     painter.setPen(pen);
00119     painter.drawRoundedRect(0,0,width()-1, height()-1,5,5);
00120     QWidget::paintEvent(e);
00121 }
00122
00123 void Keyboard::setLineEdit(QLineEdit *lineEdit)
00124 {
00125     if(lineEdit && this->lineEdit != lineEdit)
00126     {
00127 #ifdef DEBUG_KEYBOARD
00128     qDebug() << Q_FUNC_INFO << this << lineEdit;
00129 #endif
00130         this->lineEdit = lineEdit;
00131         capsLock = false;
00132         mask = false;
00133     }
00134 }
00135
00136 void Keyboard::showKeyboard(int globalX, int globalY)
00137 {
00138     QWidget::move(globalX, globalY);
00139     if(mask)
00140     {
00141         setFixedSize(hBoxLayout->sizeHint());
00142         stackedWidget->setCurrentIndex(1);
00143     }
00144     else
00145     {
00146         setFixedSize(gridLayout->sizeHint());
00147         stackedWidget->setCurrentIndex(0);
00148     }
00149     QWidget::show();
00150 }
00151
00152 void Keyboard::hideKeyboard()
00153 {
00154     QWidget::hide();
00155 }
00156
00157 bool Keyboard::keyboardVisible() const
00158 {
00159     return QWidget::isVisible();
00160 }
00161
00162 void Keyboard::buttonClicked(int key)
00163 {
00164 #ifdef DEBUG_KEYBOARD
00165     qDebug() << Q_FUNC_INFO << key;
00166 #endif
00167     if ((key == Qt::Key_Escape))
00168     {
00169         mask = !mask;
00170         if(mask)
00171         {
00172             setFixedSize(hBoxLayout->sizeHint());
00173             stackedWidget->setCurrentIndex(1);
00174         }
00175         else
00176         {
00177             setFixedSize(gridLayout->sizeHint());
```

```
00178             stackedWidget->setCurrentIndex(0);
00179         }
00180         return;
00181     }
00182     if ((key == Qt::Key_CapsLock))
00183     {
00184         capsLock = !capsLock;
00185         makeCapsLock();
00186         return;
00187     }
00188     if ((key == Qt::Key_Delete))
00189     {
00190         if(lineEdit)
00191             lineEdit->clear();
00192         return;
00193     }
00194     if ((key == Qt::Key_Enter) || (key == Qt::Key_Backspace))
00195     {
00196         emit specialKeyClicked(key);
00197     }
00198     else
00199     {
00200         emit keyClicked(keyToCharacter(key));
00201     }
00202 }
00203
00204 void Keyboard::focusChange(QWidget* oldWidget,
      QWidget* newWidget)
00205 {
00206     if(!lineEdit || !newWidget)
00207         return;
00208 #ifdef DEBUG_KEYBOARD
00209     qDebug() << Q_FUNC_INFO << oldWidget << newWidget;
00210 #endif
00211     if(newWidget == lineEdit)
00212     {
00213         QPoint globalPos(0, 0);
00214         globalPos = newWidget->mapToGlobal(QPoint(0, newWidget->height()));
00215         showKeyboard(globalPos.x(), globalPos.y());
00216     }
00217     else
00218     {
00219         hideKeyboard();
00220     }
00221 }
00222
00223 void Keyboard::keyboardSpecialKeyClicked(int key)
00224 {
00225 #ifdef DEBUG_KEYBOARD
00226     qDebug() << Q_FUNC_INFO << this << key;
00227 #endif
00228     if (!lineEdit)
00229         return;
00230
00231     if (key == Qt::Key_Enter)
00232     {
00233         QKeyEvent *pressEvent = new QKeyEvent(QEvent::KeyPress, Qt::Key_Enter, Qt::NoModifier);
00234         QGuiApplication::postEvent(lineEdit, pressEvent);
00235
00236         QKeyEvent *releaseEvent = new QKeyEvent(QEvent::KeyRelease, Qt::Key_Enter, Qt::NoModifier);
00237         QGuiApplication::postEvent(lineEdit, releaseEvent);
00238     }
00239     else if (key == Qt::Key_Backspace)
00240     {
00241         QKeyEvent *pressEvent = new QKeyEvent(QEvent::KeyPress, Qt::Key_Backspace, Qt::NoModifier);
00242         QGuiApplication::postEvent(lineEdit, pressEvent);
00243
00244         QKeyEvent *releaseEvent = new QKeyEvent(QEvent::KeyRelease, Qt::Key_Backspace, Qt::NoModifier);
00245         QGuiApplication::postEvent(lineEdit, releaseEvent);
00246     }
00247 }
00248
00249 void Keyboard::keyboardKeyClicked(const QString &characters)
00250 {
00251 #ifdef DEBUG_KEYBOARD
00252     qDebug() << Q_FUNC_INFO << lineEdit << characters;
00253 #endif
00254     if (!lineEdit)
00255         return;
00256
00257     QInputMethodEvent event;
00258     event.setCommitString(characters);
00259
00260     QGuiApplication::sendEvent(lineEdit, &event);
00261 }
00262
00263 void Keyboard::init()
00264 {
00265     setWindowFlags(Qt::WindowDoesNotAcceptFocus | Qt::Tool | Qt::FramelessWindowHint |
      Qt::WindowStaysOnTopHint | Qt::BypassWindowManagerHint);
00266
```

```
00267     stackedWidget = new QStackedWidget(this);
00268     QWidget *widgetKeyboard = new QWidget(stackedWidget);
00269     QWidget *widgetMask = new QWidget(stackedWidget);
00270 
00271     gridLayout = new QGridLayout(widgetKeyboard);
00272     hBoxLayout = new QHBoxLayout(widgetMask);
00273 
00274     mapper = new QSignalMapper(this);
00275     connect(mapper, SIGNAL(mapped(int)), SLOT(buttonClicked(int)));
00276 
00277     int row = 0;
00278     int column = 0;
00279 
00280     for (int i = 0; i < layoutSize; ++i)
00281     {
00282         if (keyboardLayout[i].key == NEXT_ROW_MARKER)
00283         {
00284             row++;
00285             column = 0;
00286             continue;
00287         }
00288 
00289         QPushButton *button = new QPushButton(this);
00290         if (keyboardLayout[i].key == Qt::Key_Enter)
00291             button->setFixedWidth(55);
00292         else
00293             button->setFixedWidth(35);
00294         button->setFixedHeight(25);
00295         button->setText(QString::fromUtf8(keyboardLayout[i].label));
00296 
00297         mapper->setMapping(button, keyboardLayout[i].key);
00298         connect(button, SIGNAL(clicked()), mapper, SLOT(map()));
00299 
00300         if (keyboardLayout[i].key == Qt::Key_Escape)
00301         {
00302             buttonEsc = new QPushButton(this);
00303             buttonEsc->setFixedWidth(35);
00304             buttonEsc->setText(QString::fromUtf8(keyboardLayout[i].label));
00305             mapper->setMapping(buttonEsc, keyboardLayout[i].key);
00306             connect(buttonEsc, SIGNAL(clicked()), mapper, SLOT(map()));
00307             hBoxLayout->addWidget(buttonEsc);
00308         }
00309         gridLayout->addWidget(button, row, column);
00310         column++;
00311     }
00312     widgetMask->setFixedSize(hBoxLayout->sizeHint());
00313     stackedWidget->addWidget(widgetKeyboard);
00314     stackedWidget->addWidget(widgetMask);
00315 }
00316 
00317 void Keyboard::makeCapsLock()
00318 {
00319     int row = 0;
00320     int column = 0;
00321     for (int i = 0; i < layoutSize; ++i)
00322     {
00323         if (keyboardLayout[i].key == NEXT_ROW_MARKER)
00324         {
00325             row++;
00326             column = 0;
00327             continue;
00328         }
00329 
00330         QPushButton *button = dynamic_cast<QPushButton*>(
      gridLayout->itemAtPosition(row, column)->widget());
00331         if(capsLock)
00332         {
00333             mapper->setMapping(button, keyboardLayout[i].keyCapsLock);
00334             button->setText(QString::fromUtf8(keyboardLayout[i].labelCapsLock));
00335         }
00336         else
00337         {
00338             mapper->setMapping(button, keyboardLayout[i].key);
00339             button->setText(QString::fromUtf8(keyboardLayout[i].label));
00340         }
00341         column++;
00342     }
00343 }
00344 
00345 QString Keyboard::keyToCharacter(int key)
00346 {
00347     for (int i = 0; i < layoutSize; ++i)
00348     {
00349         if (keyboardLayout[i].key == key)
00350         {
00351             if(capsLock)
00352                 return QString::fromUtf8(keyboardLayout[i].labelCapsLock);
00353             else
00354                 return QString::fromUtf8(keyboardLayout[i].label);
00355         }
00356         else if (keyboardLayout[i].keyCapsLock == key)
```

```
00357             return QString::fromUtf8(keyboardLayout[i].labelCapsLock);
00358     }
00359
00360     return QString();
00361 }
```

9.33 Référence du fichier Keyboard.h

Fichier qui contient la déclaration de la classe Keyboard.

```
#include <QtWidgets>
```

Classes

— class Keyboard
Déclaration de la classe Keyboard.

Macros

— #define KEYBOARD_SINGLETON

9.33.1 Description détaillée

Fichier qui contient la déclaration de la classe Keyboard.

Auteur

Thierry Vaira

based on The virtual keyboard demo software is Copyright (C) 2015 Klaralvdalens Datakonsult AB.

Définition dans le fichier Keyboard.h.

9.33.2 Documentation des macros

9.33.2.1 KEYBOARD_SINGLETON

```
#define KEYBOARD_SINGLETON
```

Définition à la ligne 14 du fichier Keyboard.h.

9.34 Keyboard.h

```
00001 #ifndef KEYBOARD_H
00002 #define KEYBOARD_H
00003
00012 #include <QtWidgets>
00013
00014 #define KEYBOARD_SINGLETON
00015 //#define DEBUG_KEYBOARD
00016
00022 class Keyboard : public QWidget
00023 {
00024     Q_OBJECT
00025 private:
00026 #ifdef KEYBOARD_SINGLETON
00027     explicit Keyboard(QLineEdit *lineEdit = Q_NULLPTR, QWidget *parent = Q_NULLPTR);
00028     static Keyboard* instance;
00029 #endif
00030     QStackedWidget *stackedWidget;
00031     QGridLayout *gridLayout;
00032     QHBoxLayout *hBoxLayout;
00033     QSignalMapper *mapper;
00034     QLineEdit *lineEdit;
00035     QPushButton *buttonEsc;
00036     bool capsLock;
00037     bool mask;
00038     void init();
00039     void makeCapsLock();
00040     QString keyToCharacter(int key);
00041
00042 protected:
00043     void paintEvent(QPaintEvent *e);
00044
00045 public:
00046 #ifndef KEYBOARD_SINGLETON
00047     explicit Keyboard(QLineEdit *lineEdit = Q_NULLPTR, QWidget *parent = Q_NULLPTR);
00048 #else
00049     static Keyboard* getInstance(QWidget *parent = Q_NULLPTR);
00050     static Keyboard* getInstance(QLineEdit *lineEdit, QWidget *parent = Q_NULLPTR
    );
00051 #endif
00052     void setLineEdit(QLineEdit *lineEdit);
00053
00054 public slots:
00055     void showKeyboard(int globalX, int globalY);
00056     void hideKeyboard();
00057     bool keyboardVisible() const;
00058
00059 signals:
00060     void specialKeyClicked(int key);
00061     void keyClicked(const QString &text);
00062
00063 private slots:
00064     void buttonClicked(int key);
00065     void focusChange(QWidget*, QWidget*);
00066     void keyboardSpecialKeyClicked(int key);
00067     void keyboardKeyClicked(const QString &characters);
00068 };
00069
00070 #endif
```

9.35 Référence du fichier main.cpp

Programme prinicpal e-stock.

```
#include "Ihm.h"
#include <QApplication>
```

Fonctions

— int main (int argc, char ∗argv[ ])

9.35.1 Description détaillée

Programme prinicpal e-stock.

Définition dans le fichier main.cpp.

9.35.2 Documentation des fonctions

9.35.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Définition à la ligne 10 du fichier main.cpp.

```
00011 {
00012     QApplication a(argc, argv);
00013     Ihm w;
00014
00015     w.show();
00016
00017     return a.exec();
00018 }
```

9.36 main.cpp

```
00001 #include "Ihm.h"
00002 #include <QApplication>
00003
00010 int main(int argc, char *argv[])
00011 {
00012     QApplication a(argc, argv);
00013     Ihm w;
00014
00015     w.show();
00016
00017     return a.exec();
00018 }
```

9.37 Référence du fichier README.md

9.38 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 e-stock est un système de gestion de stock automatisé qui permettra :
00006
00007 * de contrôler et gérer l'utilisation de produits stockés dans une armoire sensible
00008 * d'assurer la traçabilité de l'attribution du matériel et des consommables stockés
00009 * de sécuriser l'accès par un contrôle d'accès par badge RFID
00010
00011 \section section_tdm Table des matières
00012 - \ref page_README
00013 - \ref page_changelog
00014 \if todo
00015 - \ref todo
00016 \endif
00017 \if bug
00018 - \ref bug
00019 \endif
00020 - \ref page_about
00021 - \ref page_licence
00022
00023 \section section_infos Informations
00024
00025 \author Pierre-Antoine Legger <pierreantoinelegger@gmail.com>
00026 \author Joffrey Tranchat <joffrey.tranchat@gmail.com>
00027 \date 2020
00028 \version 0.2
00029 \see https://svn.riouxsvn.com/e-stock
00030
00031
```

```
00032 \page page_README README
00033
00034 [TOC]
00035
00036 # Projet {#projet}
00037
00038 ## Présentation {#presentation}
00039
00040 __e-stock__ est un système de gestion de stock automatisé qui permettra :
00041
00042 * de contrôler et gérer l'utilisation de produits stockés dans une armoire sensible
00043 * d'assurer la traçabilité de l'attribution du matériel et des consommables stockés
00044 * de sécuriser l'accès par un contrôle d'accès par badge RFID
00045
00046 Une armoire sera composée de 8 casiers maximum. Chaque casier sera équipé :
00047
00048 * d'une gâche électrique afin d'assurer son ouverture/fermeture ;
00049 * d'une balance pour assurer le comptage automatique des articles.
00050
00051 Le comptage automatique de la quantité est déterminé en fonction du poids unitaire et du poids mesuré
       sur la balance.
00052
00053 Un lecteur de badge RFID est intégré à chaque armoire pour contrôler l'accès. L'exploitation de
       l'armoire  e-stock est possible à partir de l'écran tactile intégré.
00054
00055 On distinguera deux type d'articles :
00056
00057 * les « consommables » qui sortent définitivement du stock
00058 * les « empruntables » qui peuvent être restitués après leur utilisation
00059
00060 ## Base de données MySQL {#bdd}
00061
00062 ![](./sql/e-stock-v0.4.png)
00063
00064 ~~~ {.sql}
00065 DROP DATABASE IF EXISTS `e-stock`;
00066 CREATE DATABASE IF NOT EXISTS `e-stock`;
00067 USE `e-stock`;
00068
00069 -- Création du compte d'accès à la base de données e-stock
00070 -- CREATE USER 'estock'@'%' IDENTIFIED BY 'password';
00071 -- GRANT ALL PRIVILEGES ON `e-stock`.* TO 'estock'@'%';
00072 -- FLUSH PRIVILEGES;
00073 -- --------------------------------------------------------
00074
00075 CREATE TABLE IF NOT EXISTS `Profil` (
00076   `idProfil` int(11) NOT NULL AUTO_INCREMENT,
00077   `Nom` varchar(64) NOT NULL,
00078   PRIMARY KEY (`idProfil`)
00079 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00080
00081 INSERT INTO `Profil` (`Nom`) VALUES
00082 ('Administrateur'),
00083 ('Gestionnaire'),
00084 ('Utilisateur');
00085
00086 -- --------------------------------------------------------
00087
00088 CREATE TABLE IF NOT EXISTS `Groupe` (
00089   `idGroupe` int(11) NOT NULL AUTO_INCREMENT,
00090   `Nom` varchar(64) NOT NULL,
00091   PRIMARY KEY (`idGroupe`),
00092   CONSTRAINT Unique_Groupe UNIQUE (`Nom`)
00093 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00094
00095 -- --------------------------------------------------------
00096
00097 INSERT INTO `Groupe` (`Nom`) VALUES
00098 ('PROFESSEUR'),
00099 ('1-BTS-SN'),
00100 ('T-BTS-SN');
00101
00102 -- --------------------------------------------------------
00103
00104 CREATE TABLE IF NOT EXISTS `Utilisateur` (
00105   `idUtilisateur` int(11) NOT NULL AUTO_INCREMENT,
00106   `idProfil` int(11) NOT NULL,
00107   `idGroupe` int(11) NOT NULL,
00108   `Nom` varchar(64) NOT NULL,
00109   `Prenom` varchar(64) NOT NULL,
00110   `DateValidite` date NOT NULL,
00111   `Identifiant` varchar(255) DEFAULT NULL,
00112   `MotDePasse` varchar(255) DEFAULT NULL,
00113   `Badge` varchar(11) NOT NULL,
00114   `Email` varchar(64) NOT NULL,
00115   PRIMARY KEY (`idUtilisateur`),
00116 --  CONSTRAINT Unique_Utilisateur UNIQUE (`Badge`),
00117   CONSTRAINT Utilisateur_fk_1 FOREIGN KEY (`idProfil`) REFERENCES Profil(`idProfil`) ON DELETE
       CASCADE,
00118   CONSTRAINT Utilisateur_fk_2 FOREIGN KEY (`idGroupe`) REFERENCES Groupe(`idGroupe`) ON DELETE CASCADE
00119 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
00120
00121 -- -----------------------------------------------------
00122
00123 CREATE TABLE IF NOT EXISTS `Armoire` (
00124   `idArmoire` int(11) NOT NULL,
00125   `Nom` varchar(255) NOT NULL,
00126   `Description` varchar(255) DEFAULT NULL,
00127   `nbCasiers` int(11) NOT NULL DEFAULT 8,
00128   PRIMARY KEY (`idArmoire`)
00129 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00130
00131 INSERT INTO `Armoire` (`idArmoire`,`Nom`,`Description`,`nbCasiers`) VALUES('1','B22','Atelier','2');
00132
00133 -- -----------------------------------------------------
00134
00135 CREATE TABLE IF NOT EXISTS `Type` (
00136   `idType` int(11) NOT NULL AUTO_INCREMENT,
00137   `Nom` varchar(64) NOT NULL,
00138   PRIMARY KEY (`idType`)
00139 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00140
00141 INSERT INTO `Type` (`Nom`) VALUES
00142 ('Equipement'),
00143 ('Consommable');
00144
00145 -- -----------------------------------------------------
00146
00147 CREATE TABLE IF NOT EXISTS `Unite` (
00148   `idUnite` int(11) NOT NULL AUTO_INCREMENT,
00149   `Nom` varchar(64) NOT NULL,
00150   PRIMARY KEY (`idUnite`)
00151 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00152
00153 INSERT INTO `Unite` (`Nom`) VALUES
00154 ('Metre'),
00155 ('Piece'),
00156 ('Pourcentage'),
00157 ('Poids g'),
00158 ('Poids kg');
00159
00160 -- -----------------------------------------------------
00161
00162 CREATE TABLE IF NOT EXISTS `Comptage` (
00163   `idComptage` int(11) NOT NULL AUTO_INCREMENT,
00164   `Nom` varchar(64) NOT NULL,
00165   PRIMARY KEY (`idComptage`)
00166 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00167
00168 INSERT INTO `Comptage` (`Nom`) VALUES
00169 ('Aucun'),
00170 ('Automatique'),
00171 ('CodeBarre');
00172
00173 -- -----------------------------------------------------
00174
00175 CREATE TABLE IF NOT EXISTS `Action` (
00176   `idAction` int(11) NOT NULL AUTO_INCREMENT,
00177   `Nom` varchar(64) NOT NULL,
00178   PRIMARY KEY (`idAction`)
00179 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00180
00181 INSERT INTO `Action` (`Nom`) VALUES
00182 ('Entree'),
00183 ('Sortie');
00184
00185 -- -----------------------------------------------------
00186
00187 CREATE TABLE IF NOT EXISTS `Article` (
00188   `idArticle` int(11) NOT NULL AUTO_INCREMENT,
00189   `idType` int(11) NOT NULL,
00190   --   `Type` enum('Equipement','Consommable'),
00191   `Nom` varchar(255) NOT NULL,
00192   `Code` varchar(255) NOT NULL,
00193   `Designation` varchar(255) NOT NULL,
00194   `Poids` int(11) NOT NULL,
00195   PRIMARY KEY (`idArticle`),
00196   CONSTRAINT Article_fk_1 FOREIGN KEY (`idType`) REFERENCES Type(`idType`) ON DELETE CASCADE
00197 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00198
00199 -- -----------------------------------------------------
00200
00201 CREATE TABLE IF NOT EXISTS `Stock` (
00202   `idStock` int(11) NOT NULL AUTO_INCREMENT,
00203   `idArticle` int(11) NOT NULL,
00204   `idComptage` int(11) NOT NULL,
00205   `idUnite` int(11) NOT NULL,
00206   `Quantite` int(11) DEFAULT 0,
00207   `Disponible` int(11) DEFAULT 0,
00208   `Tare` int(11) NOT NULL,
00209   `numeroCasier` int(11) NOT NULL,
00210   PRIMARY KEY (`idStock`),
```

```
00211   CONSTRAINT Unique_NumeroCasier UNIQUE (`numeroCasier`),
00212   CONSTRAINT Stock_fk_2 FOREIGN KEY (`idArticle`) REFERENCES Article(`idArticle`) ON DELETE CASCADE,
00213   CONSTRAINT Stock_fk_3 FOREIGN KEY (`idComptage`) REFERENCES Comptage(`idComptage`) ON DELETE
       CASCADE,
00214   CONSTRAINT Stock_fk_4 FOREIGN KEY (`idUnite`) REFERENCES Unite(`idUnite`) ON DELETE CASCADE
00215 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00216
00217 -- -----------------------------------------------------
00218
00219 CREATE TABLE IF NOT EXISTS `Mouvement` (
00220   `idMouvement` int(11) NOT NULL AUTO_INCREMENT,
00221   `idUtilisateur` int(11) NOT NULL,
00222   `idStock` int(11) NOT NULL,
00223   `idAction` int(11) NOT NULL,
00224   --    `Action` enum('Entree','Sortie'),
00225   `Quantite` int(11) NOT NULL,
00226   `Horodatage` datetime NOT NULL,
00227   PRIMARY KEY (`idMouvement`),
00228   CONSTRAINT Mouvement_fk_1 FOREIGN KEY (`idUtilisateur`) REFERENCES Utilisateur(`idUtilisateur`) ON
       DELETE CASCADE,
00229   CONSTRAINT Mouvement_fk_2 FOREIGN KEY (`idStock`) REFERENCES Stock(`idStock`) ON DELETE CASCADE,
00230   CONSTRAINT Mouvement_fk_3 FOREIGN KEY (`idAction`) REFERENCES Action(`idAction`) ON DELETE CASCADE
00231 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
00232 ~~~
00233
00234 ## Recette {#recette}
00235
00236 - Pierre-Antoine Legger
00237     * S'authentifier
00238     * Rechercher un article
00239     * Consulter le stock
00240     * Communiquer avec le SE pour :
00241         * Commander l'ouverture/fermeture des casiers
00242         * Afficher l'état ouvert/fermé des casiers
00243
00244 - Joffrey Tranchat
00245     * Prendre et rapporter un article
00246     * Mettre à jour le stock et les mouvements
00247     * Consulter les mouvements
00248     * Communiquer avec le SE pour :
00249     * Récupérer les pesées des casiers
00250     * Assurer le comptage automatique
00251
00252 ## Informations {#informations}
00253
00254 \author Pierre-Antoine Legger <pierreantoinelegger@gmail.com>
00255 \author Joffrey Tranchat <joffrey.tranchat@gmail.com>
00256 \date 2020
00257 \version 0.2
00258 \see https://svn.riouxsvn.com/e-stock
00259
00260
00261 \page page_about A propos
00262
00263 \author Pierre-Antoine Legger <pierreantoinelegger@gmail.com>
00264 \author Joffrey Tranchat <joffrey.tranchat@gmail.com>
00265 \date 2020
00266 \version 0.2
00267 \see https://svn.riouxsvn.com/e-stock
00268
00269
00270 \page page_licence Licence GPL
00271
00272 This program is free software; you can redistribute it and/or modify
00273 it under the terms of the GNU General Public License as published by
00274 the Free Software Foundation; either version 2 of the License, or
00275 (at your option) any later version.
00276
00277 This program is distributed in the hope that it will be useful,
00278 but WITHOUT ANY WARRANTY; without even the implied warranty of
00279 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00280 GNU General Public License for more details.
00281
00282 You should have received a copy of the GNU General Public License
00283 along with this program; if not, write to the Free Software
00284 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

9.39 Référence du fichier Rfid.cpp

Définition de la classe Rfid.

```
#include "Rfid.h"
#include <QDebug>
```

9.39.1 Description détaillée

Définition de la classe Rfid.

Auteur

Legger Pierre-Antoine

Version

1.0

Date

14 Février 2020

Définition dans le fichier Rfid.cpp.

9.40 Rfid.cpp

```
00001 #include "Rfid.h"
00002 #include <QDebug>
00003
00022 Rfid::Rfid(QObject *parent) : QObject(parent), badge("")
00023 {
00024 }
00025
00031 Rfid::~Rfid()
00032 {
00033
00034 }
00035
00043 QString Rfid::corrigerBadge(QString badge)
00044 {
00045     QString badgeCorrige = "";
00046
00047     if(!badge.isEmpty())
00048     {
00049         // effectue les remplacements des touches QWERTY en touches AZERTY
00050         badgeCorrige = badge.replace(QString::fromUtf8("Q"), "A");
00051         badgeCorrige = badge.replace(QString::fromUtf8("W"), "Z");
00052         badgeCorrige = badge.replace(QString::fromUtf8("q"), "q");
00053         badgeCorrige = badge.replace(QString::fromUtf8("w"), "z");
00054         badgeCorrige = badge.replace(QString::fromUtf8("M"), ":");
00055         badgeCorrige = badge.replace(QString::fromUtf8("à"), "0");
00056         badgeCorrige = badge.replace(QString::fromUtf8("&"), "1");
00057         badgeCorrige = badge.replace(QString::fromUtf8("é"), "2");
00058         badgeCorrige = badge.replace(QString::fromUtf8("\""), "3");
00059         badgeCorrige = badge.replace(QString::fromUtf8("'"), "4");
00060         badgeCorrige = badge.replace(QString::fromUtf8("("), "5");
00061         badgeCorrige = badge.replace(QString::fromUtf8("-"), "6");
00062         badgeCorrige = badge.replace(QString::fromUtf8("è"), "7");
00063         badgeCorrige = badge.replace(QString::fromUtf8("_"), "8");
00064         badgeCorrige = badge.replace(QString::fromUtf8("ç"), "9");
00065     }
00066     return badgeCorrige;
00067 }
00068
00075 void Rfid::setBadge(QString badge)
00076 {
00077     this->badge = badge;
00078 }
00079
00086 void Rfid::setUid(QString uid)
00087 {
00088     this->uid = uid;
00089 }
00090
00097 void Rfid::traiterBadge(QString trameBadge)
00098 {
00099     /*
00100      * Format trame reçue : RFID:xxxxxxxx
00101      * xxxxxxxx -> uid du badge
00102      */
00103     setBadge(corrigerBadge(trameBadge));
00104
00105     #ifdef DEBUG_RFID
00106         qDebug() << Q_FUNC_INFO << "Badge" << badge;
```

```
00107     #endif
00108
00109     // Vérifier si la trame est valide
00110     if(badge.startsWith("RFID:"))
00111     {
00112         extraireUid();
00113
00114         emit nouveauUidBadge(uid);
00115         #ifdef DEBUG_RFID
00116             qDebug() << Q_FUNC_INFO << "UID" << uid;
00117         #endif
00118     }
00119     else
00120     {
00121         emit erreurBadgeInvalide(ERREUR_BADGE_INVALIDE);
00122     }
00123 }
00124
00130 void Rfid::extraireUid()
00131 {
00132     setUid(badge.section(':',1,1));
00133 }
```

9.41 Référence du fichier Rfid.h

Déclaration de la classe Rfid.

```
#include <QObject>
```

Classes

— class Rfid
La classe Rfid traite la trame reçue d'un lecteur Rfid.

Macros

— #define ERREUR_BADGE_INVALIDE "Erreur badge invalide"
Message d'erreur pour un basge invalide.

9.41.1 Description détaillée

Déclaration de la classe Rfid.

Auteur

Legger Pierre-Antoine

Version

1.0

Date

Mercredi 4 Mars 2020

Définition dans le fichier Rfid.h.

9.41.2 Documentation des macros

9.41.2.1 ERREUR_BADGE_INVALIDE

```
#define ERREUR_BADGE_INVALIDE "Erreur badge invalide"
```

Message d'erreur pour un basge invalide.

Définition à la ligne 18 du fichier Rfid.h.

Référencé par Rfid : :traiterBadge().

9.42 Rfid.h

```
00001 #ifndef RFID_H
00002 #define RFID_H
00003
00017 //#define DEBUG_RFID
00018 #define ERREUR_BADGE_INVALIDE "Erreur badge invalide"
00019
00020 #include <QObject>
00021
00035 class Rfid : public QObject
00036 {
00037     Q_OBJECT
00038
00039 public:
00040     Rfid(QObject *parent = nullptr);
00041     ~Rfid();
00042
00043     QString corrigerBadge(QString badge);
00044
00045     void setBadge(QString badge);
00046     void setUid(QString uid);
00047
00048 private slots:
00049     void traiterBadge(QString trameBadge);
00050
00051 signals:
00052     void erreurBadgeInvalide(QString message);
00053     void nouveauUidBadge(QString badge);
00054
00055 private:
00056     QString badge;
00057     QString uid;
00058
00059     void extraireUid();
00060 };
00061
00062 #endif // RFID_H
```

9.43 Référence du fichier Supervision.cpp

Définition de la classe Supervision.

```
#include "Supervision.h"
#include "Ihm.h"
#include "Bdd.h"
#include "CodeBarre.h"
#include "Communication.h"
#include "Rfid.h"
#include "Utilisateur.h"
#include "Armoire.h"
#include "Article.h"
#include "Casier.h"
#include <QDate>
#include <QTime>
#include <QMessageBox>
#include <QCryptographicHash>
```

9.43.1 Description détaillée

Définition de la classe Supervision.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition dans le fichier Supervision.cpp.

9.44 Supervision.cpp

```
00001 #include "Supervision.h"
00002 #include "Ihm.h"
00003 #include "Bdd.h"
00004 #include "CodeBarre.h"
00005 #include "Communication.h"
00006 #include "Rfid.h"
00007 #include "Utilisateur.h"
00008 #include "Armoire.h"
00009 #include "Article.h"
00010 #include "Communication.h"
00011 #include "Casier.h"
00012 #include <QDate>
00013 #include <QTime>
00014 #include <QMessageBox>
00015 #include <QCryptographicHash>
00016
00036 Supervision::Supervision(Ihm *parent) : QObject(parent), ihm(parent)
00037 {
00038     // Instancie les objets dont la classe Supervision coordonne les actions
00039     bdd = Bdd::getInstance();
00040     bdd->connecter();
00041     codeBarre = new CodeBarre(this);
00042     //portSerie = new Communication(this);
00043     rfid = new Rfid(this);
00044     utilisateur = nullptr;
00045     armoire = new Armoire(this);
00046     communication = new Communication(this);
00047
00048     connecterSignauxSlots();
00049
00050     communication->demarrerCommunicationPort();
00051
00052 }
00053
00059 Supervision::~Supervision()
00060 {
00061
00062 }
00063
00069 void Supervision::deconnecterUtilisateur()
00070 {
00071     if(utilisateur != nullptr)
00072     {
00073         delete utilisateur;
00074         utilisateur = nullptr;
00075     }
00076 }
00077
00083 void Supervision::creerCasiers()
00084 {
00085     QString nbCasiers = armoire->getNbCasiers();
00086     qDebug() << Q_FUNC_INFO << "nbCasiers" << nbCasiers;
00087     if(!nbCasiers.isEmpty())
00088     {
00089         for(int i=0; i < nbCasiers.toInt(); i++)
00090         {
```

```
00091             Casier* casier = new Casier(i+1, ihm);
00092             connect(casier, SIGNAL(estOuvert(int,bool)), ihm, SLOT(gererPageScanObjet(int,bool)));
00093             casiers.push_back(casier);
00094         }
00095     }
00096     ihm->placerCasiers(casiers, FENETRE_MENU);
00097 }
00098
00105 QStringList Supervision::getInformationsArmoire()
00106 {
00107     QStringList informationsArmoire = armoire->getInformations();
00108
00109     return informationsArmoire;
00110 }
00111
00117 QVector<Casier*> Supervision::getCasiers()
00118 {
00119     return casiers;
00120 }
00121
00128 void Supervision::verifierAuthentificationBadge(QString badge)
00129 {
00130     QString requeteBDD = "SELECT * from Utilisateur where Badge = '" + badge + "';";
00131     QStringList donnees = recupererDonneesUtilisateur(requeteBDD);
00132     if(verifierDonneesUtilisateur(donnees))
00133     {
00134         connecterUtilisateur(donnees);
00135     }
00136 }
00137
00145 void Supervision::verifierAuthentificationIdentifiant(
    QString identifiant, QString motDePasse)
00146 {
00147     this->crypterMotDepasse(motDePasse);
00148
00149     #ifdef CHANGE_PASSWORD_BEFORE
00150     QString requete = QString("UPDATE Utilisateur SET MotDePasse='%1' WHERE Identifiant='%2'").arg(
    motDePasse).arg(identifiant);
00151     bdd->executer(requete);
00152     #endif
00153
00154     QString requeteBDD = "SELECT * from Utilisateur where Identifiant =  '" + identifiant + "' &&
     MotDePasse = '" + motDePasse + "';";
00155     QStringList donnees = recupererDonneesUtilisateur(requeteBDD);
00156     if(verifierDonneesUtilisateur(donnees))
00157     {
00158         connecterUtilisateur(donnees);
00159     }
00160 }
00161
00169 QStringList Supervision::recupererDonneesUtilisateur(QString
    requeteBDD)
00170 {
00171     QStringList donnees;
00172     bdd->recuperer(requeteBDD, donnees);
00173
00174     return donnees;
00175 }
00176
00183 void Supervision::crypterMotDepasse(QString &motDePasse)
00184 {
00185     if(!motDePasse.isEmpty())
00186     {
00187     motDePasse = QString(QCryptographicHash::hash((motDePasse).toLatin1(), QCryptographicHash::Md5).toHex()
    );
00188     }
00189
00190     #ifdef DEBUG_SUPERVISION
00191         qDebug() << Q_FUNC_INFO << "Mot de passe crypte" << motDePasse;
00192     #endif
00193 }
00194
00202 bool Supervision::verifierDateValidite(QString stringDateValidite)
00203 {
00204     // Verification de la date de validité
00205     QDate dateValidite = dateValidite.fromString(stringDateValidite,"yyyy-MM-dd");
00206     QDate dateActuelle = QDate::currentDate();
00207
00208     #ifdef DEBUG_SUPERVISION
00209         qDebug() << "Date actuelle" << dateActuelle;
00210         qDebug() << "Date validité" << dateValidite;
00211     #endif
00212
00213     if(dateActuelle <= dateValidite)
00214     {
00215         return true;
00216     }
00217
00218     return false;
00219 }
00220
```

```
00228 bool Supervision::verifierDonneesUtilisateur(QStringList &donnees)
00229 {
00230     #ifdef DEBUG_SUPERVISON
00231         qDebug() << Q_FUNC_INFO << donnees;
00232     #endif
00233 
00234     if(!donnees.isEmpty())
00235     {
00236         if(verifierDateValidite(donnees.at(
      TABLE_UTILISATEUR_DATE_VALIDITE)))
00237         {
00238             emit reponseDemandeDeConnexion(true, "");
00239             return true;
00240         }
00241         else
00242         {
00243             emit reponseDemandeDeConnexion(false,
      MESSAGE_ERREUR_UTILISATEUR_DATE_NON_VALIDE);
00244             return false;
00245         }
00246     }
00247     else
00248     {
00249         emit reponseDemandeDeConnexion(false,
      MESSAGE_ERREUR_UTILISATEUR_NON_VALIDE);
00250         return false;
00251     }
00252 }
00253 
00260 void Supervision::connecterUtilisateur(QStringList &donnees)
00261 {
00262     if(utilisateur != nullptr)
00263     {
00264         deconnecterUtilisateur();
00265     }
00266     utilisateur = new Utilisateur(donnees, this);
00267     #ifdef DEBUG_SUPERVISION
00268         qDebug() << Q_FUNC_INFO << utilisateur->
      getIdentifiantUtilisateur() << "authentifié";
00269     #endif
00270 
00271     #ifdef SUPERVISION_TEST_POIDS
00272         QString trameTest = "CASIERS;3;2;1745";
00273         traiterTramePoids(trameTest);
00274     #endif
00275 }
00276 
00282 void Supervision::connecterSignauxSlots()
00283 {
00284     // Armoire
00285     connect(armoire, SIGNAL(informationsArmoire(QStringList)), ihm, SLOT(
      afficherInformationsArmoire(QStringList)));
00286 
00287     // Authentification Badge
00288     connect(ihm, SIGNAL(badgeDetecte(QString)), rfid, SLOT(traiterBadge(QString)));
00289     connect(rfid, SIGNAL(erreurBadgeInvalide(QString)), ihm, SLOT(afficherErreurBadge(QString)));
00290     connect(rfid, SIGNAL(nouveauUidBadge(QString)), this, SLOT(
      verifierAuthentificationBadge(QString)));
00291 
00292     // Authentification Identifiant
00293     connect(ihm, SIGNAL(identifiantDetecte(QString, QString)), this, SLOT(
      verifierAuthentificationIdentifiant(QString, QString)));
00294 
00295     // Authentification Utilisateur
00296     connect(this, SIGNAL(reponseDemandeDeConnexion(bool,QString)),
      ihm, SLOT(traiterDemandeDeConnexion(bool,QString)));
00297 
00298     // Article
00299     connect(communication, SIGNAL(envoieTramePoids(QString)), this, SLOT(
      traiterTramePoids(QString)));
00300     connect(ihm, SIGNAL(rechercheArticle(QString)), this, SLOT(
      rechercherArticle(QString)));
00301     connect(this, SIGNAL(articlesTrouves(QVector<QStringList>)),
      ihm, SLOT(mettreAJourListeArticles(QVector<QStringList>)));
00302     connect(ihm, SIGNAL(articleSelectionne(QString)), this, SLOT(
      selectionnerArticle(QString)));
00303     connect(this, SIGNAL(donneesArticleSelectionne(QStringList)),
      ihm, SLOT(afficherDonneesArticleSelectionne(QStringList)));
00304     connect(this, SIGNAL(donneesArticleSelectionne(QVector<QStringList>)),
      ihm, SLOT(afficherDonneesArticleSelectionne(QVector<QStringList>)));
00305     connect(this, SIGNAL(erreurDepassementQuantite()),
      ihm, SLOT(afficherErreurDepassementQuantite()));
00306 
00307     // CodeBarre
00308     connect(ihm, SIGNAL(boutonPrendre(bool)), codeBarre, SLOT(prendreOuAjouter(bool)));
00309     connect(ihm, SIGNAL(boutonAjouter(bool)), codeBarre, SLOT(prendreOuAjouter(bool)));
00310     connect(ihm, SIGNAL(codeBarreObjetScanner(QString)), codeBarre, SLOT(traiterCodeBarre(
      QString)));
00311     connect(this, SIGNAL(erreurAucunArticleAvecCodeBarre()),
      ihm, SLOT(afficherErreurPasArticleAvecCodeBarre()));
00312     connect(codeBarre, SIGNAL(prendreObjet(QString)), this, SLOT(
```

```
      prendreObjetAvecCodeBarre(QString)));
00313     connect(codeBarre, SIGNAL(ajouterObjet(QString)), this, SLOT(
      ajouterObjetAvecCodeBarre(QString)));
00314     connect(ihm, SIGNAL(envoyerQuantite(int)), codeBarre, SLOT(changerQuantiteObjet(int)));
00315     connect(this, SIGNAL(erreurArticleInsuffisants()),
      ihm, SLOT(afficherErreurArticleInsuffisants()));
00316     connect(this, SIGNAL(erreurQuantiteTropElevee()),
      ihm, SLOT(afficherErreurDepassementQuantite()));
00317     connect(this, SIGNAL(erreurAucunCasierOuvert()), ihm, SLOT(
      afficherErreurAucunCasierOuvert()));
00318 }
00319
00326 void Supervision::rechercherArticle(QString recherche)
00327 {
00328     QString requete = "SELECT Stock.NumeroCasier, Article.idType, Article.Nom, Stock.Quantite,
       Stock.Disponible, Article.Designation FROM Stock INNER JOIN Article ON Stock.idArticle = Article.idArticle WHERE
       Article.Nom LIKE '%" + recherche + "%' OR Article.Code LIKE '%" + recherche + "%' OR Article.Designation LIKE '%" +
       recherche + "%' ORDER BY Stock.NumeroCasier ASC";
00329
00330     QVector<QStringList> listeArticlesTrouves;
00331     bdd->recuperer(requete, listeArticlesTrouves);
00332
00333     emit articlesTrouves(listeArticlesTrouves);
00334 }
00335
00341 void Supervision::selectionnerArticle(QString nomArticle)
00342 {
00343     #ifdef DEBUG_SUPERVISION
00344         qDebug() << Q_FUNC_INFO << "Nom article" << nomArticle;
00345     #endif
00346
00347     Article *article = new Article(this);
00348     QVector<QStringList> donneesArticle;
00349     QStringList donnees;
00350
00351     unsigned int nombreCasiers = Article::recupererNombreCasiersPourNomArticle
      (nomArticle);
00352     #ifdef DEBUG_SUPERVISION
00353         qDebug() << Q_FUNC_INFO << "nombreCasiers" << nombreCasiers;
00354     #endif
00355
00356     if(nombreCasiers > 1)
00357     {
00358         QVector<QString> numeroDesCasiers;
00359
00360         numeroDesCasiers = Article::recupererNumeroCasierPourNomArticle
      (nomArticle);
00361
00362         for(int i = 0; i < numeroDesCasiers.size(); i++)
00363         {
00364             article->recupererDonneesArticleParNumeroCasier(
      numeroDesCasiers[i]);
00365             ajouterDonneesArticle(article, donneesArticle, donnees);
00366         }
00367
00368         if(!donneesArticle.isEmpty())
00369         {
00370             emit donneesArticleSelectionne(donneesArticle);
00371         }
00372     }
00373     else
00374     {
00375         article->recupererDonneesArticleParNom(nomArticle);
00376         ajouterDonneesArticle(article, donneesArticle, donnees);
00377
00378         if(!donneesArticle.isEmpty())
00379         {
00380             emit donneesArticleSelectionne(donneesArticle.at(0));
00381         }
00382     }
00383 }
00384
00391 void Supervision::traiterTramePoids(QString trame)
00392 {
00393     #ifdef DEBUG_SUPERVISION
00394         qDebug() << Q_FUNC_INFO << trame;
00395     #endif
00396
00397     QString numCasier = extraireNumeroCasier(trame);
00398
00399     Article *article = new Article(this);
00400     if(article->recupererDonneesArticleParNumeroCasier(numCasier))
00401     {
00402         #ifdef DEBUG_SUPERVISION
00403             qDebug() << Q_FUNC_INFO << "Article" << article->get(
      TABLE_ARTICLE_NOM_ARTICLE) << article->get(
      TABLE_ARTICLE_QUANTITE) << article->get(
      TABLE_ARTICLE_DISPONIBLE);
00404         #endif
00405
00406         int nombreArticle = compter(article->get(TABLE_ARTICLE_POIDS),
```

```
      extrairePoids(trame), article->get(TABLE_ARTICLE_TARE));
00407
00408         QString strArticleQuantite = article->get(TABLE_ARTICLE_QUANTITE);
00409
00410         int articleQuantite = strArticleQuantite.toInt();
00411
00412         if(nombreArticle > articleQuantite)
00413         {
00414             emit erreurDepassementQuantite();
00415         }
00416         else
00417         {
00418             article->mettreAJourQuantite(QString::number(nombreArticle));
00419
00420             QString idUtilisateur = utilisateur->getIdUtilisateur();
00421             QString idStock = article->get(TABLE_ARTICLE_ID_STOCK);
00422             QString idAction = QString::number(
      comptageAutomatiqueAjouterOuPrendre(article->
      get(TABLE_ARTICLE_DISPONIBLE), nombreArticle));
00423             QString quantite = QString::number(comptageAutomatiqueQuantite(
      article->get(TABLE_ARTICLE_DISPONIBLE), nombreArticle));
00424
00425             mettreAJourMouvement(idUtilisateur, idStock,  idAction, quantite);
00426         }
00427     }
00428     else
00429     {
00430         #ifdef DEBUG_SUPERVISION
00431             qDebug() << Q_FUNC_INFO << "Article introuvable !";
00432         #endif
00433     }
00434 }
00435
00443 unsigned int Supervision::comptageAutomatiqueAjouterOuPrendre
      (QString nbArticleAvant, int nbArticleApres)
00444 {
00445     int intNbArticleAvant = nbArticleAvant.toInt();
00446
00447     if(intNbArticleAvant > nbArticleApres)
00448     {
00449         return 2;
00450     }
00451     else
00452     {
00453         return 1;
00454     }
00455 }
00456
00464 unsigned int Supervision::comptageAutomatiqueQuantite(QString
      nbArticleAvant, int nbArticleApres)
00465 {
00466     int intNbArticleAvant = nbArticleAvant.toInt();
00467
00468     if(intNbArticleAvant > nbArticleApres)
00469     {
00470         return intNbArticleAvant - nbArticleApres;
00471     }
00472     else
00473     {
00474         return nbArticleApres - intNbArticleAvant;
00475     }
00476 }
00477
00485 QString Supervision::extrairePoids(QString trame)
00486 {
00487     QString poids = trame.section(';',3,3);
00488
00489     #ifdef DEBUG_SUPERVISION
00490         qDebug() << Q_FUNC_INFO << "poids:" << poids;
00491     #endif
00492
00493     return poids;
00494 }
00495
00503 QString Supervision::extraireNumeroCasier(QString trame)
00504 {
00505     QString numCasier = trame.section(';',2,2);
00506
00507     #ifdef DEBUG_SUPERVISION
00508         qDebug() << Q_FUNC_INFO << "numCasier:" << numCasier;
00509     #endif
00510
00511     return numCasier;
00512 }
00513
00522 int Supervision::compter(QString poidsArticle, QString poidsTotal, QString tare)
00523 {
00524     double doublePoidsArticle = poidsArticle.toDouble();
00525     double doublePoidsTotal = poidsTotal.toDouble();
00526     double doubleTare = tare.toDouble();
00527
```

```
00528     //comptage du nombre d'articles
00529
00530     double doubleNombreArticle = qRound((doublePoidsTotal - doubleTare) / doublePoidsArticle);
00531     QString strNombreArticle = QString::number(doubleNombreArticle, 'f',
      PRECISION);
00532     int nombreArticle = strNombreArticle.toInt();
00533
00534     #ifdef DEBUG_SUPERVISION
00535         qDebug() << Q_FUNC_INFO << "nombreArticle:" << nombreArticle;
00536     #endif
00537
00538     return nombreArticle;
00539 }
00540
00546 QString Supervision::recupererHorodatage()
00547 {
00548     QDate qDate(QDate::currentDate());
00549     QString date = qDate.toString("yyyy-MM-dd");
00550
00551     QTime time(QTime::currentTime());
00552     QString heure = time.toString("hh:mm:ss");
00553
00554     return date + " " + heure;
00555 }
00556
00565 void Supervision::mettreAJourMouvement(QString idUtilisateur, QString
      idStock, QString idAction, QString quantite)
00566 {
00567     QString horodatage = recupererHorodatage();
00568     QString requete ="INSERT INTO Mouvement(idUtilisateur, idStock, idAction, Quantite, Horodatage)
       VALUES('" + idUtilisateur + "', '" + idStock + "', '" + idAction + "', '" + quantite + "', '" + horodatage + "');";
00569     bdd->executer(requete);
00570 }
00571
00579 void Supervision::ajouterDonneesArticle(
      Article *article, QVector<QStringList> &donneesArticle, QStringList &donnees)
00580 {
00581     donnees << article->get(TABLE_ARTICLE_QUANTITE);
00582     donnees << article->get(TABLE_ARTICLE_DISPONIBLE);
00583     donnees << article->get(TABLE_ARTICLE_NUMERO_CASIER);
00584
00585     donneesArticle.push_back(donnees);
00586     donnees.clear();
00587 }
00588
00596 bool Supervision::verifierArticlePresentDansCasier(QString
      numCasier, QString idArticle)
00597 {
00598     QString requete = "SELECT idArticle FROM Stock WHERE numeroCasier = "+ numCasier + ";";
00599     QString donnees;
00600
00601     bdd->recuperer(requete, donnees);
00602
00603     if(donnees == idArticle)
00604     {
00605         return true;
00606     }
00607     else
00608     {
00609         return false;
00610     }
00611 }
00612
00618 void Supervision::prendreObjetAvecCodeBarre(QString
      codeBarre)
00619 {
00620     int numeroCasier = rechercherCasierOuvert();
00621
00622     if(numeroCasier == -1)
00623         return;
00624
00625     #ifdef DEBUG_SUPERVISION
00626         qDebug() << Q_FUNC_INFO << "codeBarre" << codeBarre << "casier" << numeroCasier;
00627     #endif
00628
00629     unsigned int quantiteDisponible = this->codeBarre->recupererQuantiteDisponibleParNumeroCasier(
      QString::number(casiers[numeroCasier]->getNumero()));
00630     int quantite = quantiteDisponible - this->codeBarre->getQuantiteObjet();
00631     QString idArticle = QString::number(this->codeBarre->recupererIdArticleAvecCodeBarres(codeBarre));
00632
00633     if(casiers[numeroCasier]->estOuvert())
00634     {
00635         if(verifierArticlePresentDansCasier(QString::number(
      casiers[numeroCasier]->getNumero()), idArticle))
00636         {
00637             if(quantite >= 0)
00638             {
00639                 QString strQuantite = QString::number(quantite);
00640                 QString requete = "UPDATE Stock SET Disponible = '" +  strQuantite + "' WHERE
       Stock.idArticle = '" + idArticle + "'";
00641                 bdd->executer(requete);
```

```
00642                 QString idUtilisateur = utilisateur->getIdUtilisateur();
00643                 QString idStock = recupererIdStockAvecNumeroCasier(
      numeroCasier+1 );
00644                 QString idAction = "1";
00645                 QString quantiteMouvement = QString::number(this->codeBarre->getQuantiteObjet());
00646                 mettreAJourMouvement(idUtilisateur, idStock, idAction,
      quantiteMouvement);
00647             }
00648             else
00649             {
00650                 erreurArticleInsuffisants();
00651             }
00652         }
00653         else
00654         {
00655             emit erreurAucunArticleAvecCodeBarre();
00656         }
00657     }
00658     else
00659     {
00660         erreurAucunCasierOuvert();
00661     }
00662 }
00663
00669 void Supervision::ajouterObjetAvecCodeBarre(QString
      codeBarre)
00670 {
00671     int numeroCasier = rechercherCasierOuvert();
00672
00673     if(numeroCasier == -1)
00674         return;
00675
00676     #ifdef DEBUG_SUPERVISION
00677         qDebug() << Q_FUNC_INFO << "codeBarre" << codeBarre << "casier" << numeroCasier;
00678     #endif
00679
00680     unsigned int quantiteMax = this->codeBarre->recupererQuantiteMaxParNumeroCasier(QString::number(
      casiers[numeroCasier]->getNumero()));
00681     unsigned int quantiteDisponible = this->codeBarre->recupererQuantiteDisponibleParNumeroCasier(
      QString::number(casiers[numeroCasier]->getNumero()));
00682     unsigned int quantite = quantiteDisponible + this->codeBarre->getQuantiteObjet();
00683     QString idArticle = QString::number(this->codeBarre->recupererIdArticleAvecCodeBarres(codeBarre));
00684
00685     if(casiers[numeroCasier]->estOuvert())
00686     {
00687         if(verifierArticlePresentDansCasier(QString::number(
      casiers[numeroCasier]->getNumero()), idArticle))
00688         {
00689             if(quantite <= quantiteMax)
00690             {
00691                 QString strQuantite = QString::number(quantite);
00692                 QString requete = "UPDATE Stock SET Disponible = '" +  strQuantite + "' WHERE
       Stock.idArticle = '" + idArticle + "'";
00693                 bdd->executer(requete);
00694                 QString idUtilisateur = utilisateur->getIdUtilisateur();
00695                 QString idStock = recupererIdStockAvecNumeroCasier(
      numeroCasier+1);
00696                 QString idAction = "2";
00697                 QString quantiteMouvement = QString::number(this->codeBarre->getQuantiteObjet());
00698                 mettreAJourMouvement(idUtilisateur, idStock, idAction,
      quantiteMouvement);
00699             }
00700             else
00701             {
00702                 emit erreurQuantiteTropElevee();
00703             }
00704         }
00705         else
00706         {
00707             emit erreurAucunArticleAvecCodeBarre();
00708         }
00709     }
00710     else
00711     {
00712         erreurAucunCasierOuvert();
00713     }
00714 }
00715
00721 int Supervision::rechercherCasierOuvert()
00722 {
00723     for(int i=0; i < casiers.size(); i++)
00724     {
00725         if(casiers[i]->estOuvert())
00726             return i;
00727     }
00728     return -1;
00729 }
00730
00737 QString Supervision::recupererIdStockAvecNumeroCasier(int
      numeroCasier)
00738 {
```

```
00739     QString strNumeroCasier = QString::number(numeroCasier);
00740     QString requete = "SELECT idStock FROM Stock WHERE numeroCasier = " + strNumeroCasier + ";";
00741     QString donnees;
00742     bdd->recuperer(requete, donnees);
00743
00744     return donnees;
00745 }
```

9.45 Référence du fichier Supervision.h

Déclaration de la classe Supervision.

```
#include <QObject>
#include <QString>
#include <QVector>
```

Classes

— class Supervision
La classe Supervision permet de superviser l'ensemble de l'application.

Macros

— #define DEBUG_SUPERVISION
— #define PRECISION 0
— #define SUPERVISION_TEST_POIDS

9.45.1 Description détaillée

Déclaration de la classe Supervision.

Auteur

Legger Pierre-Antoine
Tranchat Joffrey

Version

1.0

Date

Mercredi 12 Février 2020

Définition dans le fichier Supervision.h.

9.45.2 Documentation des macros

9.45.2.1 DEBUG_SUPERVISION

```
#define DEBUG_SUPERVISION
```

Définition à la ligne 22 du fichier Supervision.h.

9.45.2.2 PRECISION

```
#define PRECISION 0
```

Définition à la ligne 26 du fichier Supervision.h.

Référencé par Supervision : :compter().

9.45.2.3 SUPERVISION_TEST_POIDS

```
#define SUPERVISION_TEST_POIDS
```

Définition à la ligne 24 du fichier Supervision.h.

9.46 Supervision.h

```
00001 #ifndef SUPERVISION_H
00002 #define SUPERVISION_H
00003
00018 #include <QObject>
00019 #include <QString>
00020 #include <QVector>
00021
00022 #define DEBUG_SUPERVISION
00023 //#define CHANGE_PASSWORD_BEFORE
00024 #define SUPERVISION_TEST_POIDS
00025
00026 #define PRECISION 0
00027
00028 class Armoire;
00029 class Article;
00030 class Ihm;
00031 class Bdd;
00032 class CodeBarre;
00033 class Communication;
00034 class Rfid;
00035 class Utilisateur;
00036 class Communication;
00037 class Casier;
00038
00052 class Supervision : QObject
00053 {
00054     Q_OBJECT
00055
00056 public:
00057     Supervision(Ihm *parent = nullptr);
00058     ~Supervision();
00059
00060     void deconnecterUtilisateur();
00061     void creerCasiers();
00062     QStringList getInformationsArmoire();
00063     QVector<Casier*> getCasiers();
00064
00065 public slots:
00066     void verifierAuthentificationBadge(QString badge);
00067     void verifierAuthentificationIdentifiant(QString identifiant,
      QString motDePasse);
00068     void rechercherArticle(QString recherche);
00069     void selectionnerArticle(QString nomArticle);
00070     void traiterTramePoids(QString trame);
00071     void prendreObjetAvecCodeBarre(QString codeBarre);
00072     void ajouterObjetAvecCodeBarre(QString codeBarre);
00073
```

```
00074 signals:
00075     void reponseDemandeDeConnexion(bool, QString);
00076     void erreurDepassementQuantite();
00077     void articlesTrouves(QVector<QStringList>);
00078     void donneesArticleSelectionne(QVector<QStringList>);
00079     void donneesArticleSelectionne(QStringList);
00080     void erreurArticleInsuffisants();
00081     void erreurQuantiteTropElevee();
00082     void erreurAucunCasierOuvert();
00083     void erreurAucunArticleAvecCodeBarre();
00084
00085 private:
00086     Ihm *ihm;
00087     Bdd *bdd;
00088     Rfid *rfid;
00089     Utilisateur *utilisateur;
00090     CodeBarre *codeBarre;
00091     Armoire *armoire;
00092     Communication *communication;
00093     QVector<Casier*> casiers;
00094
00095     void connecterSignauxSlots();
00096
00097     // Athentification
00098     QStringList recupererDonneesUtilisateur(QString requeteBDD);
00099     void crypterMotDepasse(QString &motDePasse);
00100     bool verifierDateValidite(QString stringDateValidite);
00101     bool verifierDonneesUtilisateur(QStringList &donnees);
00102     void connecterUtilisateur(QStringList &donnees);
00103
00104     QString extrairePoids(QString trame);
00105     QString extraireNumeroCasier(QString trame);
00106     int compter(QString poidArticle, QString poidTotal, QString tare);
00107     unsigned int comptageAutomatiqueAjouterOuPrendre(QString
      nbArticleAvant, int nbArticleApres);
00108     unsigned int comptageAutomatiqueQuantite(QString nbArticleAvant, int
      nbArticleApres);
00109     void mettreAJourMouvement(QString idUtilisateur, QString idStock, QString idAction,
       QString quantite);
00110
00111     void ajouterDonneesArticle(Article *article, QVector<QStringList> &
      donneesArticle, QStringList &donnees);
00112
00113     bool verifierArticlePresentDansCasier(QString numCasier, QString
      idArticle);
00114
00115     QString recupererHorodatage();
00116     QString recupererIdStockAvecNumeroCasier(int numeroCasier);
00117     int rechercherCasierOuvert();
00118
00119 };
00120
00121 #endif // SUPERVISION_H
```

9.47 Référence du fichier Utilisateur.cpp

Définition de la classe Utilisateur.

```
#include "Utilisateur.h"
#include <QDebug>
```

9.47.1 Description détaillée

Définition de la classe Utilisateur.

Auteur

Legger Pierre-Antoine

Version

1.0

Date

mercredi 04 Mars 2020

Définition dans le fichier Utilisateur.cpp.

9.48 Utilisateur.cpp

```
00001 #include "Utilisateur.h"
00002 #include <QDebug>
00003
00022 Utilisateur::Utilisateur(QObject *parent) :
      QObject(parent)
00023 {
00024     #ifdef DEBUG_UTILISATEUR
00025         qDebug() << Q_FUNC_INFO;
00026     #endif
00027     idUtilisateur = "";
00028     idProfil = "";
00029     idGroupe = "";
00030     nom = "";
00031     prenom = "";
00032     dateValidite = "";
00033     identifiant = "";
00034     motDePasse = "";
00035     badge = "";
00036     email = "";
00037 }
00038
00045 Utilisateur::Utilisateur(QStringList donnees, QObject *parent) :
      QObject(parent)
00046 {
00047     #ifdef DEBUG_UTILISATEUR
00048         qDebug() << Q_FUNC_INFO << donnees;
00049     #endif
00050     idUtilisateur = donnees.at(TABLE_UTILISATEUR_ID_UTILISATEUR
      );
00051     idProfil = donnees.at(TABLE_UTILISATEUR_ID_PROFIL);
00052     idGroupe = donnees.at(TABLE_UTILISATEUR_ID_GROUPE);
00053     nom = donnees.at(TABLE_UTILISATEUR_NOM);
00054     prenom = donnees.at(TABLE_UTILISATEUR_PRENOM);
00055     dateValidite = donnees.at(TABLE_UTILISATEUR_DATE_VALIDITE);
00056     identifiant = donnees.at(TABLE_UTILISATEUR_IDENTIFIANT);
00057     motDePasse = donnees.at(TABLE_UTILISATEUR_MOT_DE_PASSE);
00058     badge = donnees.at(TABLE_UTILISATEUR_BADGE);
00059     email = donnees.at(TABLE_UTILISATEUR_EMAIL);
00060 }
00061
00067 Utilisateur::~Utilisateur()
00068 {
00069
00070 }
00071
00078 QString Utilisateur::getIdentifiantUtilisateur()
00079 {
00080     return nom + " " + prenom;
00081 }
00082
00088 QString Utilisateur::getIdUtilisateur()
00089 {
00090     return idUtilisateur;
00091 }
```

9.49 Référence du fichier Utilisateur.h

Déclaration de la classe Utilisateur.

```
#include <QObject>
```

Classes

— class Utilisateur
La classe Utilisateur gère les données relative à l'utilisateur.

Énumérations

— enum ChampsUtilisateur {
TABLE_UTILISATEUR_ID_UTILISATEUR, TABLE_UTILISATEUR_ID_PROFIL, TABLE_UTILISATEUR_ID_GROUPE, TA↩
BLE_UTILISATEUR_NOM,
TABLE_UTILISATEUR_PRENOM, TABLE_UTILISATEUR_DATE_VALIDITE, TABLE_UTILISATEUR_IDENTIFIANT, TABL↩
E_UTILISATEUR_MOT_DE_PASSE,
TABLE_UTILISATEUR_BADGE, TABLE_UTILISATEUR_EMAIL }
Définit les différents champs de la table Utilisateur.

9.49.1 Description détaillée

Déclaration de la classe Utilisateur.

Auteur

Legger Pierre-Antoine

Version

1.0

Date

mercredi 04 Mars 2020

Définition dans le fichier Utilisateur.h.

9.49.2 Documentation du type de l'énumération

9.49.2.1 ChampsUtilisateur

```
enum ChampsUtilisateur
```

Définit les différents champs de la table Utilisateur.

Valeurs énumérées

TABLE_UTILISATEUR_ID_UTILISATEUR	Emplacment de l'idUtilisateur.
TABLE_UTILISATEUR_ID_PROFIL	Emplacment de L'idProfil.
TABLE_UTILISATEUR_ID_GROUPE	Emplacment de l'idGroupe.
TABLE_UTILISATEUR_NOM	Emplacment du nom.
TABLE_UTILISATEUR_PRENOM	Emplacment du prenom.
TABLE_UTILISATEUR_DATE_VALIDITE	Emplacment de la date de validite.
TABLE_UTILISATEUR_IDENTIFIANT	Emplacment de l'identifiant.
TABLE_UTILISATEUR_MOT_DE_PASSE	Emplacment du mot de passe.
TABLE_UTILISATEUR_BADGE	Emplacment du badge.
TABLE_UTILISATEUR_EMAIL	Emplacment du mail.

Définition à la ligne 25 du fichier Utilisateur.h.

```
00026 {
00027     TABLE_UTILISATEUR_ID_UTILISATEUR,
00028     TABLE_UTILISATEUR_ID_PROFIL,
00029     TABLE_UTILISATEUR_ID_GROUPE,
00030     TABLE_UTILISATEUR_NOM,
00031     TABLE_UTILISATEUR_PRENOM,
00032     TABLE_UTILISATEUR_DATE_VALIDITE,
00033     TABLE_UTILISATEUR_IDENTIFIANT,
00034     TABLE_UTILISATEUR_MOT_DE_PASSE,
00035     TABLE_UTILISATEUR_BADGE,
00036     TABLE_UTILISATEUR_EMAIL
00037 };
```

9.50 Utilisateur.h

```
00001 #ifndef UTILISATEUR_H
00002 #define UTILISATEUR_H
00003
00017 #include <QObject>
00018
00019 //#define DEBUG_UTILISATEUR
00020
00025 enum ChampsUtilisateur
00026 {
00027     TABLE_UTILISATEUR_ID_UTILISATEUR,
00028     TABLE_UTILISATEUR_ID_PROFIL,
00029     TABLE_UTILISATEUR_ID_GROUPE,
00030     TABLE_UTILISATEUR_NOM,
00031     TABLE_UTILISATEUR_PRENOM,
00032     TABLE_UTILISATEUR_DATE_VALIDITE,
00033     TABLE_UTILISATEUR_IDENTIFIANT,
00034     TABLE_UTILISATEUR_MOT_DE_PASSE,
00035     TABLE_UTILISATEUR_BADGE,
00036     TABLE_UTILISATEUR_EMAIL
00037 };
00038
00052 class Utilisateur : public QObject
00053 {
00054     Q_OBJECT
00055 public:
00056     Utilisateur(QObject *parent = nullptr);
00057     Utilisateur(QStringList donnees, QObject *parent = nullptr);
00058     ~Utilisateur();
00059
00060     QString getIdentifiantUtilisateur();
00061     QString getIdUtilisateur();
00062
00063 private slots:
00064
00065 signals:
00066
00067 private:
00068      QString idUtilisateur;
00069      QString idProfil;
00070      QString idGroupe;
00071      QString nom;
00072      QString prenom;
00073      QString dateValidite;
00074      QString identifiant;
00075      QString motDePasse;
00076      QString badge;
00077      QString email;
00078 };
00079
00080 #endif // UTILISATEUR_H
```