Projet Chrono-Cross

1.1

BTS SN-IR LaSalle Avignon 2019

ANDREO Michael
TURLIN Suzie

Table des matières

1 Le projet Chrono-Cross

Système informatisé de gestion des courses de cross chronométrées pour un établissement scolaire.

Suzie TURLIN suzie.turlin@gmail.com

Michaël ANDREO andreo.michael@outlook.fr

1.1 Table des matières

2 Changelog

r78 | mandreo | 2019-06-05 16 :59 :16 +0200 (mer. 05 juin 2019) | 1 ligne

Ajout du bouton inscrire (pour l'instant affiche juste en debug), ajout du premier chiffre dans le lineInscrire

r77 | mandreo | 2019-06-04 21 :14 :00 +0200 (mar. 04 juin 2019) | 1 ligne

Modification des noms de certaines méthodes

r76 | mandreo | 2019-06-04 16 :40 :22 +0200 (mar. 04 juin 2019) | 1 ligne

Affichage des manifestations et des courses disponnibles pour inscrire un coureur sélectionné, affichage des coureurs déjà inscrits

r75 | mandreo | 2019-06-04 11 :11 :58 +0200 (mar. 04 juin 2019) | 2 lignes

Ajout d'une condition si Windows ou linux pour le port

r74 | mandreo | 2019-06-04 10 :56 :42 +0200 (mar. 04 juin 2019) | 1 ligne

Modification du script SQL , ajout de la colonne Etat à Course de type enum (ACourir, Prete, EnCours, Arretee, Terminee), ajout d'une manifestation et de deux courses

r73 | mandreo | 2019-06-04 09 :35 :01 +0200 (mar. 04 juin 2019) | 2 lignes

Tests sous Windows

r72 | tvaira | 2019-06-03 18 :06 :28 +0200 (lun. 03 juin 2019) | 2 lignes

Mise en place QStackedWidget

r71 | mandreo | 2019-06-03 02 :10 :31 +0200 (lun. 03 juin 2019) | 1 ligne

Modification de la classe course

r70 | mandreo | 2019-06-03 02 :08 :56 +0200 (lun. 03 juin 2019) | 1 ligne

Modification de l'aspect de l'IHM, affichage de la base de donnée

r69 | mandreo | 2019-06-02 19 :19 :00 +0200 (dim. 02 juin 2019) | 1 ligne

Modification des classes de Gestion-Cross, ajout de la classe GestionBDD

r68 | mandreo | 2019-05-30 21 :45 :24 +0200 (jeu. 30 mai 2019) | 1 ligne

Ajout et mise en place des widgets, aucun connect ou méthode pour l'instant

r67 | mandreo | 2019-05-29 17 :14 :16 +0200 (mer. 29 mai 2019) | 1 ligne

Ajout des premiers widgets pour IHMGestionCross, ajout de la connection avec la BDD

r66 | mandreo | 2019-05-29 15 :29 :52 +0200 (mer. 29 mai 2019) | 1 ligne

Ajout du projet Gestion-Cross

r65 | mandreo | 2019-05-29 15 :08 :08 +0200 (mer. 29 mai 2019) | 1 ligne

Modification de la méthode associerDossard() de la classe IHM

r64 | mandreo | 2019-05-28 18 :07 :39 +0200 (mar. 28 mai 2019) | 1 ligne

Modification des noms et mise en ordre des attributs des classes

r63 | mandreo | 2019-05-24 11 :32 :57 +0200 (ven. 24 mai 2019) | 1 ligne

Modification de nommage des classes et du doxygene

r62 | mandreo | 2019-05-23 16 :40 :50 +0200 (jeu. 23 mai 2019) | 1 ligne

Ajout et modification du doxygene des classes

r61 | mandreo | 2019-05-23 14 :45 :21 +0200 (jeu. 23 mai 2019) | 1 ligne

Ajout de paramètres supplémentaire pour la vérification de dossard, ajout de couleur or argent et bronze pour les trois premiers avec une police différentes, ajout du code '0000' pour supprimer le premier temps affiche ensuite une page pour demande la confirmation

r60 | mandreo | 2019-05-21 18 :52 :47 +0200 (mar. 21 mai 2019) | 1 ligne

Ajout à la table Course une colonne Etat (Acourir, Prete, Encours, Arretee, Terminee), Mise en place de signal au niveau de l'acquitement dans la classe Chrono, Ajout des deux touches entrer pour associer un numero de dossard

r59 | mandreo | 2019-05-20 15 :37 :49 +0200 (lun. 20 mai 2019) | 1 ligne

Ajout des informations des courses et des arrivées. Ajout de l'état led orange pour course et chrono. Ajout des qlcd pour les arrivées et le nombre d'inscris. Modification de la mise en page

r58 | mandreo | 2019-05-20 12 :23 :19 +0200 (lun. 20 mai 2019) | 1 ligne

Changement nom des attributs de la classe IHMChronoCross (widget)

r57 | tvaira | 2019-05-20 11 :01 :46 +0200 (lun. 20 mai 2019) | 1 ligne

Ajout documentation Doxygen pour tag 0.2

r56 | tvaira | 2019-05-20 10 :58 :42 +0200 (lun. 20 mai 2019) | 1 ligne

Passage des fichiers Doxygen au format Markdown

r55 | mandreo | 2019-05-17 19 :48 :33 +0200 (ven. 17 mai 2019) | 1 ligne

Ajout de boolean pour le modeClock, le chrono et pour la course

r54 | mandreo | 2019-05-17 11 :47 :52 +0200 (ven. 17 mai 2019) | 1 ligne

Modification de la police de l'IHM(listecourse et listemanifestation), ajout du message pour le numéro de dossard (valie et invalide)

r53 | mandreo | 2019-05-10 12 :40 :42 +0200 (ven. 10 mai 2019) | 1 ligne

Ajout d'une fonctionnalité si la liste de temps est vide on associe pas de dossard

r52 | mandreo | 2019-05-10 09 :13 :59 +0200 (ven. 10 mai 2019) | 1 ligne

Ajout du tag 0.2

r51 | mandreo | 2019-05-10 08 :52 :12 +0200 (ven. 10 mai 2019) | 1 ligne

Ajout du doxygène pour le tag 0.2

r50 | mandreo | 2019-05-09 16 :24 :49 +0200 (jeu. 09 mai 2019) | 1 ligne

Synchronisation chronomètre QLCDChrono. Lance la communication une fois que l'utilisateur a choisi une manifestation et une course.

r49 | sturlin | 2019-05-09 10 :43 :10 +0200 (jeu. 09 mai 2019) | 1 ligne

ajout du projet resultat-cross

r48 | sturlin | 2019-05-09 10 :40 :48 +0200 (jeu. 09 mai 2019) | 1 ligne

suppression du projet resultat-cross

r47 | sturlin | 2019-05-03 14 :28 :40 +0200 (ven. 03 mai 2019) | 1 ligne

ajout du doxygen sur coureur.h ∗.cpp et sur IHMResultatCross.h ∗.cpp

r46 | sturlin | 2019-05-03 14 :26 :26 +0200 (ven. 03 mai 2019) | 1 ligne

ajout du tableau pour le classement des coureurs, plus méthode getListeCoureurs

r45 | mandreo | 2019-05-03 11 :43 :46 +0200 (ven. 03 mai 2019) | 1 ligne

Modification du listView dorénavant un dossard est entré, le temps est retiré de la liste. CBListe oppérationnel et se met à jour selon la manifestation

r44 | mandreo | 2019-05-02 17 :34 :36 +0200 (jeu. 02 mai 2019) | 1 ligne

Maj de la liste Manifestation (affichage de la BDD), Verification des dossards oppérationelle, Affichage des deux modelview list et table, après vérification des dossards ajout des information dans le table classement

r43 | sturlin | 2019-05-02 14 :18 :57 +0200 (jeu. 02 mai 2019) | 1 ligne

Ajout du tableau pour l'application IHM manifestation

r42 | mandreo | 2019-04-25 17 :27 :28 +0200 (jeu. 25 avril 2019) | 1 ligne

Ajout verification des numéro de dossards affectée à une arrivée, ajout d'une arrivée et de son dossard à la BDD, modification des leds, modification du bDemarrer

r41 | mandreo | 2019-04-20 18 :06 :15 +0200 (sam. 20 avril 2019) | 1 ligne

Modification d'un temps d'arrviee. Passage de MS en S

r40 | mandreo | 2019-04-16 16 :46 :43 +0200 (mar. 16 avril 2019) | 1 ligne

Modification des noms des slots (nom en verbe)

r39 | tvaira | 2019-04-06 10 :07 :24 +0200 (sam. 06 avril 2019) | 1 ligne

Ajout de la documentation pour le tag 0.1

r38 | mandreo | 2019-04-05 15 :01 :54 +0200 (ven. 05 avril 2019) | 1 ligne

Modification du squelette de Chrono-Cross, modification des noms, changement dans la lecture de la trame : emit avec arrivée

r37 | mandreo | 2019-04-05 09 :49 :56 +0200 (ven. 05 avril 2019) | 1 ligne

création du tag 1.0

r36 | mandreo | 2019-04-04 15 :31 :41 +0200 (jeu. 04 avril 2019) | 1 ligne

Zone de numéro dossard opérationnelle, doxygene modifier sur les slots et ajouter sur les nouvelles méthodes

r35 | mandreo | 2019-04-04 12 :12 :37 +0200 (jeu. 04 avril 2019) | 1 ligne

Connection entre le TAG et l'IHM terminé. Etat des LED opérarionnelle. Mise à jour du classement avec les arrivées.

r34 | mandreo | 2019-04-03 13 :15 :37 +0200 (mer. 03 avril 2019) | 1 ligne

ajustement IHM, connect des signaux et des slots

r33 | tvaira | 2019-04-01 11 :27 :39 +0200 (lun. 01 avril 2019) | 1 ligne

Controle Qualité Revue 2

r32 | mandreo | 2019-03-30 19 :42 :56 +0100 (sam. 30 mars 2019) | 1 ligne

Ajout du dossier Image avec le logo du projet mais aussi des photos pour l'IHM. Modification de l'IHM et ajout d'une page Qinput pour le numero de dossard quand quelqu'un passe la ligne d'arrivée

r31 | mandreo | 2019-03-29 11 :46 :41 +0100 (ven. 29 mars 2019) | 1 ligne

Ajout du projet test-mo-hl975 et modification du doxyfile

r30 | mandreo | 2019-03-29 11 :45 :15 +0100 (ven. 29 mars 2019) | 1 ligne

Ajout de la méthode lecture tram opérationnelle

r29 | sturlin | 2019-03-29 11 :43 :00 +0100 (ven. 29 mars 2019) | 1 ligne

ajout dy doxygen

r28 | sturlin | 2019-03-29 11 :40 :27 +0100 (ven. 29 mars 2019) | 1 ligne

ajoute des commentaires dowigen

r27 | sturlin | 2019-03-29 10 :47 :54 +0100 (ven. 29 mars 2019) | 1 ligne

création des associations et agrégations

r26 | sturlin | 2019-03-29 10 :38 :50 +0100 (ven. 29 mars 2019) | 1 ligne

création des classes coureur et resultat plus ajouts des attributs dans résultats cross

r25 | mandreo | 2019-03-28 22 :14 :09 +0100 (jeu. 28 mars 2019) | 1 ligne

Configuration définitive du port. Base pour la réception de trame^C

r24 | mandreo | 2019-03-28 18 :21 :48 +0100 (jeu. 28 mars 2019) | 1 ligne

Paramétrage des boutons demarrer, arreter et synchroniser. Communication entre l'IHM et le TAG HEUER effectuée. Lancer une course avec chrono maintenant operationnel.

r23 | mandreo | 2019-03-27 13 :42 :46 +0100 (mer. 27 mars 2019) | 1 ligne

Remplacement du classement dans l'ihm par une zone de texte. Ajout de la configuration du port

r22 | mandreo | 2019-03-22 10 :47 :58 +0100 (ven. 22 mars 2019) | 1 ligne

Modification de l'IHM, affichage opérationnel

r21 | mandreo | 2019-03-21 16 :40 :22 +0100 (jeu. 21 mars 2019) | 1 ligne

Ajout de la classe chrono, ajout des premiers widgets de l'IHM

r20 | mandreo | 2019-03-20 12 :07 :16 +0100 (mer. 20 mars 2019) | 1 ligne

Ajout des méthodes getNom getHeureDepart getDistance. Suppression des méthodes SQL inutiles. Ajout des attributs de la classe Course

r19 | tvaira | 2019-03-17 06 :58 :09 +0100 (dim. 17 mars 2019) | 1 ligne

Modifications pour Doxygen

r18 | tvaira | 2019-03-17 06 :55 :10 +0100 (dim. 17 mars 2019) | 1 ligne

Modifications pour Doxygen

r17 | tvaira | 2019-03-16 18 :03 :43 +0100 (sam. 16 mars 2019) | 1 ligne

Mise à jour de la classe BaseDeDonnees

r16 | tvaira | 2019-03-16 14 :41 :05 +0100 (sam. 16 mars 2019) | 2 lignes

Ajout target.path dans le fichier de projet

r15 | tvaira | 2019-03-16 14 :39 :22 +0100 (sam. 16 mars 2019) | 1 ligne

Ajout des dossiers Resultats-Cross et Gestion-Cross

r14 | sturlin | 2019-03-15 12 :05 :55 +0100 (ven. 15 mars 2019) | 1 ligne

Suppression des fichiers inutiles

r13 | sturlin | 2019-03-15 12 :05 :11 +0100 (ven. 15 mars 2019) | 1 ligne

Rajout du doxygen du constructeur, du destructeur des classes. Rajout de l'introduction du projet

r12 | mandreo | 2019-03-15 11 :00 :06 +0100 (ven. 15 mars 2019) | 1 ligne

Ajout de la méthode getTableau et formulerRequeteSelect

r11 | tvaira | 2019-03-15 05 :34 :41 +0100 (ven. 15 mars 2019) | 1 ligne

Modification Doxyfile

r10 | tvaira | 2019-03-15 05 :05 :29 +0100 (ven. 15 mars 2019) | 1 ligne

Initialisation de la documentation de code pour Chrono-Cross

r9 | mandreo | 2019-03-14 17 :23 :04 +0100 (jeu. 14 mars 2019) | 1 ligne

Ajout des méthodes : getRenseignement(), getEnregistrement(), getColonne.

r8 | mandreo | 2019-03-14 13 :53 :09 +0100 (jeu. 14 mars 2019) | 1 ligne

Ajout de la première widget QTextEdit

r7 | mandreo | 2019-03-14 13 :47 :02 +0100 (jeu. 14 mars 2019) | 1 ligne

Ajout de la classe base de donnée

r6 | mandreo | 2019-03-13 14 :28 :12 +0100 (mer. 13 mars 2019) | 1 ligne

Création de la classe IHM

r5 | sturlin | 2019-03-13 14 :01 :55 +0100 (mer. 13 mars 2019) | 1 ligne

Création du .pro, création de la classe Course et communication avec la raspberry Pi

r4 | tvaira | 2019-03-09 07 :54 :29 +0100 (sam. 09 mars 2019) | 1 ligne

Initialisation Doxygen et ajout fichier SQL

r3 | mandreo | 2019-02-07 16 :00 :59 +0100 (jeu. 07 févr. 2019) | 1 ligne

supression du repertoire doc pour les transferer dans le drive

r2 | mandreo | 2019-02-07 13 :39 :29 +0100 (jeu. 07 févr. 2019) | 1 ligne

Ajout d'un repertoire doc qui contient pour l'instant le TODO, le Changelog et le README

r1 | www-data | 2019-02-06 20 :07 :38 +0100 (mer. 06 févr. 2019) | 1 ligne

Creating initial repository structure

3 Installation

Fabrication

```
$ qmake
$ make
$ sudo make install
```

Base de données

Serveur MySQL

```
$ sudo apt install mysql-server
```

Chrono-Cross

```
$ mysql -uroot -ppassword -hlocalhost < Chrono-Cross-v0.1.sql
```

Compte organisateur

```
$ mysql -uroot -ppassword -hlocalhost
mysql> CREATE USER 'organisateur'@'%' IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON `Chrono-Cross`.* TO 'organisateur'@'%';
mysql> FLUSH PRIVILEGES;
```

Accès distant

```
$ sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf
bind-address 0.0.0.0
```

Redémarrage

```
$ sudo systemctl restart mysql.service
```

Vérification

```
$ sudo systemctl status mysql.service
```

4 README

Nom : Chrono-Cross

Système informatisé de gestion de courses à pied.

Numéro de version : 1.0

Auteurs

Suzie TURLIN suzie.turlin@gmail.com

Michaël ANDREO andreo.michael@outlook.fr

Objectif

Gérer des courses de cross chronométrées pour un établissement scolaire.

Ce système va servir au sein du collège Lasalle Avignon. Il doit pouvoir gérer une ou plusieurs courses de cross, chronométrer la course, afficher les classements des coureurs avec leur nom, prénom, classe, et les afficher sur un écran en temps réel.

Nom des logiciels composant le système

- Chrono-Cross : Chronomètre les courses et classe les coureurs à l'arrivée (PC Course)
- Gestion-Cross : Gère les manifestations, les courses et les inscriptions des coureurs (PC Course)
- Resultats-Cross : Affiche en temps-réel le classement à l'arrivée d'une course (Raspberry Pi + Écran)

Dépôt SVN

https://svn.riouxsvn.com/chrono-cross

Recette IR

- Étudiant 3 (IR) : TURLIN Suzie suzie.turlin@gmail.com
  - la création d'une manifestation est possible
  - la création des courses pour une manifestation est possible
  - l'affichage des informations pendant une course est fonctionnel
  - l'affichage du classement d'une course est fonctionnel
  - une impression des résultats est possible
- Étudiant 4 (IR) : ANDREO Michaël andreo.michael@outlook.fr
  - l'inscription des coureurs est possible
  - les associations coureurs/transpondeurs sont stockées dans la base de données
  - les temps d'arrivée et le classement sont affichés sur l'écran
  - les temps d'arrivée et le classement sont stockées dans la base de données
  - le démarrage d'une course est possible

Base de données MySQL

```
--
-- Base de données: `Chrono-Cross`
--

DROP DATABASE `Chrono-Cross`;

CREATE DATABASE IF NOT EXISTS `Chrono-Cross`;

USE `Chrono-Cross`;

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

--
-- Structure de la table `Categorie`
--

CREATE TABLE IF NOT EXISTS `Categorie` (
  `idCategorie` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  `Sexe` enum('M','F') NOT NULL,
  PRIMARY KEY (`idCategorie`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

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

INSERT INTO `Categorie` (`Nom`, `Sexe`) VALUES
('M13', 'M'),
('M13', 'F'),
('M15', 'M'),
('M15', 'F'),
('M17', 'M'),
('M17', 'F');

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

--
-- Structure de la table `Classe`
--

CREATE TABLE IF NOT EXISTS `Classe` (
  `idClasse` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(64) NOT NULL,
  `Numero` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`idClasse`),
  CONSTRAINT Unique_Classe UNIQUE (`Nom`,`Numero`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

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

INSERT INTO `Classe` (`Nom`,`Numero`) VALUES
('6E', '1'),
('6E', '2'),
('6E', '3'),
('5E', '1'),
('5E', '2'),
('5E', '3'),
('4E', '1'),
('4E', '2'),
('4E', '3'),
('3E', '1'),
('3E', '2'),
('3E', '3');

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

--
-- Structure de la table `Coureur`
--

CREATE TABLE IF NOT EXISTS `Coureur` (
  `idCoureur` int(11) NOT NULL AUTO_INCREMENT,
  `idCategorie` int(11) NOT NULL,
  `idClasse` int(11) NOT NULL,
  `INE` varchar(11) NOT NULL,
  `Nom` varchar(64) NOT NULL,
  `Prenom` varchar(64) NOT NULL,
  `DateNaissance` date NOT NULL,
  `Sexe` enum('M','F') NOT NULL,
  PRIMARY KEY (`idCoureur`),
  CONSTRAINT Unique_Coureur UNIQUE (`INE`),
  CONSTRAINT Coureur_fk_1 FOREIGN KEY (`idCategorie`) REFERENCES Categorie(`idCategorie`) ON DELETE
      CASCADE,
  CONSTRAINT Coureur_fk_2 FOREIGN KEY (`idClasse`) REFERENCES Classe(`idClasse`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789AA','PERRICHON','Julia','2004-04-16','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
```

```
       VALUES('4','7','123456789BB','MOUTARD','Camille','2004-01-08','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789CC','MOLIST','Lucille','2004-09-26','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789DD','RIES','Clementine','2004-06-16','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789EE','LAMOUREUX','Felicia','2004-01-15','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789FF','STEY','Pauline','2004-05-01','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789GG','BIRE-HESLOUIS','Maele','2004-02-27','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789HH','BODIN','Alexia','2004-10-29','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789II','TERREC','Maelle','2004-11-05','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789JJ','FORNES','Marie','2004-02-01','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789KK','WINTREBERT','Pauline','2004-12-04','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789LL','GOURLET','Romane','2004-03-10','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789MM','VINCENT','Ines','2004-04-01','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','123456789NN','DUTOT','Camille','2004-09-13','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('4','7','1234567890O','PREVOST','Emmie','2004-01-02','F');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891AA','PELIOT','Julien','2004-02-16','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891BB','MOULARD','Charles','2004-03-08','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891CC','DURAND','Lucien','2004-10-26','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891DD','RIOUX','Clement','2004-07-07','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891EE','LAMOUR','Felicien','2004-08-24','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891FF','STEY','Paul','2004-06-25','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891GG','HERROUIS','Maurice','2004-03-09','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891HH','BOTIN','Alexis','2004-11-28','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891II','TORRES','Marcel','2004-12-07','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891JJ','FURLES','Mario','2004-08-01','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891KK','INTERBERT','Alexandre','2004-07-01','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891LL','GOUSSET','Romain','2004-04-17','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891MM','VINEAU','Thomas','2004-05-18','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','234567891NN','PIGNON','Jean','2004-10-20','M');
INSERT INTO `Coureur` (`idCategorie`,`idClasse`,`INE`,`Nom`,`Prenom`,`DateNaissance`,`Sexe`)
       VALUES('3','7','2345678910O','CLEVOST','Emile','2004-02-21','M');
-- --------------------------------------------------------

--
-- Structure de la table `Manifestation`
--

CREATE TABLE IF NOT EXISTS `Manifestation` (
  `idManifestation` int(11) NOT NULL AUTO_INCREMENT,
  `Nom` varchar(255) NOT NULL,
  `Date` date NOT NULL,
  PRIMARY KEY (`idManifestation`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Manifestation` (`Nom`,`Date`) VALUES('Cross College 2019','2019:04:30');
INSERT INTO `Manifestation` (`Nom`,`Date`) VALUES('Cross Lycee 2019','2019:04:30');

--
-- Structure de la table `Course`
--

CREATE TABLE IF NOT EXISTS `Course` (
  `idCourse` int(11) NOT NULL AUTO_INCREMENT,
  `idManifestation` int(11) NOT NULL,
  `Nom` varchar(255) NOT NULL,
  `Distance` int(11) NOT NULL,
  `HeureDepart` time NOT NULL,
  `Etat` enum('ACourir','Prete','EnCours','Arretee','Terminee') NOT NULL DEFAULT 'ACourir',
  PRIMARY KEY (`idCourse`),
  CONSTRAINT Course_fk_1 FOREIGN KEY (`idManifestation`) REFERENCES Manifestation(`idManifestation`) ON
       DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Course` (`idManifestation`, `Nom`, `Distance`, `HeureDepart`, `Etat`) VALUES('1','Cross M15
       F','3500','13:00:00', 'ACourir');
```

```
INSERT INTO `Course` (`idManifestation`, `Nom`, `Distance`, `HeureDepart`, `Etat`) VALUES('1','Cross M15
       M','5000','14:00:00', 'ACourir');
INSERT INTO `Course` (`idManifestation`, `Nom`, `Distance`, `HeureDepart`, `Etat`) VALUES('2','Cross M17
       F','3500','15:00:00', 'ACourir');
INSERT INTO `Course` (`idManifestation`, `Nom`, `Distance`, `HeureDepart`, `Etat`) VALUES('2','Cross M17
       M','5000','16:00:00', 'ACourir');

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

--
-- Structure de la table `Inscrit`
--

CREATE TABLE IF NOT EXISTS `Inscrit` (
  `idInscrit` int(11) NOT NULL AUTO_INCREMENT,
  `idCoureur` int(11) NOT NULL,
  `idCourse` int(11) NOT NULL,
  `NumeroDossard` varchar(16) NOT NULL,
  PRIMARY KEY (`idInscrit`),
  CONSTRAINT Unique_CoureurCourse UNIQUE (`idCoureur`,`idCourse`),
  CONSTRAINT Inscrit_fk_1 FOREIGN KEY (`idCoureur`) REFERENCES Coureur(`idCoureur`) ON DELETE CASCADE,
  CONSTRAINT Inscrit_fk_2 FOREIGN KEY (`idCourse`) REFERENCES Course(`idCourse`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('1','1','101');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('2','1','102');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('3','1','103');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('4','1','104');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('5','1','105');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('6','1','106');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('7','1','107');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('8','1','108');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('9','1','109');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('10','1','110');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('11','1','111');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('12','1','112');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('13','1','113');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('14','1','114');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('15','1','115');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('16','2','201');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('17','2','202');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('18','2','203');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('19','2','204');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('20','2','205');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('21','2','206');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('22','2','207');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('23','2','208');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('24','2','209');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('25','2','210');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('26','2','211');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('27','2','212');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('28','2','213');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('29','2','214');
INSERT INTO `Inscrit` (`idCoureur`,`idCourse`,`NumeroDossard`) VALUES('30','2','215');

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

--
-- Structure de la table `Arrivee`
--

CREATE TABLE IF NOT EXISTS `Arrivee` (
  `idArrivee` int(11) NOT NULL AUTO_INCREMENT,
  `idInscrit` int(11) NOT NULL,
  `Temps` time NOT NULL,
  PRIMARY KEY (`idArrivee`),
  CONSTRAINT Arrivee_fk_1 FOREIGN KEY (`idInscrit`) REFERENCES Inscrit(`idInscrit`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

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

INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('1','00:12:56');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('2','00:13:44');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('3','00:14:11');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('4','00:14:52');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('5','00:15:17');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('6','00:16:30');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('2','00:17:50');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('8','00:18:54');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('9','00:21:06');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('10','00:21:46');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('11','00:22:08');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('12','00:22:25');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('13','00:23:01');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('14','00:25:06');
INSERT INTO `Arrivee` (`idInscrit`,`Temps`) VALUES('15','00:29:12');
```

5 A propos

Auteur

Suzie TURLIN suzie.turlin@gmail.com
Michaël ANDREO andreo.michael@outlook.fr

Version

1.0

Date

2019

6 Licence GPL

7 Liste des choses à faire

Membre Course : :getListeCourses (QString manifestation)
verifier date

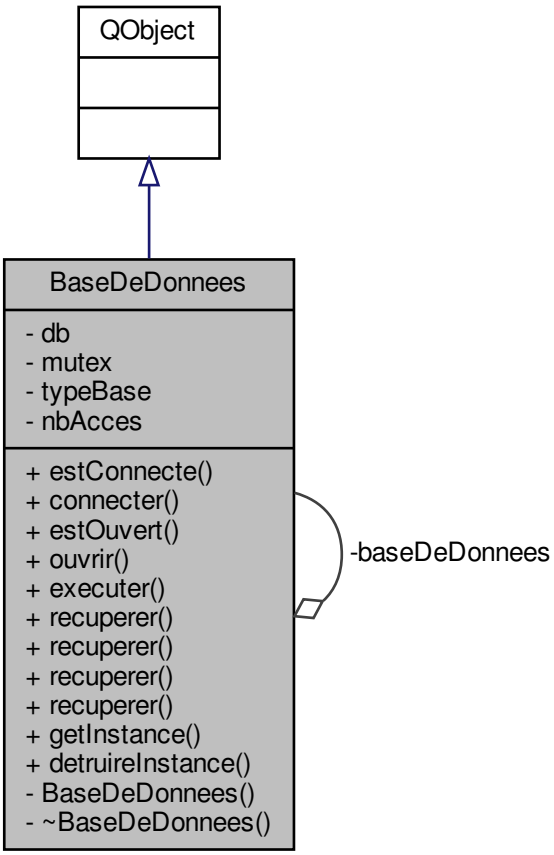
Membre IHMResultatsCross : :IHMResultatsCross (QWidget ∗parent=nullptr)
Définir le contenu de l'IHM

8 Documentation des classes

8.1 Référence de la classe BaseDeDonnees

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

Graphe de collaboration de BaseDeDonnees :



Fonctions membres publiques

- bool estConnecte ()
- bool connecter (QString nomBase, QString username=BDD_USERNAME, QString password=BDD_PASSWORD, QString serveur=BDD_HOSTNAME)
- bool estOuvert ()
- bool ouvrir (QString fichierBase)
- bool executer (QString requete)
- bool recuperer (QString requete, QString &donnees)
- bool recuperer (QString requete, QStringList &donnees)
- bool recuperer (QString requete, QVector< QString > &donnees)
- bool recuperer (QString requete, QVector< QStringList > &donnees)

Fonctions membres publiques statiques

- static BaseDeDonnees * getInstance (QString type="QMYSQL")
- static void detruireInstance ()

Fonctions membres privées

- BaseDeDonnees (QString type)
- ~BaseDeDonnees ()

Attributs privés

- QSqlDatabase db
- QMutex mutex

Attributs privés statiques

- static BaseDeDonnees * baseDeDonnees = nullptr
- static QString typeBase = "QMYSQL"
- static int nbAcces = 0

8.1.1 Documentation des constructeurs et destructeur

8.1.1.1 BaseDeDonnees()

```
BaseDeDonnees::BaseDeDonnees (
            QString type )  [private]
```

Références db, et typeBase.

Référencé par getInstance().

```
00023 {
00024     #ifdef DEBUG_BASEDEDONNEES
00025     qDebug() << Q_FUNC_INFO << type;
00026     #endif
00027     db = QSqlDatabase::addDatabase(type);
00028     typeBase = type;
00029 }
```

8.1.1.2 ~BaseDeDonnees()

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

```
00032 {
00033     #ifdef DEBUG_BASEDEDONNEES
00034     qDebug() << Q_FUNC_INFO;
00035     #endif
00036 }
```

8.1.2 Documentation des fonctions membres

8.1.2.1 connecter()

```
bool BaseDeDonnees::connecter (
            QString nomBase,
            QString username = BDD_USERNAME,
            QString password = BDD_PASSWORD,
            QString serveur = BDD_HOSTNAME )
```

Références db, mutex, et typeBase.

Référencé par Coureur : :Coureur(), Course : :Course(), GestionBDD : :GestionBDD(), IHMResultatsCross : :IHMResultatsCross(), et Resultat : :Resultat().

```
00077 {
00078     if(typeBase != "QMYSQL")
00079         return false;
00080     QMutexLocker verrou(&mutex);
00081     if(!db.isOpen())
00082     {
00083        db.setHostName(serveur);
00084        db.setUserName(username);
00085        db.setPassword(password);
00086        db.setDatabaseName(nomBase);
00087
00088        qDebug() << Q_FUNC_INFO << db;
00089        #ifdef DEBUG_BASEDEDONNEES
00090        qDebug() << Q_FUNC_INFO;
00091        qDebug() << "HostName : " << db.hostName();
00092        qDebug() << "UserName : " << db.userName();
00093        qDebug() << "DatabaseName : " << db.databaseName();
00094        #endif
00095        if(db.open())
00096        {
00097            #ifdef DEBUG_BASEDEDONNEES
00098            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Connexion réussie à %1").arg(db.hostName());
00099            #endif
00100            return true;
00101        }
00102        else
00103        {
00104            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : impossible de se connecter à la base de
      données !");
00105            QMessageBox::critical(0, QString::fromUtf8("BaseDeDonnees"), QString::fromUtf8("Impossible de se
      connecter à la base de données !"));
00106            return false;
00107        }
00108     }
00109     else
00110         return true;
00111 }
```

8.1.2.2 detruireInstance()

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

Références baseDeDonnees, et nbAcces.

Référencé par Course : :~Course(), GestionBDD : :~GestionBDD(), IHMChronoCross : :~IHMChronoCross(), IHMGestionCross : :~IHMGestionCross(), IHMResultatsCross : :~IHMResultatsCross(), et Resultat : :~Resultat().

```
00052 {
00053     // instance ?
00054     if(baseDeDonnees != nullptr)
00055     {
00056         if(nbAcces > 0)
00057             nbAcces--;
00058         #ifdef DEBUG_BASEDEDONNEES
00059         qDebug() << Q_FUNC_INFO << "nbAcces restants" << nbAcces;
00060         #endif
00061         // dernier ?
00062         if(nbAcces == 0)
00063         {
00064             delete baseDeDonnees;
00065             baseDeDonnees = nullptr;
00066         }
00067     }
00068 }
```

8.1.2.3 estConnecte()

```
bool BaseDeDonnees::estConnecte ( )
```

Références db, et mutex.

Référencé par Coureur : :Coureur(), Course : :Course(), GestionBDD : :GestionBDD(), IHMResultatsCross : :IHMResultatsCross(), et Resultat : :Resultat().

```
00071 {
00072     QMutexLocker verrou(&mutex);
00073     return db.isOpen();
00074 }
```

8.1.2.4 estOuvert()

```
bool BaseDeDonnees::estOuvert ( )
```

Références db, et mutex.

```
00114 {
00115     QMutexLocker verrou(&mutex);
00116     return db.isOpen();
00117 }
```

8.1.2.5 executer()

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

Références db, et mutex.

Référencé par Course : :ajouteArriveeBDD(), GestionBDD : :ajouterNouveauCoureur(), GestionBDD : :ajouterNouvelInscrit(), GestionBDD : :modifierCoureur(), Course : :setEtat(), et GestionBDD : :supprimerCoureur().

```
00152 {
00153     QMutexLocker verrou(&mutex);
00154     QSqlQuery r;
00155     bool retour;
00156
00157     if(db.isOpen())
00158     {
00159         if(requete.contains("UPDATE") || requete.contains("INSERT") || requete.contains("DELETE"))
00160         {
00161             retour = r.exec(requete);
00162             #ifdef DEBUG_BASEDEDONNEES
00163             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00164             #endif
00165             if(retour)
00166             {
00167                 return true;
00168             }
00169             else
00170             {
00171                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00172                 return false;
00173             }
00174         }
00175         else
00176         {
00177             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00178             return false;
00179         }
00180     }
00181     else
00182         return false;
00183
00184 }
```

8.1.2.6 getInstance()

```
BaseDeDonnees * BaseDeDonnees::getInstance (
            QString type = "QMYSQL" )  [static]
```

Références BaseDeDonnees(), baseDeDonnees, et nbAcces.

Référencé par Coureur : :Coureur(), Course : :Course(), GestionBDD : :GestionBDD(), IHMResultatsCross : :IHMResultatsCross(), et Resultat : :Resultat().

```
00039 {
00040     if(baseDeDonnees == nullptr)
00041         baseDeDonnees = new BaseDeDonnees(type);
00042
00043     nbAcces++;
00044     #ifdef DEBUG_BASEDEDONNEES
00045     qDebug() << Q_FUNC_INFO << "nbAcces" << nbAcces;
00046     #endif
00047
00048     return baseDeDonnees;
00049 }
```

8.1.2.7 ouvrir()

```
bool BaseDeDonnees::ouvrir (
            QString fichierBase )
```

Références db, mutex, et typeBase.

```
00120 {
00121     if(typeBase != "QSQLITE")
00122         return false;
00123     QMutexLocker verrou(&mutex);
00124     if(!db.isOpen())
00125     {
00126        db.setDatabaseName(fichierBase);
00127
00128        #ifdef DEBUG_BASEDEDONNEES
00129        qDebug() << Q_FUNC_INFO << db.databaseName();
00130        #endif
00131        if(db.open())
00132        {
00133            #ifdef DEBUG_BASEDEDONNEES
00134            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Ouvertir réussie à %1").arg(
      db.databaseName());
00135            #endif
00136
00137            return true;
00138        }
00139        else
00140        {
00141            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : impossible d'ouvrir la base de données !"
      );
00142            QMessageBox::critical(0, QString::fromUtf8("BaseDeDonnees"), QString::fromUtf8("Impossible
       d'ouvrir la base de données !"));
00143            return false;
00144        }
00145     }
00146     else
00147         return true;
00148 }
```

8.1.2.8 recuperer() [1/4]

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

Références db, et mutex.

Référencé par Course : :ajouteArriveeBDD(), GestionBDD : :ajouterNouveauCoureur(), GestionBDD : :ajouterNouvelInscrit(), Course : :getDistance(), Course : :getHeure(), Course : :getInformationCoureur(), Course : :getListeCourses(), Course : :getListeManifestations(), Course : :getNbArrivee(), Course : :getNbInscrit(), Course : :getNomCourse(), GestionBDD : :modifierCoureur(), GestionBDD : :recupererCategoriesCreation(), GestionBDD : :recupererCatergorieCoureur(), GestionBDD : :recupererClasseCoureur(), GestionBDD : :recupererClassesCreation(), GestionBDD : :recupererInformation(), GestionBDD : :recupererListeCoureursInscrit(), GestionBDD : :recupererListeCoursesGestion(), GestionBDD : :recupererListeCoursesInscription(), GestionBDD : :recupererListeManifestationsInscription(), GestionBDD : :recupererTableBDD(), Course : :setIdCourse(), GestionBDD : :supprimerCoureur(), GestionBDD : :verifierDossard(), Course : :verifierDossard(), et GestionBDD : :verifierInformation().

```
00190 {
00191     QMutexLocker verrou(&mutex);
00192     QSqlQuery r;
00193     bool retour;
00194
00195     if(db.isOpen())
00196     {
00197         if(requete.contains("SELECT"))
00198         {
00199             retour = r.exec(requete);
00200             #ifdef DEBUG_BASEDEDONNEES
00201             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
      QString::number(retour)).arg(requete);
00202             #endif
00203             if(retour)
00204             {
00205                 // on se positionne sur l'enregistrement
00206                 r.first();
00207
00208                 // on vérifie l'état de l'enregistrement retourné
00209                 if(!r.isValid())
00210                 {
00211                     #ifdef DEBUG_BASEDEDONNEES
00212                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Résultat non valide !");
00213                     #endif
00214                     return false;
00215                 }
00216
00217                 // on récupère sous forme de QString la valeur du champ
00218                 if(r.isNull(0))
00219                 {
00220                     #ifdef DEBUG_BASEDEDONNEES
00221                     qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucun résultat !");
00222                     #endif
00223                     return false;
00224                 }
00225                 donnees = r.value(0).toString();
00226                 #ifdef DEBUG_BASEDEDONNEES
00227                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00228                 #endif
00229                 return true;
00230             }
00231             else
00232             {
00233                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00234                 return false;
00235             }
00236         }
00237         else
00238         {
00239             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00240             return false;
00241         }
00242     }
00243     else
00244         return false;
00245 }
```

8.1.2.9 recuperer() [2/4]

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

Références db, et mutex.

```
00251 {
00252     QMutexLocker verrou(&mutex);
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("<BaseDeDonnees::recuperer(QString, QStringList)> 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 }
```

8.1.2.10 recuperer() [3/4]

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

Références db, et mutex.

```
00308 {
00309     QMutexLocker verrou(&mutex);
00310     QSqlQuery r;
00311     bool retour;
00312     QString data;
00313 
00314     if(db.isOpen())
00315     {
00316         if(requete.contains("SELECT"))
00317         {
00318             retour = r.exec(requete);
00319             #ifdef DEBUG_BASEDEDONNEES
00320             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
    QString::number(retour)).arg(requete);
00321             #endif
00322             if(retour)
00323             {
00324                 // pour chaque enregistrement
00325                 while ( r.next() )
00326                 {
00327                     // on récupère sous forme de QString la valeur du champs sélectionné
00328                     data = r.value(0).toString();
00329 
00330                     #ifdef DEBUG_BASEDEDONNEES
00331                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00332                     #endif
00333 
00334                     // on stocke l'enregistrement dans le QVector
00335                     donnees.push_back(data);
00336                 }
00337                 #ifdef DEBUG_BASEDEDONNEES
00338                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00339                 #endif
00340                 return true;
00341             }
00342             else
00343             {
00344                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
    lastError().text()).arg(requete);
00345                 return false;
00346             }
00347         }
00348         else
00349         {
00350             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
    );
00351             return false;
00352         }
00353     }
00354     else
00355         return false;
00356 }
```

8.1.2.11 recuperer() [4/4]

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

Références db, et mutex.

```
00362 {
00363     QMutexLocker verrou(&mutex);
00364     QSqlQuery r;
00365     bool retour;
00366     QStringList data;
00367 
00368     if(db.isOpen())
00369     {
00370         if(requete.contains("SELECT"))
00371         {
00372             retour = r.exec(requete);
00373             #ifdef DEBUG_BASEDEDONNEES
00374             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Retour %1 pour la requete : %2").arg(
    QString::number(retour)).arg(requete);
00375             #endif
00376             if(retour)
00377             {
00378                 // pour chaque enregistrement
00379                 while ( r.next() )
```

```
00380                 {
00381                     // on récupère sous forme de QString la valeur de tous les champs sélectionnés
00382                     // et on les stocke dans une liste de QString
00383                     for(int i=0;i<r.record().count();i++)
00384                         data << r.value(i).toString();
00385 
00386                     #ifdef DEBUG_BASEDEDONNEES
00387                     //qDebug() << Q_FUNC_INFO << "Enregistrement -> " << data;
00388                     /*for(int i=0;i<r.record().count();i++)
00389                         qDebug() << r.value(i).toString();*/
00390                     #endif
00391 
00392                     // on stocke l'enregistrement dans le QVector
00393                     donnees.push_back(data);
00394 
00395                     // on efface la liste de QString pour le prochain enregistrement
00396                     data.clear();
00397                 }
00398                 #ifdef DEBUG_BASEDEDONNEES
00399                 qDebug() << Q_FUNC_INFO << "Enregistrement -> " << donnees;
00400                 #endif
00401                 return true;
00402             }
00403             else
00404             {
00405                 qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : %1 pour la requête %2").arg(r.
      lastError().text()).arg(requete);
00406                 return false;
00407             }
00408         }
00409         else
00410         {
00411             qDebug() << Q_FUNC_INFO << QString::fromUtf8("Erreur : requête %1 non autorisée !").arg(requete
      );
00412             return false;
00413         }
00414     }
00415     else
00416         return false;
00417 }
```

8.1.3 Documentation des données membres

8.1.3.1 baseDeDonnees

```
BaseDeDonnees * BaseDeDonnees::baseDeDonnees = nullptr  [static], [private]
```

Référencé par detruireInstance(), et getInstance().

8.1.3.2 db

```
QSqlDatabase BaseDeDonnees::db  [private]
```

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

8.1.3.3 mutex

```
QMutex BaseDeDonnees::mutex  [private]
```

Référencé par connecter(), estConnecte(), estOuvert(), executer(), ouvrir(), et recuperer().

8.1.3.4 nbAcces

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

Référencé par detruireInstance(), et getInstance().

8.1.3.5 typeBase

```
QString BaseDeDonnees::typeBase = "QMYSQL"  [static], [private]
```

Référencé par BaseDeDonnees(), connecter(), et ouvrir().

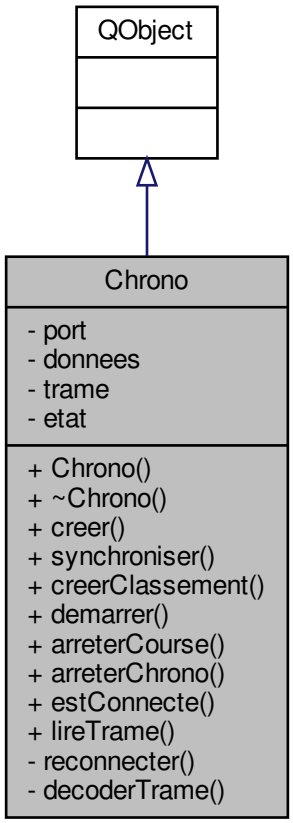
La documentation de cette classe a été générée à partir des fichiers suivants :
- basededonnees.h
- basededonnees.cpp

8.2 Référence de la classe Chrono

Déclaration de la classe Chrono.

```
#include <chrono.h>
```

Graphe de collaboration de Chrono :



Connecteurs publics

- void lireTrame ()
  Méthode lireTrame() de la classe Chrono.

Signaux

- void chronoCreer ()
- void chronoSynchroniser ()
- void classementCree ()
- void chronoLance ()
- void courseArretee ()
- void chronoArrete ()
- void chronoRecommence ()
- void nouvelleArrivee (QString tempsArrivee)

Fonctions membres publiques

- Chrono (QObject *parent=nullptr)
  Constructeur de la classe Chrono.
- ~Chrono ()
  Destructeur de la fenêtre principale.
- void creer ()
  Méthode creer() de la classe Chrono.
- bool synchroniser ()
  Méthode synchroniser() de la classe Chrono.
- bool creerClassement ()
  Méthode creerClassement() de la classe Chrono.
- bool demarrer ()
  Méthode demarrer() de la classe Chrono.
- bool arreterCourse ()
  Méthode arreterCourse() de la classe Chrono.
- bool arreterChrono ()
  Méthode arreterChrono() de la classe Chrono.
- bool estConnecte ()
  Méthode estConnecte() de la classe Chrono.

Fonctions membres privées

- bool reconnecter ()
  Méthode reconnecter() de la classe Chrono.
- void decoderTrame ()
  Méthode decoderTrame() de la classe Chrono.

Attributs privés

- QSerialPort * port
  Port RS232 du TAG HEUER.
- QByteArray donnees
  Données brutes d'une trame THCOM08.
- QString trame
  Trame THCOM08.
- unsigned int etat
  État du chronomètre.

8.2.1 Description détaillée

Déclaration de la classe Chrono.

Auteur

Michael Andréo

Version

1.1

8.2.2 Documentation des constructeurs et destructeur

8.2.2.1 Chrono()

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

Constructeur de la classe Chrono.

Paramètres

parent	QObject Adresse de l'objet Qt parent (0 = pas de parent car c'est la fenêtre principale)

Références etat, PORT, et port.

```
00020                               : QObject (parent)
00021 {
00022     port = new QSerialPort(PORT);
00023     port->setBaudRate(QSerialPort::Baud9600);
00024     port->setFlowControl(QSerialPort::NoFlowControl);
00025     port->setParity(QSerialPort::NoParity);
00026     port->setDataBits(QSerialPort::Data8);
00027     port->setStopBits(QSerialPort::OneStop);
00028     port->open(QIODevice::ReadWrite);
00029
00030     etat = 0;
00031
00032     qDebug() << Q_FUNC_INFO << "Etat port : " << port->isOpen();
00033 }
```

8.2.2.2 ~Chrono()

```
Chrono::~Chrono ( )
```

Destructeur de la fenêtre principale.

Références port.

```
00040 {
00041     port->close();
00042 }
```

8.2.3 Documentation des fonctions membres

8.2.3.1 arreterChrono()

```
bool Chrono::arreterChrono ( )
```

Méthode arreterChrono() de la classe Chrono.

Si le port est ouvert on écrit la trame MODEND

Renvoie

bool

Références MODEND, port, et trame.

Référencé par Course : :arreterChrono().

```
00282 {
00283     if(!port->isOpen())
00284         return false;
00285     qDebug() << Q_FUNC_INFO << "MODEND";
00286     trame = MODEND;
00287     qDebug() << Q_FUNC_INFO << "trame" << trame;
00288     int nb = int(port->write(trame.toLatin1()));
00289     if(nb > 0)
00290         return true;
00291     else
00292         return false;
00293 }
```

8.2.3.2 arreterCourse()

```
bool Chrono::arreterCourse ( )
```

Méthode arreterCourse() de la classe Chrono.

Si le port est ouvert on écrit la trame CLOSERUN

Références CLOSERUN, port, et trame.

Référencé par Course : :arreterClassement().

```
00262 {
00263     if(!port->isOpen())
00264         return false;
00265     qDebug() << Q_FUNC_INFO << "CLOSERUN";
00266     trame = CLOSERUN;
00267     qDebug() << Q_FUNC_INFO << "trame" << trame;
00268     int nb = int(port->write(trame.toLatin1()));
00269     if(nb > 0)
00270         return true;
00271     else
00272         return false;
00273 }
```

8.2.3.3 chronoArrete

```
void Chrono::chronoArrete ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.4 chronoCreer

```
void Chrono::chronoCreer ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.5 chronoLance

```
void Chrono::chronoLance ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.6 chronoRecommence

```
void Chrono::chronoRecommence ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.7 chronoSynchroniser

```
void Chrono::chronoSynchroniser ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.8 classementCree

```
void Chrono::classementCree ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.9 courseArretee

```
void Chrono::courseArretee ( )  [signal]
```

Référencé par decoderTrame().

8.2.3.10 creer()

```
void Chrono::creer ( )
```

Méthode creer() de la classe Chrono.

Si le port est ouvert on écrit la trame MODECLOCK

Références lireTrame(), MODECLOCK, port, et trame.

Référencé par Course : :creerChrono(), et reconnecter().

```
00189 {
00190     if(port->isOpen())
00191     {
00192         connect(port, SIGNAL(readyRead()), this, SLOT(lireTrame()));
00193         qDebug() << Q_FUNC_INFO << "MODECLOCK";
00194         trame = MODECLOCK;
00195         qDebug() << Q_FUNC_INFO << "trame émise" << trame;
00196         port->write(trame.toLatin1());
00197     }
00198 }
```

8.2.3.11 creerClassement()

```
bool Chrono::creerClassement ( )
```

Méthode creerClassement() de la classe Chrono.

Si le port est ouvert on écrit la trame NEWRUN

Références NEWRUN, port, et trame.

Référencé par Course : :aChronoSynchronise().

```
00225 {
00226     if(!port->isOpen())
00227         return false;
00228     qDebug() << Q_FUNC_INFO << "NEWRUN";
00229     trame = NEWRUN;
00230     int nb = int(port->write(trame.toLatin1()));
00231     if(nb > 0)
00232         return true;
00233     else
00234         return false;
00235 }
```

8.2.3.12 decoderTrame()

```
void Chrono::decoderTrame ( )  [private]
```

Méthode decoderTrame() de la classe Chrono.

Décode les trames émise du TAG HEUER

Références CHAMPS_TRAME_TEMPS, chronoArrete(), chronoCreer(), chronoLance(), chronoRecommence(), chronoSynchroniser(), classementCree(), courseArretee(), etat, ETAT_CLOSERUN, ETAT_MODECLOCK, ETAT_MODEND, ETAT_NEWRUN, ETAT_NEWSYNCHRO, ETAT_NONSYNCHRO, ETAT_STARTMANUALSYNCHRO, nouvelleArrivee(), reconnecter(), trame, TRAME_ACQUITTEMENT, TRAME_COURSE_TERMINEE, et TRAME_TEMPS.

Référencé par lireTrame().

```
00072 {
00073     /*
00074       Liste des trames reçues :
00075       - "AK X" acquittement avec : X = 'C' accepted, 'F' rejected, 'R' not supported
00076       - "TS 00:00:00 00/00/01" synchro
00077       - "&P 003 XX 90 00:00:00 0" paramètre 003 (demande du numéro de course) XX = le numéro de course
00078       - "TN 1 2     3.71300  365" temps
00079       - "CL XX" close course XX = le numéro de course terminée
00080      */
00081
00082     trame.remove("\r\n");
00083     qDebug() << Q_FUNC_INFO << "Trame after remove rn:\t" << trame;
00084
00085     QStringList champsTrame;
00086     if(trame.startsWith(TRAME_ACQUITTEMENT))
00087     {
00088         champsTrame = trame.split(" ", QString::SkipEmptyParts);
00089         qDebug() << Q_FUNC_INFO << "Trame acquitement :" << champsTrame[1];
00090
00091         if(champsTrame[1].startsWith("C"))
00092         {
00093             if(etat == ETAT_MODEND)
00094             {
00095                 etat = ETAT_NONSYNCHRO;
00096                 emit chronoRecommence();
00097             }
00098             //première trame d'acquittement
00099             if(etat == ETAT_NONSYNCHRO)
00100             {
00101                 etat = ETAT_MODECLOCK;
00102                 emit chronoCreer();
00103                 qDebug() << Q_FUNC_INFO << "AK\tETAT chrono creer! 1" << etat;
00104             }
00105             //deuxieme trame d'acquittement
00106             else if(etat == ETAT_MODECLOCK)
00107             {
00108                 etat = ETAT_NEWSYNCHRO;
00109                 emit chronoSynchroniser();
00110                 qDebug() << Q_FUNC_INFO << "AK\tETAT Mode clock 2" << etat;
00111             }
00112             //troisieme trame d'acquittement
00113             else if(etat == ETAT_NEWSYNCHRO)
00114             {
00115                 etat = ETAT_NEWRUN;
00116                 emit classementCree();
00117                 qDebug() << Q_FUNC_INFO << "ETAT NEW SYNCHR 3" << etat;
00118             }
00119             //quatrième trame d'acquittement
00120             else if(etat == ETAT_NEWRUN)
00121             {
00122                 etat = ETAT_STARTMANUALSYNCHRO;
00123                 emit chronoLance();
00124                 qDebug() << Q_FUNC_INFO << "ETAT NEWRUN 4" << etat;
00125             }
00126             //cinquième trame d'acquittement
00127             else if(etat == ETAT_STARTMANUALSYNCHRO)
00128             {
00129                 etat = ETAT_CLOSERUN;
00130                 emit courseArretee();
00131                 qDebug() << Q_FUNC_INFO << "ETAT CLOSERUN 5" << etat;
00132             }
00133             else if(etat == ETAT_CLOSERUN)
00134             {
00135                 etat = ETAT_MODEND;
00136                 emit chronoArrete();
00137                 qDebug() << Q_FUNC_INFO << "ETAT MODEND 6" << etat;
00138             }
```

```
00139         }
00140     }
00141     else if(trame.startsWith(TRAME_TEMPS))
00142     {
00143         /*
00144             Time (TN) : TN SSSS CC HH:MM:SS.FFFFF DDDDD
00145
00146         S = Sequential number (0 - 9999) -> ordre arrivée
00147         C = Channel number (1 - 99) in case of manual entry (M1 - M4) -> numéro cellule détection (1 =
      départ et 2 = arrivée)
00148         H = Hours (0 - 23)
00149         M = Minutes (0 - 59)
00150         S = Seconds (0 - 59)
00151         F = decimal part (0 - 99999)
00152         D = Days (0 - 32767) counting from 01.01.2000 -> non utilisé
00153
00154         "TN", "1", "1", "4.30700", "365 05FE"
00155         "TN         1  1        4.30700   365"
00156         */
00157
00158         champsTrame = trame.split(" ", QString::SkipEmptyParts);
00159         qDebug() << Q_FUNC_INFO << "Trame TN" << champsTrame;
00160         emit nouvelleArrivee(champsTrame[CHAMPS_TRAME_TEMPS]);
00161     }
00162     else if(trame.startsWith(TRAME_COURSE_TERMINEE))
00163     {
00164         /*
00165             Closing of a Run (CL) : CL RR
00166             R = Run number (1 - 99)
00167
00168             "CL 06  0115"
00169          */
00170         champsTrame = trame.split(" ", QString::SkipEmptyParts);
00171         qDebug() << Q_FUNC_INFO << "Trame CL" << champsTrame;
00172         QStringList champs = trame.split("\t");
00173         champsTrame = champs.at(0).split(" ", QString::SkipEmptyParts);
00174         qDebug() << Q_FUNC_INFO << "Run number" << champsTrame.at(1);
00175     }
00176     else if(etat == ETAT_NONSYNCHRO)
00177     {
00178         qDebug() << Q_FUNC_INFO << "Etat synchronisation : " << etat;
00179         this->reconnecter();
00180     }
00181 }
```

8.2.3.13 demarrer()

```
bool Chrono::demarrer ( )
```

Méthode demarrer() de la classe Chrono.

Si le port est ouvert on écrit la trame STARTMANUALSYNCHRO

Références port, STARTMANUALSYNCHRO, et trame.

Référencé par Course : :chronometrer().

```
00243 {
00244     if(!port->isOpen())
00245         return false;
00246     qDebug() << Q_FUNC_INFO << "STARTMANUALSYNCHRO";
00247     trame = STARTMANUALSYNCHRO;
00248     qDebug() << Q_FUNC_INFO << "trame" << trame;
00249     int nb = int(port->write(trame.toLatin1()));
00250     if(nb > 0)
00251         return true;
00252     else
00253         return false;
00254 }
```

8.2.3.14 estConnecte()

```
bool Chrono::estConnecte ( )
```

Méthode estConnecte() de la classe Chrono.

Retourne l'état de connexion avec le TAG HEUER

Références port.

Référencé par Course : :estChronometragePret().

```
00301 {
00302     port->waitForReadyRead(950);
00303     return port->isOpen();
00304 }
```

8.2.3.15 lireTrame

```
void Chrono::lireTrame ( )  [slot]
```

Méthode lireTrame() de la classe Chrono.

Lit ligne par ligne les trames du TAG HEUER

Références decoderTrame(), donnees, port, TIMEOUT, et trame.

Référencé par creer(), et reconnecter().

```
00312 {
00313     while(port->canReadLine())
00314     {
00315         donnees += port->readAll();
00316         usleep(TIMEOUT);
00317     }
00318
00319     if(donnees.contains("\r\n"))
00320     {
00321         trame = QString(donnees.data());
00322         decoderTrame();
00323         donnees.clear();
00324     }
00325 }
```

8.2.3.16 nouvelleArrivee

```
void Chrono::nouvelleArrivee (
            QString tempsArrivee )  [signal]
```

Référencé par decoderTrame().

8.2.3.17 reconnecter()

```
bool Chrono::reconnecter ( )  [private]
```

Méthode reconnecter() de la classe Chrono.

Reconnecter le TAG HEUER

Références creer(), lireTrame(), et port.

Référencé par decoderTrame().

```
00050 {
00051     if(port->isOpen())
00052     {
00053         disconnect(port, SIGNAL(readyRead()), this, SLOT(lireTrame()));
00054         port->close();
00055     }
00056     port->open(QIODevice::ReadWrite);
00057     qDebug() << Q_FUNC_INFO << "Etat port : " << port->isOpen();
00058     if(port->isOpen())
00059     {
00060         this->creer();
00061         return true;
00062     }
00063     return false;
00064 }
```

8.2.3.18 synchroniser()

```
bool Chrono::synchroniser ( )
```

Méthode synchroniser() de la classe Chrono.

Si le port est ouvert on écrit la trame NEWSYNCHRO

Références NEWSYNCHRO, port, TIMEOUT, et trame.

Référencé par Course : :preparerChrono().

```
00206 {
00207     if(!port->isOpen())
00208         return false;
00209     qDebug() << Q_FUNC_INFO << "NEWSYNCHRO";
00210     trame = NEWSYNCHRO;
00211     int nb = int(port->write(trame.toLatin1()));
00212     port->waitForReadyRead(TIMEOUT);
00213     if(nb > 0)
00214         return true;
00215     else
00216         return false;
00217 }
```

8.2.4 Documentation des données membres

8.2.4.1 donnees

```
QByteArray Chrono::donnees  [private]
```

Données brutes d'une trame THCOM08.

Référencé par lireTrame().

8.2.4.2 etat

```
unsigned int Chrono::etat  [private]
```

État du chronomètre.

Référencé par Chrono(), et decoderTrame().

8.2.4.3 port

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

Port RS232 du TAG HEUER.

Référencé par arreterChrono(), arreterCourse(), Chrono(), creer(), creerClassement(), demarrer(), estConnecte(), lireTrame(), reconnecter(), synchroniser(), et ~Chrono().

8.2.4.4 trame

```
QString Chrono::trame  [private]
```

Trame THCOM08.

Référencé par arreterChrono(), arreterCourse(), creer(), creerClassement(), decoderTrame(), demarrer(), lireTrame(), et synchroniser().

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

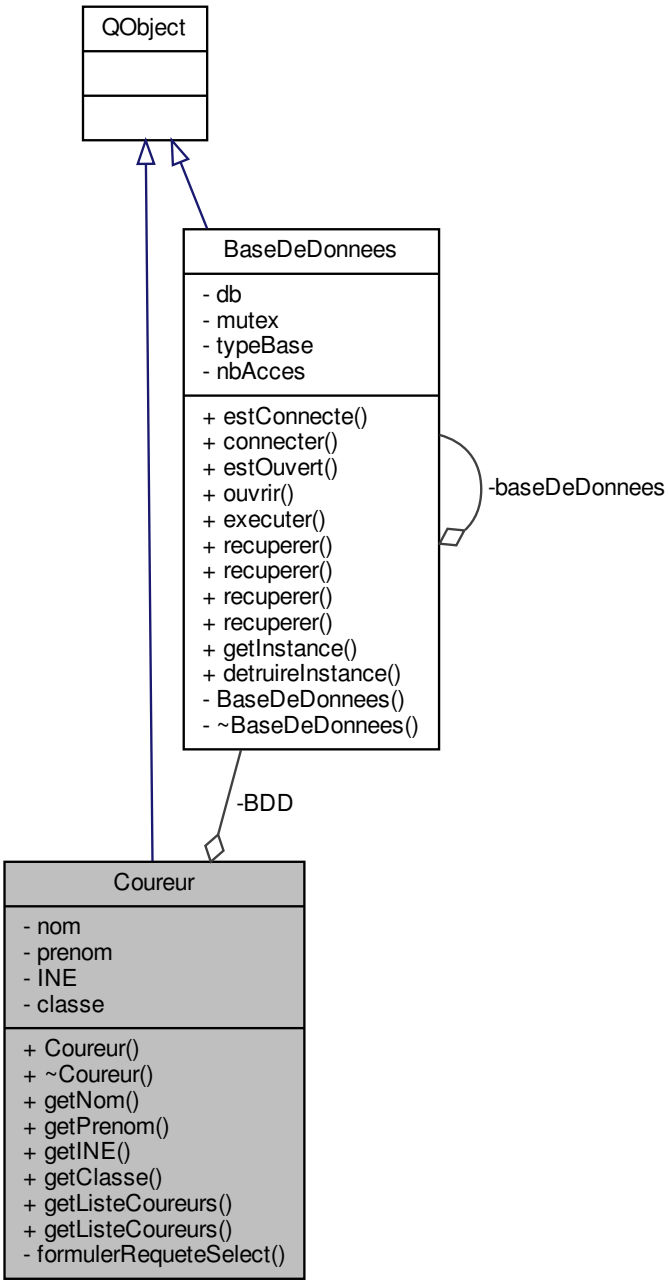
- chrono.h
- chrono.cpp

8.3 Référence de la classe Coureur

Gérer les coureurs.

```
#include <coureur.h>
```

Graphe de collaboration de Coureur :



Connecteurs publics

— QStringList getListeCoureurs (QString, QString, QString, QString)

Fonctions membres publiques

— Coureur (QObject ∗parent=nullptr)
Coureur.
— ∼Coureur ()

- QString getNom () const
- QString getPrenom () const
- QString getINE () const
- QString getClasse () const
- QVector< QString > getListeCoureurs (QString Coureur)

Fonctions membres privées

- QString formulerRequeteSelect (QString renseignements, QString sources, QString conditions)

Attributs privés

- QString nom
- QString prenom
- QString INE
- QString classe
- BaseDeDonnees * BDD
  agrégation BaseDeDonnees

8.3.1 Description détaillée

Gérer les coureurs.

Auteur

Suzie Turlin

Version

0.1

8.3.2 Documentation des constructeurs et destructeur

8.3.2.1 Coureur()

```
Coureur::Coureur (
            QObject * parent = nullptr )  [explicit]
```

Coureur.

Paramètres

parent	QObject Adresse de l'objet Qt parent

Références BDD, BaseDeDonnees : :connecter(), BaseDeDonnees : :estConnecte(), et BaseDeDonnees : :getInstance().

```
00025                                : QObject(parent)
00026 {
00027     //**TODO BASE DE DONNEE**//
00028
00029     BDD = BaseDeDonnees::getInstance();
00030     if(!BDD->estConnecte())
00031         BDD->connecter("Resultats-Cross");
00032     qDebug() << Q_FUNC_INFO << "Etat connexion BDD : " << BDD->estConnecte();
```

```
00033
00034
00035 }
```

8.3.2.2 ~Coureur()

```
Coureur::~Coureur ( )
```

```
00038 {
00039
00040 }
```

8.3.3 Documentation des fonctions membres

8.3.3.1 formulerRequeteSelect()

```
QString Coureur::formulerRequeteSelect (
            QString renseignements,
            QString sources,
            QString conditions )  [private]
```

```
00069 {
00070     QString requete = QString("SELECT %1 FROM %2 WHERE %3").arg(renseignements).arg(sources).arg(conditions
      );
00071     qDebug() << Q_FUNC_INFO << "Requête select : " << requete;
00072     return requete;
00073 }
```

8.3.3.2 getClasse()

```
QString Coureur::getClasse ( ) const
```

Références classe.

```
00053 {
00054     return classe;
00055 }
```

8.3.3.3 getINE()

```
QString Coureur::getINE ( ) const
```

Références INE.

```
00058 {
00059     return INE;
00060 }
```

8.3.3.4 getListeCoureurs() [1/2]

```
QVector<QString> Coureur::getListeCoureurs (
            QString Coureur )
```

8.3.3.5 getListeCoureurs [2/2]

```
QStringList Coureur::getListeCoureurs (
            QString ,
            QString ,
            QString ,
            QString  )  [slot]
```

8.3.3.6 getNom()

```
QString Coureur::getNom ( ) const
```

Références nom.

```
00043 {
00044     return nom;
00045 }
```

8.3.3.7 getPrenom()

```
QString Coureur::getPrenom ( ) const
```

Références prenom.

```
00048 {
00049     return prenom;
00050 }
```

8.3.4 Documentation des données membres

8.3.4.1 BDD

```
BaseDeDonnees* Coureur::BDD  [private]
```

agrégation BaseDeDonnees

Référencé par Coureur().

8.3.4.2 classe

```
QString Coureur::classe  [private]
```

Référencé par getClasse().

8.3.4.3 INE

```
QString Coureur::INE  [private]
```

Référencé par getINE().

8.3.4.4 nom

```
QString Coureur::nom  [private]
```

Référencé par getNom().

8.3.4.5 prenom

```
QString Coureur::prenom  [private]
```

Référencé par getPrenom().

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

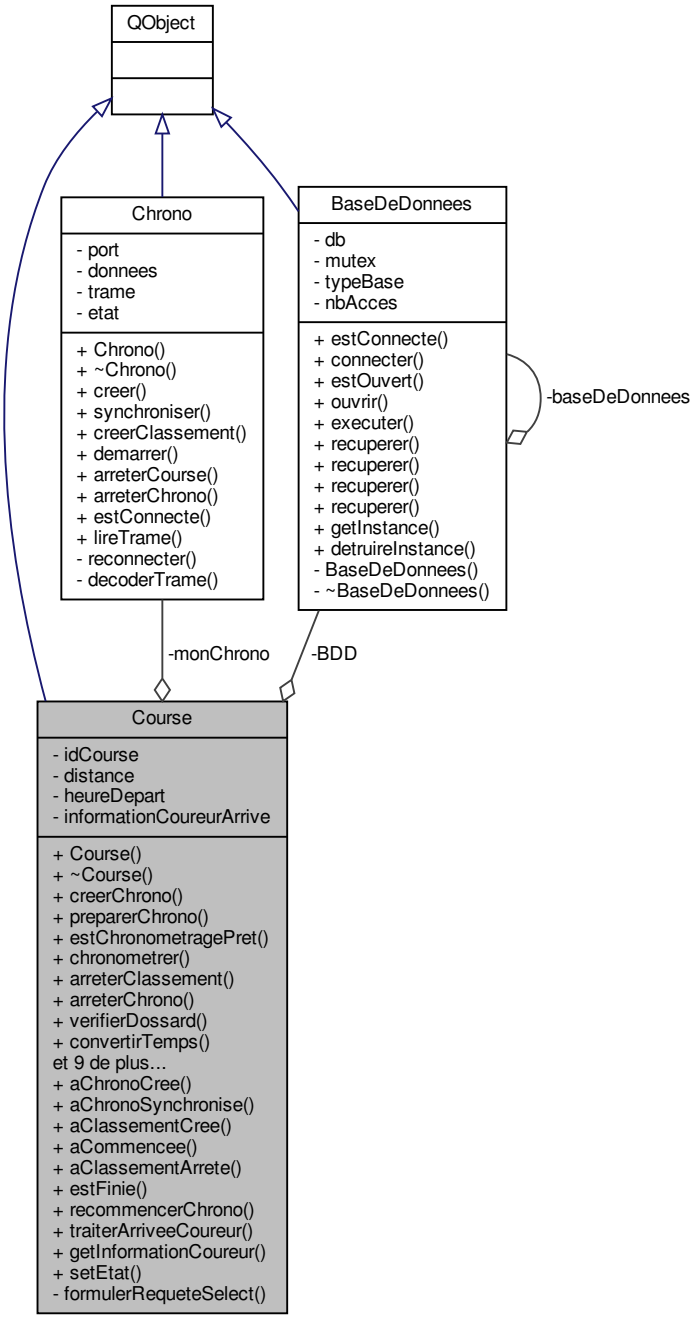
- coureur.h
- coureur.cpp

8.4 Référence de la classe Course

Déclaration de la classe Course.

```
#include <course.h>
```

Graphe de collaboration de Course :



Connecteurs publics

- void aChronoCree ()
  SLOT aChronoCree() de la classe Course.
- void aChronoSynchronise ()
  SLOT aChronoSynchronise() de la classe Course.
- void aClassementCree ()
  SLOT aClassementCree de la classe Course.
- void aCommencee ()
  SLOT aCommencee() de la classe Course.
- void aClassementArrete ()

SLOT aClassementArrete() de la classe Course.
- void estFinie ()
  SLOT estFinie() de la classe Course.
- void recommencerChrono ()
  SLOT recommencerChrono() de la classe Course.
- void traiterArriveeCoureur (QString tempsArrivee)
  SLOT traiterArriveeCoureur(QString tempsArrivee) de la classe Course.
- void getInformationCoureur (QString dossard, QString tempsARrivee)
  SLOT getInformationCoureur(QString dossard, QString tempsArrivee) de la classe Course.
- bool setEtat (QString etat)
  SLOT setEtat de la classe Course.

Signaux

- void chronoCreer ()
- void chronoCoursePret ()
- void courseCommence ()
- void classementArrete ()
- void courseFinie ()
- void chronoRecommence ()
- void nouveauTempsArrivee (QString tempsArrivee)
- void arriveeAjouteeBDD (QString dossard, QString tempsArrivee)
- void informationCoureurRecuperees (QStringList informationCoureur)

Fonctions membres publiques

- Course (QObject *parent=nullptr)
  Constructeur de la classe Course.
- ~Course ()
  Destructeur de la fenêtre principale.
- void creerChrono ()
  Méthode creerChrono() de la classe Course.
- void preparerChrono ()
  Méthode preparerChrono() de la classe Course.
- bool estChronometragePret ()
  Méthode estChronometragePret() de la classe Course.
- void chronometrer ()
  Méthode chronometrer de la classe Course.
- void arreterClassement ()
  Méthode arreterClassement() de la classe Course.
- void arreterChrono ()
  Méthode arreterChronomètre() de la classe Course.
- int verifierDossard (QString dossard)
  Méthode verifierDossard() de la classe Course.
- QString convertirTemps (QString tempsMS)
  Méthode convertirTemps() de la classe Course.
- int getNbInscrit (QString course)
  Méthode getNbInscrit de la classe Course.
- int getDistance (QString course)
  Méthode getDistance() de la classe Course.
- QString getHeure (QString course)
  Méthode getHeure() de la classe Course.
- int getNbArrivee ()
  Méthode getNbArrivee() de la classe Course.
- QString getNomCourse (QString dossard)
  Méthode getNomCourse() de la classe Course.
- void setIdCourse (QString nomCourse)
  Méthode setIdCourse() de la classe Course.
- void ajouteArriveeBDD (QString dossard, QString tempsArrivee)
  Méthode ajouteArriveeBDD() de la classe Course.
- QStringList getListeManifestations ()
  Méthode getListeManifestations() de la classe Course.
- QVector< QString > getListeCourses (QString manifestation)
  Méthode getListeCourses() de la classe Course.

Fonctions membres privées

- QString formulerRequeteSelect (QString renseignements, QString sources, QString conditions)
  Méthode formulerRequeteSelect de la classe Course.

Attributs privés

- BaseDeDonnees * BDD
  agrégation BaseDeDonnees
- Chrono * monChrono
  association Chrono
- QString idCourse
  Identifiant de la course.
- QString distance
  Distance de la course.
- QString heureDepart
  L'heure de départ de la course.
- QStringList informationCoureurArrive
  Informations d'un coure.

8.4.1 Description détaillée

Déclaration de la classe Course.

Auteur

Michael Andréo

Version

1.1

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 Course()

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

Constructeur de la classe Course.

Paramètres

parent	QObject Adresse de l'objet Qt parent (0 = pas de parent car c'est la fenêtre principale)

Références aChronoCree(), aChronoSynchronise(), aClassementArrete(), aClassementCree(), aCommencee(), arriveeAjouteeBD↩D(), BDD, chronoCreer(), chronoRecommence(), BaseDeDonnees : :connecter(), BaseDeDonnees : :estConnecte(), estFinie(), get↩InformationCoureur(), BaseDeDonnees : :getInstance(), monChrono, recommencerChrono(), et traiterArriveeCoureur().

```
00015                               : QObject(parent)
00016 {
00017     BDD = BaseDeDonnees::getInstance();
00018     if(!BDD->estConnecte())
00019         BDD->connecter("Chrono-Cross");
00020     qDebug() << Q_FUNC_INFO << "Etat connexion BDD : " << BDD->estConnecte();
00021
00022     monChrono = new Chrono(this);
00023
00024     connect(monChrono, SIGNAL(chronoCreer()), this, SLOT(
      aChronoCree()));
00025     connect(monChrono, SIGNAL(chronoSynchroniser()), this, SLOT(
      aChronoSynchronise()));
00026     connect(monChrono, SIGNAL(classementCree()), this, SLOT(
      aClassementCree()));
```

```
00027     connect(monChrono, SIGNAL(chronoLance()), this, SLOT(aCommencee()));
00028     connect(monChrono, SIGNAL(courseArretee()), this, SLOT(
      aClassementArrete()));
00029     connect(monChrono, SIGNAL(chronoArrete()), this, SLOT(estFinie()));
00030     connect(monChrono, SIGNAL(nouvelleArrivee(QString)), this, SLOT(
      traiterArriveeCoureur(QString)));
00031     connect(monChrono, SIGNAL(chronoRecommence()), this, SLOT(
      recommencerChrono()));
00032     connect(this, SIGNAL(arriveeAjouteeBDD(QString, QString)), this, SLOT(
      getInformationCoureur(QString, QString)));
00033 }
```

8.4.2.2 ~Course()

`Course::~Course ( )`

Destructeur de la fenêtre principale.

Références BaseDeDonnees : :detruireInstance().

```
00040 {
00041     BaseDeDonnees::detruireInstance();
00042     qDebug() << Q_FUNC_INFO;
00043 }
```

8.4.3 Documentation des fonctions membres

8.4.3.1 aChronoCree

`void Course::aChronoCree ( )  [slot]`

SLOT aChronoCree() de la classe Course.

Emet le signal chronoCreer

Références chronoCreer().

Référencé par Course().

```
00467 {
00468     emit chronoCreer();
00469 }
```

8.4.3.2 aChronoSynchronise

`void Course::aChronoSynchronise ( )  [slot]`

SLOT aChronoSynchronise() de la classe Course.

Appelle la méthode creerClassement() de la classe Chrono

Références Chrono : :creerClassement(), et monChrono.

Référencé par Course().

```
00477 {
00478     if(monChrono->creerClassement())
00479         qDebug() << Q_FUNC_INFO << "NouveauClassement OK";
00480     else
00481         qDebug() << Q_FUNC_INFO << "ERREUR";
00482 }
```

8.4.3.3 aClassementArrete

```
void Course::aClassementArrete ( )  [slot]
```

SLOT aClassementArrete() de la classe Course.

Emet le signal classementArrete

Références classementArrete().

Référencé par Course().

```
00512 {
00513     qDebug() << Q_FUNC_INFO << "classement est arrété";
00514     emit classementArrete();
00515 }
```

8.4.3.4 aClassementCree

```
void Course::aClassementCree ( )  [slot]
```

SLOT aClassementCree de la classe Course.

Permet de signaler à la classe IHM que le chronomètre de la course est pret.

Références chronoCoursePret().

Référencé par Course().

```
00490 {
00491     qDebug() << Q_FUNC_INFO << "Chrono pret";
00492     emit chronoCoursePret();
00493 }
```

8.4.3.5 aCommencee

```
void Course::aCommencee ( )  [slot]
```

SLOT aCommencee() de la classe Course.

Emet le signal courseCommence.

Références courseCommence().

Référencé par Course().

```
00501 {
00502     qDebug() << Q_FUNC_INFO << "Course a commencée";
00503     emit courseCommence();
00504 }
```

8.4.3.6 ajouteArriveeBDD()

```
void Course::ajouteArriveeBDD (
            QString dossard,
            QString tempsArrivee )
```

Méthode ajouteArriveeBDD() de la classe Course.

Permet de récupérer l'es informations 'idInscrit d'un coureur d'après son numéro de dosasrd et l'ajoute à la table Arrivee

Paramètres

dossard	QString
tempsArrivee	QString

Références arriveeAjouteeBDD(), BDD, BaseDeDonnees : :executer(), formulerRequeteSelect(), et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :associerArriveeDossard().

```
00382 {
00383     QString idInscrit;
00384     QString condition = QString("NumeroDossard = %1").arg(dossard);
00385
00386     bool retourId = BDD->recuperer(this->formulerRequeteSelect("idInscrit"
      , "Inscrit", condition),idInscrit);
00387
00388     bool retour = BDD->executer(QString("INSERT INTO `Arrivee`(`idInscrit`, `Temps`) VALUES
       (%1,'%2')").arg(idInscrit).arg(tempsArrivee));
00389
00390     if(retourId)
00391     {
00392         if(retour)
00393         {
00394             qDebug() << Q_FUNC_INFO << "Ajout ok !";
00395             emit arriveeAjouteeBDD(dossard, tempsArrivee);
00396         }
00397     }
00398     else
00399         qDebug() << Q_FUNC_INFO << "Ajout echec !";
00400 }
```

8.4.3.7 arreterChrono()

```
void Course::arreterChrono ( )
```

Méthode arreterChronomètre() de la classe Course.

Appelle la méthode arreterChrono() de la classe Chrono

Références Chrono : :arreterChrono(), et monChrono.

Référencé par IHMChronoCross : :arreterChrono().

```
00127 {
00128     monChrono->arreterChrono();
00129 }
```

8.4.3.8 arreterClassement()

```
void Course::arreterClassement ( )
```

Méthode arreterClassement() de la classe Course.

Appelle la méthode arreterClassement () de la classe Chrono

Références Chrono : :arreterCourse(), et monChrono.

Référencé par IHMChronoCross : :arreterCourse().

```
00117 {
00118     monChrono->arreterCourse();
00119 }
```

8.4.3.9 arriveeAjouteeBDD

```
void Course::arriveeAjouteeBDD (
            QString dossard,
            QString tempsArrivee )  [signal]
```

Référencé par ajouteArriveeBDD(), et Course().

8.4.3.10 chronoCoursePret

```
void Course::chronoCoursePret ( )  [signal]
```

Référencé par aClassementCree().

8.4.3.11 chronoCreer

```
void Course::chronoCreer ( )  [signal]
```

Référencé par aChronoCree(), et Course().

8.4.3.12 chronometrer()

```
void Course::chronometrer ( )
```

Méthode chronometrer de la classe Course.

Méthode qui permet de lancer le chrono de la course

Références Chrono : :demarrer(), et monChrono.

Référencé par IHMChronoCross : :lancerCourse().

```
00102 {
00103     if(monChrono->demarrer())
00104     {
00105         qDebug() << Q_FUNC_INFO << "Valide";
00106     }
00107     else
00108         qDebug() << Q_FUNC_INFO << "Erreur lancement chrono";
00109 }
```

8.4.3.13 chronoRecommence

```
void Course::chronoRecommence ( )  [signal]
```

Référencé par Course(), et recommencerChrono().

8.4.3.14 classementArrete

```
void Course::classementArrete ( )  [signal]
```

Référencé par aClassementArrete().

8.4.3.15 convertirTemps()

```
QString Course::convertirTemps (
            QString tempsMS )
```

Méthode convertirTemps() de la classe Course.

Permet de convertir le temps recu sous le format HH :MM :SS.DDDDD en HH :MM :SS

Paramètres

tempsMS	

Renvoie

Retourne le temps converti

Référencé par traiterArriveeCoureur().

```
00197 {
00198     QString tempsS;
00199     QString tempsFinal;
00200
00201     bool tempsConverti = false;
00202     int i = 0;
00203
00204     while(!tempsConverti)
00205     {
00206         if(tempsMS[i] == '.')
00207         {
00208             tempsConverti = true;
00209         }
00210         else
00211         {
00212             tempsS[i] = tempsMS[i];
00213             i += 1;
00214         }
00215     }
00216
00217     if(tempsS.length() <2)
00218     {
00219         tempsFinal = "00:00:0" + tempsS;
00220         return tempsFinal;
00221     }
00222     else if (tempsS.length() <3)
00223     {
00224         tempsFinal = "00:00:" + tempsS;
00225         return tempsFinal;
00226     }
00227     else if (tempsS.length() <5)
00228     {
00229         tempsFinal = "00:0" + tempsS;
00230         return tempsFinal;
00231     }
00232     else if (tempsS.length() <6)
00233     {
00234         tempsFinal = "00:" + tempsS;
00235         return tempsFinal;
00236     }
00237     else if (tempsS.length() <8)
00238     {
00239         tempsFinal = "0" + tempsS;
00240         return tempsFinal;
00241     }
00242     else
00243         return tempsS;
00244 }
```

8.4.3.16 courseCommence

```
void Course::courseCommence ( )  [signal]
```

Référencé par aCommencee().

8.4.3.17 courseFinie

```
void Course::courseFinie ( )  [signal]
```

Référencé par estFinie().

8.4.3.18 creerChrono()

```
void Course::creerChrono ( )
```

Méthode creerChrono() de la classe Course.

Utilise la méthode creerChrono() de la classe Chrono

Références Chrono : :creer(), et monChrono.

Référencé par IHMChronoCross : :creerCourse().

```
00066 {
00067     monChrono->creer();
00068 }
```

8.4.3.19 estChronometragePret()

```
bool Course::estChronometragePret ( )
```

Méthode estChronometragePret() de la classe Course.

Méthode qui indique si le chrono est prêt à chronométrer cette course (utilise a métode estConnecte de la classe Chrono)

Références Chrono : :estConnecte(), et monChrono.

Référencé par IHMChronoCross : :initialiserCourse().

```
00091 {
00092     qDebug() << Q_FUNC_INFO << "Etat :\t" << monChrono->estConnecte();
00093     return monChrono->estConnecte();
00094 }
```

8.4.3.20 estFinie

```
void Course::estFinie ( )  [slot]
```

SLOT estFinie() de la classe Course.

Emet le signal courseFinie().

Références courseFinie().

Référencé par Course().

```
00523 {
00524     emit courseFinie();
00525 }
```

8.4.3.21 formulerRequeteSelect()

```
QString Course::formulerRequeteSelect (
            QString renseignements,
            QString sources,
            QString conditions )  [private]
```

Méthode formulerRequeteSelect de la classe Course.

Méthode qui renvoie la requête SQL formatée avec les arguments demandés.

Paramètres

renseignements	QString informations que l'on recherche
sources	QString où se trouve l'information recherchée
conditions	QString paramètres de recherche

Renvoie

La requête SQL

Référencé par ajouteArriveeBDD(), getDistance(), getHeure(), getInformationCoureur(), getListeCourses(), getListeManifestations(), getNbInscrit(), getNomCourse(), setIdCourse(), et verifierDossard().

```
00055 {
00056     QString requete = QString("SELECT %1 FROM %2 WHERE %3").arg(renseignements).arg(sources).arg(conditions
      );
00057     return requete;
00058 }
```

8.4.3.22 getDistance()

```
int Course::getDistance (
              QString course )
```

Méthode getDistance() de la classe Course.

Permet de retourner la distance d'une course d'arpsè son nom

Paramètres

course	

Renvoie

Retourne la distance d'une course

Références BDD, distance, formulerRequeteSelect(), et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :afficherInformationsCourse().

```
00277 {
00278     QString condition = QString("Nom = '%1'").arg(course);
00279
00280     bool retour = BDD->recuperer(this->formulerRequeteSelect("Distance", "
      Course", condition), distance);
00281     if(retour)
00282         return distance.toInt();
00283     else
00284         qDebug() << Q_FUNC_INFO << "Erreur";
00285         return 0;
00286 }
```

8.4.3.23 getHeure()

```
QString Course::getHeure (
              QString course )
```

Méthode getHeure() de la classe Course.

Permet de retourner l'heure de départ d'une course d'après son nom

Paramètres

course	

Renvoie

Retourne l'heure de départ d'une course

Références BDD, formulerRequeteSelect(), heureDepart, et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :afficherInformationsCourse().

```
00296 {
00297     QString condition = QString("Nom = '%1'").arg(course);
00298
00299     bool retour = BDD->recuperer(this->formulerRequeteSelect("HeureDepart"
      , "Course", condition), heureDepart);
00300
00301     if(retour)
00302         return heureDepart;
00303     else
00304         qDebug() << Q_FUNC_INFO << "Erreur";
00305         return "";
00306 }
```

8.4.3.24 getInformationCoureur

```
void Course::getInformationCoureur (
             QString dossard,
             QString tempsArrivee )  [slot]
```

SLOT getInformationCoureur(QString dossard, QString tempsArrivee) de la classe Course.

Récupere les informations d'un coureur d'après son numéro de dossard, puis émet un signal contenant les informations

Paramètres

dossard	
tempsArrivee	

Références BDD, formulerRequeteSelect(), INFORMATION_COUREUR_ARRIVEE_CLASSE, informationCoureurArrive, information↩ CoureurRecuperees(), et BaseDeDonnees : :recuperer().

Référencé par Course().

```
00556 {
00557     informationCoureurArrive << tempsArrivee << dossard;
00558     QString idCoureur;
00559     bool retour;
00560
00561     qDebug() << Q_FUNC_INFO << "informationCoureurArrive" <<
      informationCoureurArrive;
00562
00563     QString condition = QString("NumeroDossard = %1").arg(dossard);
00564     retour = BDD->recuperer(this->formulerRequeteSelect("idCoureur", "
      Inscrit", condition), idCoureur);
00565     if(!retour)
00566         return;
00567
00568     condition = QString("idCoureur = %1").arg(idCoureur);
00569     retour = BDD->recuperer(this->formulerRequeteSelect("
      Nom,Prenom,idClasse","Coureur",condition), informationCoureurArrive);
```

```
00570     if(!retour)
00571         return;
00572
00573     condition = QString("idClasse = %1").arg(informationCoureurArrive[
      INFORMATION_COUREUR_ARRIVEE_CLASSE]);
00574     retour = BDD->recuperer(this->formulerRequeteSelect("Nom", "Classe",
      condition), informationCoureurArrive[INFORMATION_COUREUR_ARRIVEE_CLASSE]);
00575     if(!retour)
00576         return;
00577
00578     qDebug() << Q_FUNC_INFO << "idCoureur" << idCoureur << "informationCoureurArrive" <<
      informationCoureurArrive;
00579
00580     emit informationCoureurRecuperees(informationCoureurArrive);
00581     informationCoureurArrive.clear();
00582 }
```

8.4.3.25 getListeCourses()

```
QVector< QString > Course::getListeCourses (
            QString manifestation )
```

Méthode getListeCourses() de la classe Course.

Permet de récuperer les listes de courses d'après le nom de la manifestation

Paramètres

manifestation	

Renvoie

Retourne un vector de QString contenant le nom des courses de la manifestation choisi

A faire verifier date

Références BDD, formulerRequeteSelect(), et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :listerCourses().

```
00436 {
00437     QString condition = QString("Nom = '%1'").arg(manifestation);
00438     QString idManifestation;
00439     QVector<QString> courses;
00440
00441     bool retour = BDD->recuperer(this->formulerRequeteSelect("
      idManifestation", "Manifestation", condition), idManifestation);
00442     qDebug() << Q_FUNC_INFO << "Etat requeteIdManifestation : " << retour << "\t info : " <<
      idManifestation;
00443
00444     condition = QString("`idManifestation` = %1 AND `Etat` = 'ACourir'").arg(idManifestation);
00445     retour = BDD->recuperer(this->formulerRequeteSelect("Nom", "Course",
      condition), courses);
00446     qDebug() << Q_FUNC_INFO << "Etat requeteNomCourses : " << retour << "\t info : " << courses;
00447
00451 /*
00452 QDate date = QDate::currentDate();
00453 qDebug() << "\r\n\r\n" << date << "\r\n\r\n" << "\r\n\r\n";
00454 */
00455
00456     return courses;
00457 }
```

8.4.3.26 getListeManifestations()

```
QStringList Course::getListeManifestations ( )
```

Méthode getListeManifestations() de la classe Course.

Permet de récupérer la liste des manifestations

Renvoie

Retourne un QStringList listeManifestations contenant toute les manifestations

Références BDD, formulerRequeteSelect(), et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :listerManifestations().

```
00409 {
00410     QStringList listeManifestations;
00411     QString i;
00412
00413     bool retour = BDD->recuperer("SELECT COUNT(*) FROM `Manifestation` WHERE 1", i);
00414
00415     int j = i.toInt();
00416
00417     qDebug() << Q_FUNC_INFO << retour << listeManifestations;
00418
00419     for(int i = 1; i <= j; i += 1)
00420     {
00421         QString condition = QString("idManifestation = %1").arg(i);
00422         retour = BDD->recuperer(this->formulerRequeteSelect("NOM", "
      Manifestation", condition), listeManifestations);
00423         qDebug() << Q_FUNC_INFO << retour << "i =" << i << "j =" << j << listeManifestations;
00424     }
00425     return  listeManifestations;
00426 }
```

8.4.3.27 getNbArrivee()

```
int Course::getNbArrivee ( )
```

Méthode getNbArrivee() de la classe Course.

Permet de récupérer le nombre de coureur de la table Arrivee

Renvoie

Le nombre d'arrivee

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :IHMChronoCross().

```
00315 {
00316     QString information;
00317     int nbArrivee;
00318     bool retour = BDD->recuperer("SELECT COUNT(*) FROM Arrivee WHERE 1;", information);
00319     if(retour)
00320     {
00321         nbArrivee = information.toInt();
00322         return nbArrivee;
00323     }
00324     else
00325         qDebug() << Q_FUNC_INFO << "Erreur";
00326         return 0;
00327 }
```

8.4.3.28 getNbInscrit()

```
int Course::getNbInscrit (
            QString course )
```

Méthode getNbInscrit de la classe Course.

Méthode qui renvoie le nombre d'inscrit d'une course d'après son ID depuis la base de donnée.

Paramètres

course	string nom d'une course

Renvoie

Le nombre d'inscrit dans une course

Références BDD, formulerRequeteSelect(), idCourse, et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :afficherInformationsCourse().

```
00254 {
00255     QString condition = QString("Nom = '%1'").arg(course);
00256     QString nbInscrit;
00257
00258     bool retour = BDD->recuperer(this->formulerRequeteSelect("idCourse", "
      Course", condition), idCourse);
00259
00260     if(retour)
00261     {
00262         retour = BDD->recuperer(QString("SELECT COUNT(*) FROM Inscrit WHERE idCourse = %1").arg
      (idCourse),nbInscrit);
00263     }
00264
00265     int nb = nbInscrit.toInt();
00266     return nb;
00267 }
```

8.4.3.29 getNomCourse()

```
QString Course::getNomCourse (
            QString dossard )
```

Méthode getNomCourse() de la classe Course.

Permet de retourner le nom d'une course d'après le numéro de dossard d'un inscrit

Paramètres

dossard	

Renvoie

Retourne le nom de la course ou rien selon l'état de la requête SQL

Références BDD, formulerRequeteSelect(), et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :associerArriveeDossard().

```
00337 {
00338     QString id;
00339     QString nomCourse;
00340     QString condition = QString("NumeroDossard = %1;").arg(dossard);
00341     bool retour = BDD->recuperer(this->formulerRequeteSelect("idCourse", "
      Inscrit", condition), id);
00342     if(retour)
00343     {
00344         condition = QString("idCourse = %1").arg(id);
00345         retour = BDD->recuperer(this->formulerRequeteSelect("Nom", "Course
      ", condition), nomCourse);
00346         if(retour)
```

```
00347             return nomCourse;
00348         else
00349         {
00350             qDebug() << Q_FUNC_INFO << "ERREUR";
00351             return nomCourse;
00352         }
00353     }
00354     else
00355     {
00356         qDebug() << Q_FUNC_INFO << "ERREUR";
00357         return nomCourse;
00358     }
00359 }
```

8.4.3.30 informationCoureurRecuperees

```
void Course::informationCoureurRecuperees (
             QStringList informationCoureur )  [signal]
```

Référencé par getInformationCoureur().

8.4.3.31 nouveauTempsArrivee

```
void Course::nouveauTempsArrivee (
             QString tempsArrivee )  [signal]
```

Référencé par traiterArriveeCoureur().

8.4.3.32 preparerChrono()

```
void Course::preparerChrono ( )
```

Méthode preparerChrono() de la classe Course.

Méthode qui appelle la méthode synchroniser de la classe Chrono

Références monChrono, et Chrono : :synchroniser().

Référencé par IHMChronoCross : :preparerCourse().

```
00076 {
00077     if(monChrono->synchroniser())
00078     {
00079         qDebug() << Q_FUNC_INFO << "SYNCHRO OK";
00080     }
00081     else
00082         qDebug() << Q_FUNC_INFO << "SYNCHRO ERREUR";
00083 }
```

8.4.3.33 recommencerChrono

```
void Course::recommencerChrono ( )  [slot]
```

SLOT recommencerChrono() de la classe Course.

Emet le signal chronoRecommence().

Références chronoRecommence().

Référencé par Course().

```
00533 {
00534     emit chronoRecommence();
00535 }
```

8.4.3.34 setEtat

```
bool Course::setEtat (
            QString etat )  [slot]
```

SLOT setEtat de la classe Course.

Change l'état d'une course dans la table Course

Paramètres

etat	

Renvoie

L'état de la requête

Références BDD, BaseDeDonnees : :executer(), et idCourse.

Référencé par IHMChronoCross : :arreterCourse(), IHMChronoCross : :initialiserCourse(), IHMChronoCross : :lancerCourse(), et IHMChronoCross : :terminerCourse().

```
00592 {
00593     bool retour = BDD->executer(QString("UPDATE `Course` SET `Etat`= '%1' WHERE `idCourse` = %2;
      ").arg(etat).arg(idCourse));
00594     return retour;
00595 }
```

8.4.3.35 setIdCourse()

```
void Course::setIdCourse (
            QString nomCourse )
```

Méthode setIdCourse() de la classe Course.

Permet d'initialiser l'id de la course

Paramètres

nomCourse	

Références BDD, formulerRequeteSelect(), idCourse, et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :afficherInformationsCourse().

```
00368 {
00369     QString condition = QString("Nom = '%1'").arg(nomCourse);
00370     bool retour = BDD->recuperer(this->formulerRequeteSelect("idCourse", "
      Course", condition), idCourse);
00371     qDebug() << Q_FUNC_INFO << retour;
00372 }
```

8.4.3.36 traiterArriveeCoureur

```
void Course::traiterArriveeCoureur (
            QString tempsArrivee )  [slot]
```

SLOT traiterArriveeCoureur(QString tempsArrivee) de la classe Course.

Paramètres

tempsArrivee	QString le temps d'arrivée

Références convertirTemps(), et nouveauTempsArrivee().

Référencé par Course().

```
00543 {
00544     QString temps = this->convertirTemps(tempsArrivee);
00545     emit nouveauTempsArrivee(temps);
00546 }
```

8.4.3.37 verifierDossard()

```
int Course::verifierDossard (
            QString dossard )
```

Méthode verifierDossard() de la classe Course.

Permet de verifier si un numéro de dossard est valide, si il est associer à la bonne course et si il n'a pas déjà été utilisé

Paramètres

dossard	QString numero de dossard

Renvoie

Renvoi le résultat de la verification sous la forme d'un booléen

Références BDD, formulerRequeteSelect(), idCourse, et BaseDeDonnees : :recuperer().

Référencé par IHMChronoCross : :associerArriveeDossard().

```
00139 {
00140     // 0 = Invalide , 1 = numéro valide mais mauvaise course , 2 = valide mais dossard déjà entré , 3 =
      valide
00141     int verification = 0;
00142
00143     QString idInscrit;
00144     QString coureurs;
00145     QString condition = QString("NumeroDossard = %1").arg(dossard);
00146
00147     bool retour = BDD->recuperer(this->formulerRequeteSelect("idInscrit",
      "Inscrit", condition),idInscrit);
00148
00149     if(retour)
00150     {
00151         // le numéro de dossard existe
00152         verification = 1;
00153         condition = QString("idCourse = %1 AND idInscrit = %2;").arg(idCourse).arg(idInscrit);
00154         retour = BDD->recuperer(this->formulerRequeteSelect("*", "Inscrit"
      , condition), coureurs);
00155         if(!coureurs.isEmpty())
00156         {
00157             // le numéro de dossard est enregistré pour la bonne course
00158             verification = 2;
00159             coureurs.clear();
00160             condition = QString("idInscrit = %1").arg(idInscrit);
00161             retour = BDD->recuperer(this->formulerRequeteSelect("*","
      Arrivee",condition), coureurs);
00162             qDebug() << Q_FUNC_INFO << "coureur : " << coureurs << coureurs.isEmpty();
00163             if(coureurs.isEmpty())
00164             {
00165                 // le coureur n'a pas déjà franchi la ligne d'arrivée
00166                 verification = 3;
00167                 qDebug() << Q_FUNC_INFO << "Dossard validé";
00168                 return  verification;
00169             }
00170             else
00171             {
00172                 qDebug() << Q_FUNC_INFO << "Dossard valide mais ce dossard a déjà franchi la ligne
       d'arrivée.";
00173                 return verification;
00174             }
00175         }
00176         else
00177         {
00178             qDebug() << Q_FUNC_INFO << "Dossard valide mais pas inscrit pour cette course.";
00179             return verification;
00180         }
00181     }
00182     else
00183     {
00184         qDebug() << Q_FUNC_INFO << "Numéro dossard invalide";
00185         return verification;
00186     }
00187 }
```

8.4.4 Documentation des données membres

8.4.4.1 BDD

```
BaseDeDonnees* Course::BDD  [private]
```

agrégation BaseDeDonnees

Référencé par ajouteArriveeBDD(), Course(), getDistance(), getHeure(), getInformationCoureur(), getListeCourses(), getListeManifestations(), getNbArrivee(), getNbInscrit(), getNomCourse(), setEtat(), setIdCourse(), et verifierDossard().

8.4.4.2 distance

```
QString Course::distance  [private]
```

Distance de la course.

Référencé par getDistance().

8.4.4.3 heureDepart

```
QString Course::heureDepart  [private]
```

L'heure de départ de la course.

Référencé par getHeure().

8.4.4.4 idCourse

```
QString Course::idCourse  [private]
```

Identifiant de la course.

Référencé par getNbInscrit(), setEtat(), setIdCourse(), et verifierDossard().

8.4.4.5 informationCoureurArrive

```
QStringList Course::informationCoureurArrive  [private]
```

Informations d'un coure.

Référencé par getInformationCoureur().

8.4.4.6 monChrono

```
Chrono* Course::monChrono  [private]
```

association Chrono

Référencé par aChronoSynchronise(), arreterChrono(), arreterClassement(), chronometrer(), Course(), creerChrono(), estChronometragePret(), et preparerChrono().

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

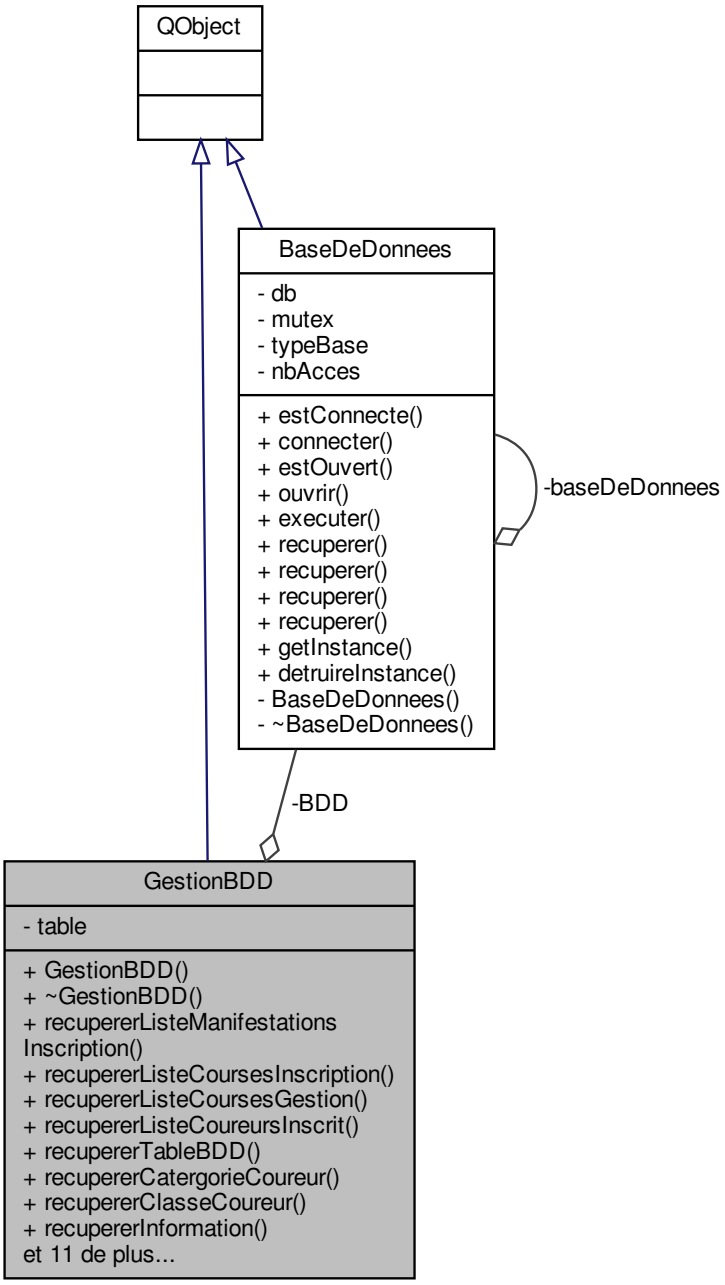
- course.h
- course.cpp

8.5 Référence de la classe GestionBDD

Déclaration de la classe GestionBDD.

```
#include <gestionbdd.h>
```

Graphe de collaboration de GestionBDD :



Signaux

- void nouvelInscrit (QStringList inscription)
- void nouveauCoureur (QStringList informationsCoureurClassement)
- void coureurSupprime ()
- void coureurModifie ()

Fonctions membres publiques

- GestionBDD (QObject *parent=nullptr)
  Constructeur GestionBDD() de la classe GestionBDD.
- ~GestionBDD ()
  Destructeur ~GestionBDD() de la classe GestionBDD()
- QVector< QString > recupererListeManifestationsInscription (QString INE)
- QVector< QString > recupererListeCoursesInscription (QString nom, QString sexe)
- QStringList recupererListeCoursesGestion (QString INE)
- QVector< QStringList > recupererListeCoureursInscrit (QString nomCourse)
- QVector< QStringList > recupererTableBDD (QString table)
- QString recupererCatergorieCoureur (QString idCategorie)
- QString recupererClasseCoureur (QString idClasse)
- QString recupererInformation (QString info, QString nomTable, QString nomEnregistrement)
- QVector< QString > recupererCategoriesCreation ()
- QVector< QString > recupererClassesCreation ()
- bool verifierCreation (QStringList enregistrement)
- bool verifierInformation (QString information, QString table)
- void ajouterNouveauCoureur (QStringList enregistrement)
- bool verifierModification (QStringList enregistrement)
- void modifierEnregistrement (QStringList enregistrement)
- bool verifierDossard (QString dossard)
- void ajouterNouvelInscrit (QStringList inscription)
- void supprimerCoureur (QString INE)
- void modifierCoureur (QStringList enregistrement)

Attributs privés

- BaseDeDonnees * BDD
- QVector< QStringList > table

8.5.1 Description détaillée

Déclaration de la classe GestionBDD.

Auteur

ANDREo Michaël

Version

1.0

8.5.2 Documentation des constructeurs et destructeur

8.5.2.1 GestionBDD()

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

Constructeur GestionBDD() de la classe GestionBDD.

Paramètres

parent	

Références BDD, BaseDeDonnees : :connecter(), BaseDeDonnees : :estConnecte(), et BaseDeDonnees : :getInstance().

```
00015                                     : QObject(parent)
00016 {
00017     BDD = BaseDeDonnees::getInstance();
00018     if(!BDD->estConnecte())
00019         BDD->connecter("Chrono-Cross");
00020
00021     qDebug() << Q_FUNC_INFO << "Etat connexion BDD : " << BDD->estConnecte();
00022 }
```

8.5.2.2 ~GestionBDD()

```
GestionBDD::~GestionBDD ( )
```

Destructeur ~GestionBDD() de la classe GestionBDD()

Références BaseDeDonnees : :detruireInstance().

```
00029 {
00030     BaseDeDonnees::detruireInstance();
00031     qDebug() << Q_FUNC_INFO;
00032 }
```

8.5.3 Documentation des fonctions membres

8.5.3.1 ajouterNouveauCoureur()

```
void GestionBDD::ajouterNouveauCoureur (
             QStringList enregistrement )
```

Références BDD, BaseDeDonnees : :executer(), nouveauCoureur(), et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :creerCoureur().

```
00219 {
00220     qDebug() << informationsCoureur;
00221     QStringList informationsCoureurClassement = informationsCoureur;
00222     // informationsCoureur [ 0 Categorie , 1 classe , 2 INE , 3 nom ,  4 prenom , 5 dateNaissance , 6 sexe
      ]
00223     qDebug() << Q_FUNC_INFO << QString("SELECT idCategorie FROM Categorie WHERE Nom = '%1' AND Sexe = '%2';
     ").arg(informationsCoureur[0]).arg(informationsCoureur[6]);
00224
00225     bool etat = BDD->recuperer(QString("SELECT idCategorie FROM Categorie WHERE Nom = '%1' AND
      Sexe = '%2';").arg(informationsCoureur[0]).arg(informationsCoureur[6]), informationsCoureur[0]);
00226
00227     if(etat)
00228     {
00229         qDebug() << Q_FUNC_INFO << QString("SELECT idCLasse FROM Classe WHERE Nom = '%1';").arg(
     informationsCoureur[1]);
00230
00231         etat = BDD->recuperer(QString("SELECT idCLasse FROM Classe WHERE Nom = '%1';").arg(
     informationsCoureur[1]),informationsCoureur[1]);
00232         if(etat)
00233         {
00234             qDebug() << Q_FUNC_INFO << QString("INSERT INTO `Coureur`(`idCoureur`, `idCategorie`,
      `idClasse`, `INE`, `Nom`, `Prenom`, `DateNaissance`, `Sexe`) VALUES (`idCoureur`,%0,%1,'%2','%3','%4','%5','%6')").
     arg(informationsCoureur[0]).arg(informationsCoureur[1]).arg(informationsCoureur[2]).arg(informationsCoureur[3]
     ).arg(informationsCoureur[4]).arg(informationsCoureur[5]).arg(informationsCoureur[6]);
00235
00236             etat = BDD->executer(QString("INSERT INTO `Coureur`(`idCoureur`, `idCategorie`,
      `idClasse`, `INE`, `Nom`, `Prenom`, `DateNaissance`, `Sexe`) VALUES (`idCoureur`,%0,%1,'%2','%3','%4','%5','%6')
     ").arg(informationsCoureur[0]).arg(informationsCoureur[1]).arg(informationsCoureur[2]).arg(
     informationsCoureur[3]).arg(informationsCoureur[4]).arg(informationsCoureur[5]).arg(informationsCoureur[6]));
00237             if(etat)
00238             {
00239                 qDebug() << Q_FUNC_INFO << "Enregistré(e) avec Succées !";
00240                 emit nouveauCoureur(informationsCoureurClassement);
00241             }
```

```
00242         }
00243         else
00244         {
00245             qDebug() << Q_FUNC_INFO << "ERREUR";
00246             informationsCoureurClassement.clear();
00247         }
00248     }
00249     else
00250     {
00251         qDebug() << Q_FUNC_INFO << "ERREUR";
00252         informationsCoureurClassement.clear();
00253     }
00254 }
```

8.5.3.2 ajouterNouvelInscrit()

```
void GestionBDD::ajouterNouvelInscrit (
            QStringList inscription )
```

Références BDD, BaseDeDonnees : :executer(), nouvelInscrit(), et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :ajouterNouvelleInscription().

```
00257 {
00258     // inscription [ idCoureur , idCourse , numeroDossard ]
00259     qDebug() << Q_FUNC_INFO << inscription;
00260     bool etat = BDD->executer(QString("INSERT INTO `Inscrit`(`idInscrit`, `idCoureur`,
       `idCourse`, `NumeroDossard`) VALUES (`idInscrit`,'%1','%2','%3')").arg(inscription[0]).arg(inscription[1]).arg(
      inscription[2]));
00261     qDebug() << Q_FUNC_INFO << "Ajout : " << etat;
00262     if(etat)
00263     {
00264         QString idCoureur = inscription[0];
00265         QString dossard = inscription[2];
00266         inscription.clear();
00267         etat = BDD->recuperer(QString("SELECT Nom, Prenom FROM Coureur WHERE idCoureur = '%1';"
      ).arg(idCoureur), inscription);
00268         inscription << dossard;
00269         qDebug() << Q_FUNC_INFO << inscription;
00270         emit nouvelInscrit(inscription);
00271     }
00272 }
```

8.5.3.3 coureurModifie

```
void GestionBDD::coureurModifie ( )  [signal]
```

Référencé par modifierCoureur().

8.5.3.4 coureurSupprime

```
void GestionBDD::coureurSupprime ( )  [signal]
```

Référencé par supprimerCoureur().

8.5.3.5 modifierCoureur()

```
void GestionBDD::modifierCoureur (
            QStringList enregistrement )
```

Références BDD, coureurModifie(), BaseDeDonnees : :executer(), et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :confirmerDialog().

```
00293 {
00294     // enregistrement [ 0 Categorie , 1 classe , 2 INE , 3 nom , 4 prenom , 5 dateNaissance , 6 sexe  , 7
       idCoureur]
00295
00296     qDebug() << Q_FUNC_INFO << enregistrement;
00297     bool etat = BDD->recuperer(QString("SELECT idCategorie FROM Categorie WHERE Nom = '%1';").
      arg(enregistrement[0]), enregistrement[0]);
00298     etat = BDD->recuperer(QString("SELECT idClasse FROM Classe WHERE Nom = '%1';").arg(
      enregistrement[1]), enregistrement[1]);
00299
00300     etat = BDD->executer(QString("UPDATE `Coureur` SET
       `idCategorie`='%1',`idClasse`='%2',`INE`='%3',`Nom`='%4',`Prenom`='%5',`DateNaissance`='%6',`Sexe`='%7' WHERE idCoureur = '%8';
      0]).arg(enregistrement[1]).arg(enregistrement[2]).arg(enregistrement[3]).arg(enregistrement[4]).arg(
      enregistrement[5]).arg(enregistrement[6]).arg(enregistrement[7]));
00301     qDebug() << Q_FUNC_INFO << "Modification : " << etat;
00302     if(etat)
00303         emit coureurModifie();
00304 }
```

8.5.3.6 modifierEnregistrement()

```
void GestionBDD::modifierEnregistrement (
            QStringList enregistrement )
```

8.5.3.7 nouveauCoureur

```
void GestionBDD::nouveauCoureur (
            QStringList informationsCoureurClassement )  [signal]
```

Référencé par ajouterNouveauCoureur().

8.5.3.8 nouvelInscrit

```
void GestionBDD::nouvelInscrit (
            QStringList inscription )  [signal]
```

Référencé par ajouterNouvelInscrit().

8.5.3.9 recupererCategoriesCreation()

```
QVector< QString > GestionBDD::recupererCategoriesCreation ( )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :passerModeNouveauCoureur().

```
00171 {
00172     QVector<QString> categories;
00173     bool etat = BDD->recuperer("SELECT DISTINCT Nom FROM Categorie WHERE 1;", categories);
00174     if(etat)
00175     {
00176         return categories;
00177     }
00178     else
00179     {
00180         qDebug() << Q_FUNC_INFO << "ERREUR";
00181         return categories;
00182     }
00183 }
```

8.5.3.10 recupererCatergorieCoureur()

```
QString GestionBDD::recupererCatergorieCoureur (
            QString idCategorie )
```

Références BDD, et BaseDeDonnees : :recuperer().

```
00133 {
00134     QString id;
00135     bool etat = BDD->recuperer(QString("SELECT Nom FROM Categorie WHERE idCategorie = %1;").arg
      (idCategorie), id);
00136     if(etat)
00137         return id;
00138     else
00139     {
00140         qDebug() << Q_FUNC_INFO << "ERREUR";
00141         return id;
00142     }
00143 }
```

8.5.3.11 recupererClasseCoureur()

```
QString GestionBDD::recupererClasseCoureur (
            QString idClasse )
```

Références BDD, et BaseDeDonnees : :recuperer().

```
00146 {
00147     QString id;
00148     bool etat = BDD->recuperer(QString("SELECT Nom FROM Classe WHERE idClasse = %1;").arg(
      idClasse), id);
00149     if(etat)
00150         return id;
00151     else
00152     {
00153         qDebug() << Q_FUNC_INFO << "ERREUR";
00154         return id;
00155     }
00156 }
```

8.5.3.12 recupererClassesCreation()

```
QVector< QString > GestionBDD::recupererClassesCreation ( )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :passerModeNouveauCoureur().

```
00186 {
00187     QVector<QString> classes;
00188     bool etat = BDD->recuperer("SELECT DISTINCT Nom FROM Classe;", classes);
00189     if(etat)
00190     {
00191         return classes;
00192     }
00193     else
00194     {
00195         qDebug() << Q_FUNC_INFO << "ERREUR";
00196         return classes;
00197     }
00198 }
```

8.5.3.13 recupererInformation()

```
QString GestionBDD::recupererInformation (
             QString info,
             QString nomTable,
             QString nomEnregistrement )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :afficherTable(), IHMGestionCross : :ajouterNouvelleInscription(), IHMGestionCross : :selectionnerCoureur(), IHMGestionCross : :selectionnerCourse(), et IHMGestionCross : :verifierNumeroDossardInscription().

```
00159 {
00160     bool etat = BDD->recuperer(QString("SELECT %1 FROM %2 WHERE %3;").arg(information).arg(
      nomTable).arg(nomEnregistrement), information);
00161     if(etat)
00162         return information;
00163     else
00164     {
00165         qDebug() << Q_FUNC_INFO << "ERREUR";
00166         return information;
00167     }
00168 }
```

8.5.3.14 recupererListeCoureursInscrit()

```
QVector< QStringList > GestionBDD::recupererListeCoureursInscrit (
             QString nomCourse )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :selectionnerCourse().

```
00043 {
00044     QString idCourse;
00045     bool etat = BDD->recuperer(QString("SELECT idCourse FROM Course WHERE Nom = '%1'").arg(
      nomCourse), idCourse);
00046
00047     QVector<QStringList> liste;
00048     etat = BDD->recuperer(QString("SELECT Nom, Prenom, NumeroDossard FROM `Inscrit` JOIN
       Coureur ON Inscrit.idCoureur=Coureur.idCoureur WHERE Inscrit.idCourse=%1;").arg(idCourse), liste);
00049     qDebug() << Q_FUNC_INFO << etat;
00050     return liste;
00051 }
```

8.5.3.15 recupererListeCoursesGestion()

```
QStringList GestionBDD::recupererListeCoursesGestion (
            QString INE )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :selectionnerCoureur().

```
00116 {
00117     QVector<QString> idCourses;
00118     QStringList nomCourses;
00119     bool etat = BDD->recuperer(QString("SELECT `Course`.idCourse FROM `Course` JOIN `Inscrit`
      ON `Course`.idCourse=`Inscrit`.idCourse JOIN `Coureur` ON `Inscrit`.idCoureur=`Coureur`.idCoureur WHERE
      INE='%1';").arg(INE), idCourses);
00120     int nbId = idCourses.size();
00121     qDebug() << Q_FUNC_INFO << etat << "nbID : " << nbId << "IDs : " << idCourses;
00122
00123     for(int i = 0; i < nbId; i += 1)
00124     {
00125         qDebug() << idCourses[i];
00126         etat = BDD->recuperer(QString("SELECT Nom FROM Course WHERE idCourse = '%1';").arg(
     idCourses[i]), nomCourses);
00127     }
00128     qDebug() << Q_FUNC_INFO << nomCourses;
00129     return nomCourses;
00130 }
```

8.5.3.16 recupererListeCoursesInscription()

```
QVector< QString > GestionBDD::recupererListeCoursesInscription (
            QString nom,
            QString sexe )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :listerCourses().

```
00103 {
00104     QVector<QString> listeCourses;
00105     bool etat = BDD->recuperer(QString("SELECT Course.Nom FROM `Course` JOIN Manifestation ON
      Manifestation.idManifestation=Course.idManifestation WHERE Manifestation.Nom = '%1' AND Course.Nom LIKE '%%2'
      ORDER BY HeureDepart ASC;").arg(nom).arg(sexe), listeCourses);
00106     if(etat)
00107         return listeCourses;
00108     else
00109     {
00110         qDebug() << Q_FUNC_INFO << "ERREUR";
00111         return listeCourses;
00112     }
00113 }
```

8.5.3.17 recupererListeManifestationsInscription()

```
QVector< QString > GestionBDD::recupererListeManifestationsInscription (
            QString INE )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :listerManifestations().

```
00054 {
00055     QVector<QString> manifestationInscrit;
00056     QVector<QString> manifestationNonInscrit;
00057     bool etat = BDD->recuperer(QString("SELECT Manifestation.Nom FROM `Manifestation` JOIN
      Course ON Manifestation.idManifestation=Course.idManifestation JOIN Inscrit ON Course.idCourse=Inscrit.idCourse
      JOIN Coureur ON Inscrit.idCoureur=Coureur.idCoureur WHERE INE = '%1';").arg(INE), manifestationInscrit);
00058
00059     int nbManifestations = manifestationInscrit.size();
00060     qDebug() << Q_FUNC_INFO << etat << manifestationInscrit;
00061
00062     if(etat)
00063     {
00064         QString nbManifestationsTotales;
00065         etat = BDD->recuperer("SELECT COUNT(*) FROM Manifestation;", nbManifestationsTotales);
00066         if(etat)
00067         {
00068             if(QString::number(nbManifestations) == nbManifestationsTotales)
00069             { // le coureur est inscrit à toutes les courses
00070                 manifestationNonInscrit << "Inscrit(e) à toute les courses disponibles";
00071                 return manifestationNonInscrit;
00072             }
00073             else if(nbManifestations == 0)
00074             { // le coureur n'est inscrit à aucune course
00075                 etat = BDD->recuperer("SELECT Nom FROM Manifestation;",manifestationNonInscrit)
      ;
00076                 qDebug()<< Q_FUNC_INFO << manifestationNonInscrit;
00077                 return manifestationNonInscrit;
00078             }
00079             else
00080             { // le coureur est inscrit à au moins une course mais pas toute
00081                 for(int i = 0; i < nbManifestations; i += 1)
00082                 {
00083                 etat = BDD->recuperer(QString("SELECT Nom FROM Manifestation WHERE Nom!='%1'
      ORDER BY Date ASC").arg(manifestationInscrit[i]), manifestationNonInscrit);
00084                 qDebug()<< Q_FUNC_INFO << manifestationNonInscrit;
00085                 }
00086                 return manifestationNonInscrit;
00087             }
00088         }
00089         else
00090         {
00091             qDebug() << Q_FUNC_INFO << "ERREUR";
00092             return manifestationNonInscrit;
00093         }
00094     }
00095     else
00096     {
00097         qDebug() << Q_FUNC_INFO << "ERREUR";
00098         return manifestationNonInscrit;
00099     }
00100 }
```

8.5.3.18 recupererTableBDD()

```
QVector< QStringList > GestionBDD::recupererTableBDD (
            QString table )
```

Références BDD, BaseDeDonnees : :recuperer(), et table.

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :mettreAJourTableCoureur().

```
00035 {
00036     table.clear();
00037     bool etat = BDD->recuperer(QString("SELECT * FROM %1 WHERE 1 ORDER BY INE ASC;").arg(
      nomTable), table);
00038     qDebug() << Q_FUNC_INFO << etat;
00039     return table;
00040 }
```

8.5.3.19 supprimerCoureur()

```
void GestionBDD::supprimerCoureur (
            QString INE )
```

Références BDD, coureurSupprime(), BaseDeDonnees : :executer(), et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :confirmerDialog().

```
00275 {
00276     qDebug() << Q_FUNC_INFO << INE;
00277     QStringList enregistrement;
00278     QString idCoureur;
00279     bool etat = BDD->recuperer(QString("SELECT idCoureur FROM Coureur WHERE INE = '%1';").arg(
      INE), idCoureur);
00280     qDebug() << Q_FUNC_INFO << idCoureur;
00281     etat = BDD->executer(QString("DELETE FROM `Coureur` WHERE `Coureur`.`idCoureur` = '%1';").
      arg(idCoureur));
00282     qDebug() << Q_FUNC_INFO << "Suppression : " << etat;
00283     if(etat)
00284         emit coureurSupprime();
00285
00286     etat = BDD->recuperer(QString("SELECT * FROM `Coureur` WHERE idCoureur = '%1';").arg(
      idCoureur), enregistrement);
00287     qDebug() << Q_FUNC_INFO << enregistrement;
00288     if(etat)
00289         qDebug() << Q_FUNC_INFO << "ERREUR";
00290 }
```

8.5.3.20 verifierCreation()

```
bool GestionBDD::verifierCreation (
            QStringList enregistrement )
```

8.5.3.21 verifierDossard()

```
bool GestionBDD::verifierDossard (
            QString dossard )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :verifierNumeroDossardInscription().

```
00208 {
00209     QStringList coureurs;
00210     bool etat = BDD->recuperer(QString("SELECT * FROM Inscrit WHERE NumeroDossard = %1;").arg(
      dossard), coureurs);
00211     qDebug() << Q_FUNC_INFO << dossard[0];
00212     if(etat)
00213         return false;
00214     else
00215         return true;
00216 }
```

8.5.3.22 verifierInformation()

```
bool GestionBDD::verifierInformation (
            QString information,
            QString table )
```

Références BDD, et BaseDeDonnees : :recuperer().

Référencé par IHMGestionCross : :verifierInformationsCreerCoureur(), et IHMGestionCross : :verifierInformationsModifierCoureur().

```
00201 { //information [ 0 info ; 1 table ]
00202     QStringList info;
00203     bool etat = BDD->recuperer(QString("SELECT * FROM %1 WHERE %2;").arg(
      table).arg(information), info);
00204     return etat;
00205 }
```

8.5.3.23 verifierModification()

```
bool GestionBDD::verifierModification (
            QStringList enregistrement )
```

8.5.4 Documentation des données membres

8.5.4.1 BDD

```
BaseDeDonnees* GestionBDD::BDD  [private]
```

Référencé par ajouterNouveauCoureur(), ajouterNouvelInscrit(), GestionBDD(), modifierCoureur(), recupererCategoriesCreation(), recupererCatergorieCoureur(), recupererClasseCoureur(), recupererClassesCreation(), recupererInformation(), recupererListe↩ CoureursInscrit(), recupererListeCoursesGestion(), recupererListeCoursesInscription(), recupererListeManifestationsInscription(), recupererTableBDD(), supprimerCoureur(), verifierDossard(), et verifierInformation().

8.5.4.2 table

```
QVector<QStringList> GestionBDD::table  [private]
```

Référencé par recupererTableBDD().

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

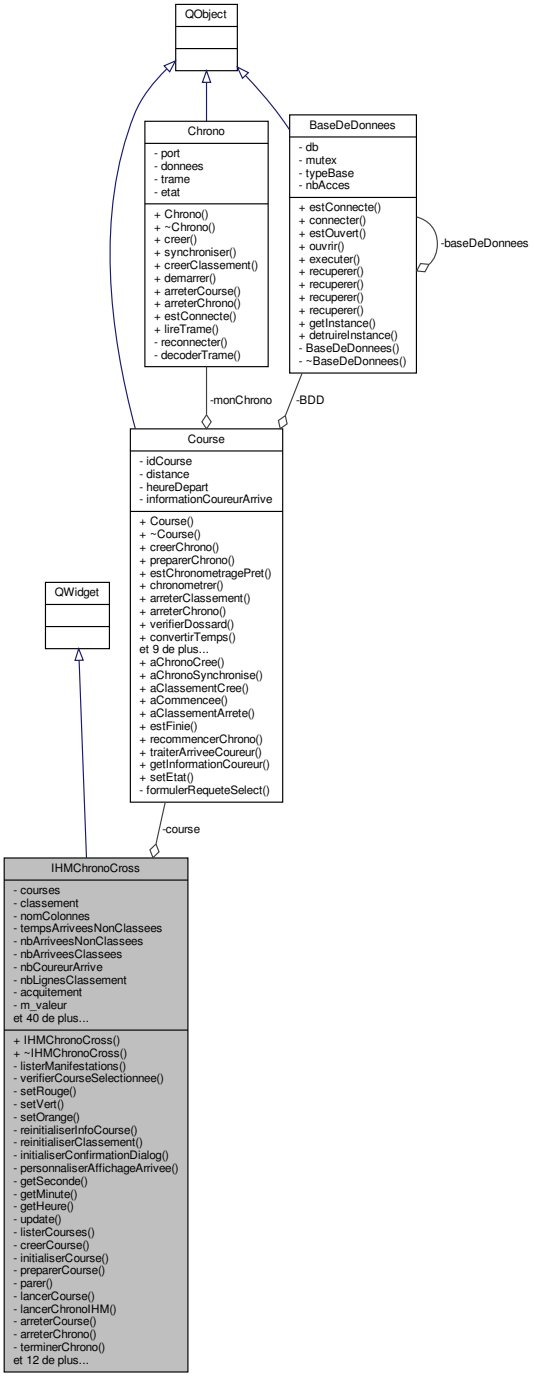
- gestionbdd.h
- gestionbdd.cpp

8.6 Référence de la classe IHMChronoCross

La fenêtre principale de l'application Chrono-Cross.

`#include <ihmchronocross.h>`

Graphe de collaboration de IHMChronoCross :



Fonctions membres publiques

— IHMChronoCross (QWidget *parent=nullptr)
Constructeur de la fenêtre principale.
— ~IHMChronoCross ()
Destructeur de la fenêtre principale.

Connecteurs privés

- void listerCourses (QString manifestation)
  Méthode listerCourses() de la Classe IHMChronoCross.
- void creerCourse (QString nomCourse)
  Méthode creerCourse() de la classe IHMChronoCross.
- void initialiserCourse ()
  SLOT initialiserCourse() de la classe IHMChrono-Cross.
- void preparerCourse ()
  Slots preparerCourse() de la classe IHMChronoCross.
- void parer ()
  Private SLOT parer() de la classe IHMChronoCross.
- void lancerCourse ()
  SLOT lancerCourse() de la classe IHMChronoCross.
- void lancerChronoIHM ()
  SLOTS lancerChronoIHM() de la classe IHMChronoCross.
- void arreterCourse ()
  SLOT arreterCourse() de la Classe IHMChronoCross.
- void arreterChrono ()
  SLOT arreterChrono() de la classe IHMChronoCross.
- void terminerChrono ()
  SLOT terminerChrono() de la classe IHMChronoCross.
- void terminerCourse ()
  Private SLOT terminerCourse() de la classe IHMChronoCross.
- void commencerNouvelleCourse ()
  SLOT commencerNouvelleCourse() de la classe IHMChronoCross.
- void ajouterArriveeCoureur (QString tempsArrivee)
  SLOT ajouterArriveeCoureur(QString tempsArrivee) de la classe IHMChronoCross.
- void associerArriveeDossard ()
  SLOT associerArriveeDossard() de la classe IHMChronoCross.
- void classerArrivee (QStringList informationCoureur)
  SLOT classerArrivee() de la classe IHMChronoCross.
- void quitter ()
  SLOT quitter() de la classe IHMChronoCross.
- void tic ()
  Méthode tic() de la classe IHMChronoCross.
- void afficherInformationsCourse (QString nomCourse)
  SLOT afficherInformationsCourse() de la classe IHMChronoCross.
- void mettreAJourNbArriveesNonClassees (int nbArrivee)
  SOT mettreAJourNbArriveesNonClassees() de la classe IHMChronoCross.
- void mettreAJourNbArriveesClassees (int nbArrivee)
  SLOT mettreAJourNbArriveesClassees() de la classe IHMChronoCross.
- void quitterDialog ()
  SLOT quitterDialog() de la classe IHMChronoCross.
- void supprimerPremierTemps ()
  SLOT supprimerPremierTemps() de la classe IHMChronoCross.

Fonctions membres privées

- void listerManifestations ()
  Méthode listerManifestations() de la Classe IHMChronoCross.
- bool verifierCourseSelectionnee (QString courseSelectionnee)
  Méthode verifierCourseSelectionnee() de la classe IHMChronoCross.
- void setRouge (QLabel ∗label)
  Méthode setRouge() de la classe IHMChronoCross.
- void setVert (QLabel ∗label)
  Méthode setVert() de la classe IHMChronoCross.
- void setOrange (QLabel ∗label)
  Méthode setOrange() de la classe IHMChronoCross.
- void reinitialiserInfoCourse ()
  Méthode reinitialiserInfoCourse() de la classe IHMChronoCross.
- void reinitialiserClassement ()
  Méthode reinitialiserClassement() de la classe IHMChronoCross.
- void initialiserConfirmationDialog ()
  Méthode initialiserConfirmationDialog() de la classe IHMChronoCross.
- void personnaliserAffichageArrivee (int nbLignesClassement)
  Méthode personnaliserAffichageArrivee() de la classe IHMChronoCross.
- long getSeconde ()
  Méthode getSeconde() de la classe IHMChronoCross.
- long getMinute ()

Méthode getMinute() de la classe IHMChronoCross.
— long getHeure ()
Méthode getHeure() de la classe IHMChronoCross.
— void update ()
Méthode update() de la classe IHMChronoCross.

Attributs privés

— Course ∗ course
agrégation d'une course
— QVector< QStringList > courses
liste des courses pour une manifestation
— QStringList classement
Liste du classement pour une course.
— QStringList nomColonnes
Liste de nom des colonnes du classement "Temps, dossard, nom...".
— QStringList tempsArriveesNonClassees
Liste des arrivées non classés.
— int nbArriveesNonClassees
— int nbArriveesClassees
— int nbCoureurArrive
Nombre de coureurs arrivés.
— int nbLignesClassement
nombre de lignes du classement
— bool acquitement
— long m_valeur
Valeur du chrono.
— QComboBox ∗ cbListeManifestations
Combobox contenant les différentes manifesations de la base de donnée.
— QComboBox ∗ cbListeCourses
Combobox contenant les différentes course de la base de donnée.
— QLabel ∗ labelManifestations
Label contenant le QString "Manifestations : " devant cbListeManifestations.
— QLabel ∗ labelListeCourses
Label contenant le QString "Courses : " devant cbListeCourses.
— QLabel ∗ labelNumeroDossard
Label contenant le QString "N°dossard : ".
— QLabel ∗ labelZoneCourse
Label contenant le texte : Course symbolisant la zone course.
— QLabel ∗ labelZoneClassement
Label contenant le texte : Classement symbolisant la zone classement.
— QLabel ∗ labelZoneChrono
Label contenant le texte "Chrono : " symbolisant la zone chrono.
— QLabel ∗ logoChronoCross
Label contenant l'image du logo Chrono-Cross.
— QLabel ∗ labelEtatChrono
Label contenant le texte "Etat : " pour la led chrono.
— QLabel ∗ labelLedChrono
Label contenant la LED (état vert, orange ou rouge) qui représente l'état de la connexion avec le TAG HEUER.
— QLabel ∗ labelLedCourse
Label contenant la LED (état vert, orange ou rouge) qui représente l'état du chrono, si il est prêt à faire une course.
— QLabel ∗ labelEtatCourse
Label contenant le texte "Etat : " pour la led course.
— QLabel ∗ labelZoneArrivees
Label contenant le QString "Arrivées : " symbolisant la zone des arrivées non classées.
— QLabel ∗ labelMessageDossard
Label du message d'érreur pour les numéros de dossards.
— QLabel ∗ labelMessageSupprimer
label contenant le message d'information pour supprimer le premier temps non classée
— QLabel ∗ labelNbInscrit
Label contentant le texte "Nombre d'inscrit :".
— QLabel ∗ labelNbArriveesNonClassees
Label contentant le texte "Nombre d'arrivées non classées :".
— QLabel ∗ labelNbArriveesClassees
Label contentant le texte "Nombre d'arrivées classées :".
— QLCDNumber ∗ QLCDNbArriveesNonClassees
QLCD qui affiche le nombre d'arrivées non classées.
— QLCDNumber ∗ QLCDNbArriveesClassees
QLCD qui affiche le nombre d'arrivées classées.
— QLabel ∗ labelNomCourse
— QLabel ∗ labelDistanceCourse

— QLabel * labelHeureCourse
— QLCDNumber * QLCDChrono
QLCD contenant le chrono.
— QLineEdit * lineEditNumeroDossard
LineEdit qui permet de rentrer un numéro de dossard.
— QPushButton * bSynchroniser
Bouton Synchroniser.
— QPushButton * bLancer
Bouton Lancer le chronomètre.
— QPushButton * bArreter
Bouton Arreter.
— QPushButton * bTerminer
Bouton Terminer.
— QPushButton * bAssocier
Bouton Associer.
— QTableView * vueTableauClassement
Vue en liste classement.
— QStandardItemModel * modeleClassement
Model du classement.
— QListView * vueListeTempsArriveesNonClassees
Vue en tableau des temps non classés.
— QStringListModel * modeleArriveesNonClassees
Model des arrivées non classés.
— QTimer * m_timer
Chrono de l'ihm.
— QDialog * confirmationDialog
— QLabel * labelConfirmationDialog
— QPushButton * bConfirmationDialog
— QPushButton * bAnnulerDialog

8.6.1 Description détaillée

La fenêtre principale de l'application Chrono-Cross.

Auteur

ANDREO Michaël

Version

1.1

8.6.2 Documentation des constructeurs et destructeur

8.6.2.1 IHMChronoCross()

```
IHMChronoCross::IHMChronoCross (
            QWidget * parent = nullptr )
```

Constructeur de la fenêtre principale.

Paramètres

parent	QObject Adresse de l'objet Qt parent (0 = pas de parent car c'est la fenêtre principale)

Références afficherInformationsCourse(), ajouterArriveeCoureur(), arreterChrono(), arreterCourse(), associerArriveeDossard(), bArreter, bAssocier, bLancer, bSynchroniser, bTerminer, cbListeCourses, cbListeManifestations, classement, classerArrivee(),

commencerNouvelleCourse(), course, creerCourse(), Course : :getNbArrivee(), IMAGECHRONOCROSS, initialiserCourse(), labelDistanceCourse, labelEtatChrono, labelEtatCourse, labelHeureCourse, labelLedChrono, labelLedCourse, labelListeCourses, labelManifestations, labelMessageDossard, labelMessageSupprimer, labelNbArriveesClassees, labelNbArriveesNonClassees, labelNbInscrit, labelNomCourse, labelNumeroDossard, labelZoneArrivees, labelZoneChrono, labelZoneClassement, labelZoneCourse, lancerChronoIHM(), lancerCourse(), lineEditNumeroDossard, listerCourses(), listerManifestations(), logoChronoCross, m_timer, m_valeur, modeleArriveesNonClassees, modeleClassement, nbArriveesClassees, nbArriveesNonClassees, nbCoureurArrive, nbLignesClassement, nomColonnes, parer(), preparerCourse(), QLCDChrono, QLCDNbArriveesClassees, QLCDNbArriveesNonClassees, quitter(), setRouge(), TAILLETEXTEBUTON, TAILLETEXTEINFO, TAILLETEXTELABEL, TAILLETEXTELISTE, TAILLETEXTESUPPRIMER, terminerChrono(), terminerCourse(), tic(), vueListeTempsArriveesNonClassees, et vueTableauClassement.

```
00017                                           : QWidget(parent)
00018 {
00019
00020     course = new Course(this);
00021
00022     nbCoureurArrive = course->getNbArrivee();
00023     nbArriveesNonClassees = 0;
00024     nbArriveesClassees =0;
00025
00026     m_valeur = long(1.33);
00027     m_timer = new QTimer(this);
00028
00029     // défini la taille du text
00030     QFont texteLabel;
00031     texteLabel.setPointSize(TAILLETEXTELABEL);
00032
00033     QFont texteListe;
00034     texteListe.setPointSize(TAILLETEXTELISTE);
00035
00036     QFont texteInformation;
00037     texteInformation.setPointSize(TAILLETEXTEINFO);
00038
00039     QFont texteMessageSupprimer;
00040     texteMessageSupprimer.setPointSize(TAILLETEXTESUPPRIMER);
00041
00042     // les widgets
00043     cbListeManifestations = new QComboBox(this);
00044     cbListeManifestations->setFixedSize(this->width()*0.66, this->height()*0.08);
00045     cbListeManifestations->addItem(("< Séléctionner une Manifestation >"));
00046     cbListeManifestations->setFont(texteListe);
00047
00048     cbListeCourses = new QComboBox(this);
00049     cbListeCourses->setFixedSize(this->width()*0.66, this->height()*0.08);
00050     cbListeCourses->addItem(("< Séléctionner une Course >"));
00051     cbListeCourses->setFont(texteListe);
00052
00053     labelManifestations = new QLabel(tr("Manifestations : "), this);
00054     labelManifestations->setFont(texteLabel);
00055     labelListeCourses = new QLabel(tr("Courses : "), this);
00056     labelListeCourses->setFont(texteLabel);
00057     labelZoneCourse = new QLabel(tr("Course "), this);
00058     labelZoneCourse->setFont(texteLabel);
00059     labelNumeroDossard = new QLabel(tr("N° dossard : "), this);
00060     labelNumeroDossard->setFont(texteLabel);
00061     labelZoneClassement = new QLabel(tr("Classement : "), this);
00062     labelZoneClassement->setFont(texteLabel);
00063     labelZoneArrivees = new QLabel(tr("Arrivées : "), this);
00064     labelZoneArrivees->setFont(texteLabel);
00065     labelZoneChrono = new QLabel(tr("Chrono "), this);
00066     labelZoneChrono->setFont(texteLabel);
00067     labelMessageDossard = new QLabel(tr(""));
00068     labelMessageDossard->setFont(texteLabel);
00069     labelNbInscrit = new QLabel(tr("Inscrits : "), this);
00070     labelNbInscrit->setFont(texteInformation);
00071     labelNbArriveesNonClassees = new QLabel(tr("Non classées: "), this);
00072     labelNbArriveesNonClassees->setFont(texteInformation);
00073     labelNbArriveesClassees = new QLabel(tr("Classées: "), this);
00074     labelNbArriveesClassees->setFont(texteInformation);
00075     labelNomCourse = new QLabel(tr("Nom : "), this);
00076     labelNomCourse->setFont(texteInformation);
00077     labelDistanceCourse = new QLabel(tr("Distance : "), this);
00078     labelDistanceCourse->setFont(texteInformation);
00079     labelHeureCourse = new QLabel(tr("Heure : "), this);
00080     labelHeureCourse->setFont(texteInformation);
00081
00082     labelEtatChrono = new QLabel(tr("Etat : "),this);
00083     labelEtatChrono->setFont(texteInformation);
00084     labelLedChrono = new QLabel(this);
00085     setRouge(labelLedChrono);
00086
00087     labelEtatCourse = new QLabel(tr("Etat : "),this);
00088     labelEtatCourse->setFont(texteInformation);
00089     labelLedCourse = new QLabel(this);
00090     setRouge(labelLedCourse);
00091
```

```
00092     labelMessageSupprimer = new QLabel(tr(""), this);
00093     labelMessageSupprimer->setFont(texteMessageSupprimer);
00094
00095     QLCDNbArriveesNonClassees = new QLCDNumber(this);
00096     QLCDNbArriveesNonClassees->setDigitCount(2);
00097     QLCDNbArriveesNonClassees->display("--");
00098     QLCDNbArriveesClassees = new QLCDNumber(this);
00099     QLCDNbArriveesClassees->setDigitCount(2);
00100     QLCDNbArriveesClassees->display("--");
00101     QLCDChrono = new QLCDNumber(this);
00102     QLCDChrono->setDigitCount(8);
00103     QLCDChrono->display("--:--:--");
00104     QLCDChrono->setFixedSize(this->width()*1.5, this->height()*0.5);
00105
00106     logoChronoCross  = new QLabel(this);
00107     QPixmap pixmap_img(IMAGECHRONOCROSS);
00108     logoChronoCross->setPixmap(pixmap_img);
00109
00110     // défini la taille du text dans les QPushButtons
00111     QFont texteBouton;
00112     texteBouton.setPointSize(TAILLETEXTEBUTON);
00113
00114     bSynchroniser = new QPushButton(QString::fromUtf8("Synchroniser"), this);
00115     bSynchroniser->setDefault(false);
00116     bSynchroniser->setEnabled(false);
00117     bSynchroniser->setFont(texteBouton);
00118
00119     bLancer = new  QPushButton(QString::fromUtf8("Lancer"), this);
00120     bLancer->setDefault(false);
00121     bLancer->setEnabled(false);
00122     bLancer->setFont(texteBouton);
00123
00124     bArreter = new QPushButton(QString::fromUtf8("Arrêter"), this);
00125     bArreter->setDefault(false);
00126     bArreter->setEnabled(false);
00127     bArreter->setFont(texteBouton);
00128
00129     bTerminer = new QPushButton(QString::fromUtf8("Terminer"), this);
00130     bTerminer->setDefault(false);
00131     bTerminer->setEnabled(false);
00132     bTerminer->setFont(texteBouton);
00133
00134     bAssocier = new QPushButton(QString::fromUtf8("Associer"), this);
00135     bAssocier->setEnabled(false);
00136     bAssocier->setFont(texteBouton);
00137
00138     lineEditNumeroDossard = new QLineEdit(this);
00139     lineEditNumeroDossard->setFont(texteLabel);
00140     lineEditNumeroDossard->setEnabled(false);
00141
00142     vueTableauClassement = new QTableView(this);
00143     modeleClassement = new QStandardItemModel(0, 5);
00144     nomColonnes << "Temps" << "Numéro de Dossard" << "Nom" << "Prenom" << "Classe";
00145     modeleClassement->setHorizontalHeaderLabels(nomColonnes);
00146     vueTableauClassement->setModel(modeleClassement);
00147     vueTableauClassement->setEditTriggers(QAbstractItemView::NoEditTriggers);
00148
00149     // Redimensionner automatiquement la colonne pour occuper l'espace disponible
00150     vueTableauClassement->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch
    );
00151     nbLignesClassement = modeleClassement->rowCount();
00152
00153     vueListeTempsArriveesNonClassees = new QListView(this);
00154     modeleArriveesNonClassees = new QStringListModel(
    classement);
00155     vueListeTempsArriveesNonClassees->setModel(
    modeleArriveesNonClassees);
00156     vueListeTempsArriveesNonClassees->setEditTriggers(
    QAbstractItemView::NoEditTriggers);
00157
00158     QAction *actionQuitter = new QAction("&Quitter", this);
00159     actionQuitter->setShortcut(QKeySequence(QKeySequence::Quit)); //Ctrl+Q
00160     addAction(actionQuitter);
00161
00162     QAction *actionEntrerPAD = new QAction("&Entrer1", this);
00163     actionEntrerPAD->setShortcut(QKeySequence(Qt::Key_Enter));
00164     addAction(actionEntrerPAD);
00165
00166     QAction *actionEntrerRETURN = new QAction("&Entrer2", this);
00167     actionEntrerRETURN->setShortcut(QKeySequence(Qt::Key_Return));
00168     addAction(actionEntrerRETURN);
00169
00170     // le positionnement
00171     QHBoxLayout *listesLayout = new QHBoxLayout;
00172     listesLayout->addWidget(labelManifestations);
00173     listesLayout->addWidget(cbListeManifestations);
00174     listesLayout->addStretch();
00175     listesLayout->addWidget(labelListeCourses);
00176     listesLayout->addWidget(cbListeCourses);
00177     listesLayout->addStretch();
00178     listesLayout->addWidget(logoChronoCross);
```

```
00179     listesLayout->addStretch();
00180     listesLayout->setContentsMargins(0, 0, 0, 20); // G H D B
00181
00182     QVBoxLayout *classementLayout = new QVBoxLayout;
00183     classementLayout->addWidget(labelZoneClassement);
00184     classementLayout->addWidget(vueTableauClassement);
00185
00186     QHBoxLayout *etatLedCourseLayout = new QHBoxLayout;
00187     etatLedCourseLayout->addWidget(labelEtatCourse);
00188     etatLedCourseLayout->addWidget(labelLedCourse);
00189
00190     QHBoxLayout *infoCourseLayout = new QHBoxLayout;
00191     infoCourseLayout->addWidget(labelZoneCourse);
00192     infoCourseLayout->addLayout(etatLedCourseLayout);
00193     infoCourseLayout->addStretch();
00194     infoCourseLayout->addWidget(labelNomCourse);
00195     infoCourseLayout->addStretch();
00196     infoCourseLayout->addWidget(labelDistanceCourse);
00197     infoCourseLayout->addStretch();
00198     infoCourseLayout->addWidget(labelHeureCourse);
00199     infoCourseLayout->addStretch();
00200     infoCourseLayout->addWidget(labelNbInscrit);
00201     infoCourseLayout->addStretch();
00202
00203
00204     QHBoxLayout *boutonCourseLayout = new QHBoxLayout;
00205     boutonCourseLayout->addStretch();
00206     boutonCourseLayout->addWidget(bSynchroniser);
00207     boutonCourseLayout->addStretch();
00208     boutonCourseLayout->addWidget(bTerminer);
00209     boutonCourseLayout->addStretch();
00210     boutonCourseLayout->setContentsMargins(0, 5, 0, 0); // G H D B
00211
00212     QVBoxLayout *courseLayout = new QVBoxLayout;
00213     courseLayout->addStretch();
00214     courseLayout->addLayout(infoCourseLayout);
00215     courseLayout->addLayout(boutonCourseLayout);
00216
00217     QHBoxLayout *etatLedChronoLayout = new QHBoxLayout;
00218     etatLedChronoLayout->addWidget(labelEtatChrono);
00219     etatLedChronoLayout->addWidget(labelLedChrono);
00220
00221     QHBoxLayout *QLCDChronoLayout = new QHBoxLayout;
00222     QLCDChronoLayout->addWidget(QLCDChrono);
00223     QLCDChronoLayout->setContentsMargins(0, 0, 0, 0); // G H D B
00224
00225     QHBoxLayout *infoBoutonChronoLayout = new QHBoxLayout;
00226     infoBoutonChronoLayout->addWidget(labelZoneChrono);
00227     infoBoutonChronoLayout->addLayout(etatLedChronoLayout);
00228     infoBoutonChronoLayout->addWidget(bLancer);
00229     infoBoutonChronoLayout->addStretch();
00230     infoBoutonChronoLayout->addWidget(bArreter);
00231     infoBoutonChronoLayout->addStretch();
00232     infoBoutonChronoLayout->setContentsMargins(0, 0, 0, 20); // G H D B
00233
00234     QHBoxLayout *dossardLayout = new QHBoxLayout;
00235     dossardLayout->addWidget(labelNumeroDossard);
00236     dossardLayout->addWidget(lineEditNumeroDossard);
00237     dossardLayout->addWidget(bAssocier);
00238     dossardLayout->setContentsMargins(0, 10, 0, 0); // G H D B
00239     dossardLayout->addStretch();
00240
00241     QHBoxLayout *infoArriveesLayout = new QHBoxLayout;
00242     infoArriveesLayout->addWidget(labelZoneArrivees);
00243     infoArriveesLayout->addStretch();
00244     infoArriveesLayout->addWidget(labelNbArriveesNonClassees);
00245     infoArriveesLayout->addWidget(QLCDNbArriveesNonClassees);
00246     infoArriveesLayout->addStretch();
00247     infoArriveesLayout->addWidget(labelNbArriveesClassees);
00248     infoArriveesLayout->addWidget(QLCDNbArriveesClassees);
00249     infoArriveesLayout->addStretch();
00250
00251     QVBoxLayout *arriveesLayout = new QVBoxLayout;
00252     arriveesLayout->addLayout(infoArriveesLayout);
00253     arriveesLayout->addWidget(vueListeTempsArriveesNonClassees);
00254     arriveesLayout->setContentsMargins(0, 0, 0, 10); // G H D B
00255
00256     QVBoxLayout *messageDossardLayout = new QVBoxLayout;
00257     messageDossardLayout->addStretch();
00258     messageDossardLayout->addWidget(labelMessageDossard);
00259     messageDossardLayout->addStretch();
00260
00261     QVBoxLayout *chronoLayout = new QVBoxLayout;
00262     chronoLayout->addLayout(courseLayout);
00263     chronoLayout->addLayout(QLCDChronoLayout);
00264     chronoLayout->addLayout(infoBoutonChronoLayout);
00265     chronoLayout->addLayout(arriveesLayout);
00266     chronoLayout->addWidget(labelMessageSupprimer);
00267     chronoLayout->addLayout(dossardLayout);
00268     chronoLayout->addLayout(messageDossardLayout);
00269
```

```
00270     QHBoxLayout *panneauLayout = new QHBoxLayout;
00271     panneauLayout->addLayout(classementLayout);
00272     panneauLayout->addLayout(chronoLayout);
00273
00274     QVBoxLayout *mainLayout = new QVBoxLayout;
00275     mainLayout->addLayout(listesLayout);
00276     mainLayout->addLayout(panneauLayout);
00277
00278     setLayout(mainLayout);
00279     setWindowTitle(tr("Chrono-Cross"));
00280     setContextMenuPolicy(Qt::ActionsContextMenu);
00281
00282     // les connexions
00283     connect(actionQuitter, SIGNAL(triggered()), this, SLOT(quitter()));
00284     connect(actionEntrerPAD, SIGNAL(triggered()), this, SLOT(
      associerArriveeDossard()));
00285     connect(actionEntrerRETURN, SIGNAL(triggered()), this, SLOT(
      associerArriveeDossard()));
00286
00287     connect(bSynchroniser, SIGNAL(clicked()), this, SLOT(
      preparerCourse()));
00288     connect(bLancer, SIGNAL(clicked()), this, SLOT(lancerCourse()));
00289     connect(bAssocier, SIGNAL(clicked()), this, SLOT(
      associerArriveeDossard()));
00290     connect(bArreter, SIGNAL(clicked()), this, SLOT(arreterCourse()));
00291     connect(bTerminer, SIGNAL(clicked()), this, SLOT(terminerCourse()));
00292
00293     connect(course, SIGNAL(chronoCreer()), this, SLOT(initialiserCourse()));
00294     connect(course, SIGNAL(chronoCoursePret()), this, SLOT(parer()));
00295     connect(course, SIGNAL(courseCommence()), this, SLOT(lancerChronoIHM()));
00296     connect(course, SIGNAL(classementArrete()), this, SLOT(arreterChrono()));
00297     connect(course, SIGNAL(courseFinie()), this, SLOT(terminerChrono()));
00298
00299     connect(course, SIGNAL(nouveauTempsArrivee(QString)), this, SLOT(
      ajouterArriveeCoureur(QString)));
00300
00301     connect(course, SIGNAL(informationCoureurRecuperees(QStringList)), this, SLOT(
      classerArrivee(QStringList)));
00302     connect(course, SIGNAL(chronoRecommence()), this, SLOT(
      commencerNouvelleCourse()));
00303
00304     connect(cbListeManifestations, SIGNAL(currentIndexChanged(QString)), this, SLOT(
      listerCourses(QString)));
00305
00306     connect(cbListeCourses, SIGNAL(currentIndexChanged(QString)), this, SLOT(
      afficherInformationsCourse(QString)));
00307     connect(cbListeCourses, SIGNAL(currentIndexChanged(QString)), this, SLOT(
      creerCourse(QString)));
00308
00309     connect(m_timer, SIGNAL(timeout()), this, SLOT(tic()));
00310
00311     listerManifestations();
00312
00313     setWindowFlags(Qt::Window | Qt::WindowCloseButtonHint);
00314     showMaximized();
00315 }
```

8.6.2.2 ~IHMChronoCross()

```
IHMChronoCross::~IHMChronoCross ( )
```

Destructeur de la fenêtre principale.

Références BaseDeDonnees : :detruireInstance().

```
00322 {
00323     BaseDeDonnees::detruireInstance();
00324     qDebug() << Q_FUNC_INFO;
00325 }
```

8.6.3 Documentation des fonctions membres

8.6.3.1 afficherInformationsCourse

```
void IHMChronoCross::afficherInformationsCourse (
            QString nomCourse )  [private], [slot]
```

SLOT afficherInformationsCourse() de la classe IHMChronoCross.

Affiche les informations d'une course

Paramètres

nomCourse	

Références course, Course : :getDistance(), Course : :getHeure(), Course : :getNbInscrit(), labelDistanceCourse, labelHeureCourse, labelNbInscrit, labelNomCourse, QLCDNbArriveesClassees, QLCDNbArriveesNonClassees, et Course : :setIdCourse().

Référencé par IHMChronoCross().

```
00908 {
00909     //information course
00910     if(nomCourse != "" )
00911     {
00912         if(nomCourse != "< Séléctionner une Course >")
00913         {
00914             // nom
00915             labelNomCourse->setText(QString("Nom : %1").arg(nomCourse));
00916 
00917             // distance
00918             int distance = course->getDistance(nomCourse);
00919             labelDistanceCourse->setText(QString("Distance : %1").arg(distance));
00920 
00921             // heure
00922             QString heure = course->getHeure(nomCourse);
00923             labelHeureCourse->setText(QString("Heure : %1").arg(heure));
00924 
00925             //setID
00926             course->setIdCourse(nomCourse);
00927         }
00928     }
00929 
00930     //information coureur course
00931     int nbInscrit = 0;
00932     if(nomCourse != "" )
00933     {
00934         if(nomCourse != "< Séléctionner une Course >")
00935         {
00936             nbInscrit = course->getNbInscrit(nomCourse);
00937             qDebug() << Q_FUNC_INFO << "nbInscrits : " << nbInscrit;
00938             labelNbInscrit->setText(QString("Inscrits : %1").arg(nbInscrit));
00939             QLCDNbArriveesClassees->display("00");
00940             QLCDNbArriveesNonClassees->display("00");
00941         }
00942     }
00943 }
```

8.6.3.2 ajouterArriveeCoureur

```
void IHMChronoCross::ajouterArriveeCoureur (
            QString tempsArrivee )  [private], [slot]
```

SLOT ajouterArriveeCoureur(QString tempsArrivee) de la classe IHMChronoCross.

Permet de récupérer la trame des coureurs, stocker les performances dans une variable et les afficher dans l'espace Arrivées non classées.

Références bAssocier, labelMessageSupprimer, lineEditNumeroDossard, mettreAJourNbArriveesNonClassees(), modeleArrivees↩NonClassees, et tempsArriveesNonClassees.

Référencé par IHMChronoCross().

```
00753 {
00754     bAssocier->setEnabled(true);
00755     lineEditNumeroDossard->setEnabled(true);
00756     lineEditNumeroDossard->setFocus();
00757 
00758     labelMessageSupprimer->setText("Pour supprimer le premier temps non classées \n
      Veuillez entrer le numéro de dossard 0000 puis confirmer.");
00759 
00760     // on ajoute le tempsArrivee au QStringList tempsArriveesNonClassees
```

```
00761     tempsArriveesNonClassees.push_back(tempsArrivee);
00762
00763     this->mettreAJourNbArriveesNonClassees(1);
00764
00765     int nbLignesListArriveesNonClassees = modeleArriveesNonClassees->rowCount();
00766     modeleArriveesNonClassees->insertRow(nbLignesListArriveesNonClassees);
00767     QModelIndex index = modeleArriveesNonClassees->index(
      nbLignesListArriveesNonClassees);
00768     modeleArriveesNonClassees->setData(index, QString(tempsArrivee));
00769 }
```

8.6.3.3 arreterChrono

```
void IHMChronoCross::arreterChrono ( )  [private], [slot]
```

SLOT arreterChrono() de la classe IHMChronoCross.

appele la méthode arreterChrono() de la classe Course

Références Course : :arreterChrono(), et course.

Référencé par IHMChronoCross().

```
00700 {
00701     course->arreterChrono();
00702 }
```

8.6.3.4 arreterCourse

```
void IHMChronoCross::arreterCourse ( )  [private], [slot]
```

SLOT arreterCourse() de la Classe IHMChronoCross.

Permet d'appeler la méthode arreterClassement() de la classe Course. On désactive le bouton arreter. On passe la course à l'état "Arretee" la led course passe au rouge.

Références Course : :arreterClassement(), bArreter, course, labelLedCourse, labelMessageDossard, m_timer, Course : :setEtat(), et setRouge().

Référencé par IHMChronoCross().

```
00685 {
00686     course->arreterClassement();
00687     course->setEtat("Arretee");
00688     setRouge(labelLedCourse);
00689     m_timer->stop();
00690     bArreter->setEnabled(false);
00691     labelMessageDossard->clear();
00692 }
```

8.6.3.5 associerArriveeDossard

```
void IHMChronoCross::associerArriveeDossard ( )  [private], [slot]
```

SLOT associerArriveeDossard() de la classe IHMChronoCross.

Selon le nombre entré dans le lineEdit on peut supprimer le premier temps de la liste des temps non classées et associer un temps et un dossard après vérification

Références Course : :ajouteArriveeBDD(), cbListeCourses, course, DOSSARD_DEJA_ARRIVE, DOSSARD_VALIDE, DOSSARD_VALIDE_COURSE_INVALIDE, Course : :getNomCourse(), initialiserConfirmationDialog(), labelMessageDossard, lineEditNumeroDossard, mettreAJourNbArriveesClassees(), mettreAJourNbArriveesNonClassees(), modeleArriveesNonClassees, NUMERO_DOSSARD_INVALIDE, tempsArriveesNonClassees, et Course : :verifierDossard().

Référencé par IHMChronoCross().

```
00777 {
00778     if(!tempsArriveesNonClassees.empty())
00779     {
00780         QString dossard = lineEditNumeroDossard->text();
00781         if(dossard != "0000")
00782         {
00783             QString tempsArrivee = tempsArriveesNonClassees.front();
00784             qDebug() << Q_FUNC_INFO << "Temps arrivee : " <<tempsArrivee;
00785             if(tempsArrivee != "")
00786             {
00787                 int verification = course->verifierDossard(dossard);
00788                 if(verification != NUMERO_DOSSARD_INVALIDE)
00789                 {
00790                     if(verification == DOSSARD_VALIDE_COURSE_INVALIDE)
00791                     {
00792                     // numéro de dossard valide mais inscrit pour une autre course
00793                         QString nomCourseInvalide = cbListeCourses->currentText();
00794                         QString nomCourseValire = course->getNomCourse(dossard);
00795                         labelMessageDossard->setStyleSheet("color : #FF0000");
00796                         labelMessageDossard->setText(QString("Numéro de dossard %1 :
      valide mais non-inscrit à la course %2.\nLe dossard %1 est inscrit pour la course %3.").arg(dossard).arg(
     nomCourseInvalide).arg(nomCourseValire));
00797                         lineEditNumeroDossard->clear();
00798                     }
00799                     else if(verification == DOSSARD_DEJA_ARRIVE)
00800                     {
00801                         // numéro de dossard valide mais le coureur a déjà franchi la ligne d'arrivée
00802                             QString nomCourse = cbListeCourses->currentText();
00803                             labelMessageDossard->setStyleSheet("color : #FF0000");
00804                             labelMessageDossard->setText(QString("Numéro de dossard %1 :
      valide mais le coureur a déjà franchi la ligne d'arrivée.").arg(dossard).arg(nomCourse));
00805                             lineEditNumeroDossard->clear();
00806                     }
00807                     else if(verification == DOSSARD_VALIDE)
00808                     {
00809                         if(labelMessageDossard->styleSheet() == "color : #FF0000")
00810                             labelMessageDossard->setStyleSheet("color : #000000");
00811
00812                         mettreAJourNbArriveesNonClassees(-1);
00813                         mettreAJourNbArriveesClassees(1);
00814
00815                         course->ajouteArriveeBDD(dossard, tempsArrivee);
00816
00817                         labelMessageDossard->setText(QString("Numéro de dossard %1 :
      validé\nLe temps %2 est associé au dossard %1").arg(dossard).arg(tempsArrivee));
00818
00819                         //enléve le premier temps et vide le lineEditeNumeroDossard
00820                         modeleArriveesNonClassees->removeRows(0,1);
00821                         tempsArriveesNonClassees.pop_front();
00822                         lineEditNumeroDossard->clear();
00823                     }
00824                 }
00825                 else
00826                 {
00827                     labelMessageDossard->setStyleSheet("color : #FF0000");
00828                     labelMessageDossard->setText(QString("Numéro de dossard %1 :
      invalide").arg(dossard));
00829                     lineEditNumeroDossard->clear();
00830                 }
00831             }
00832             else
00833             {
00834                 qDebug() << Q_FUNC_INFO << "Aucun numéro entré";
00835                 labelMessageDossard->setText("Aucun numéro de dossard entré");
00836             }
00837
```

```
00838         }
00839         else
00840         {
00841             lineEditNumeroDossard->clear();
00842             this->initialiserConfirmationDialog();
00843             qDebug() << Q_FUNC_INFO << "DEMANDE DE SUPPRESSION DU PREMIER TEMPS";
00844         }
00845     }
00846 }
```

8.6.3.6 classerArrivee

```
void IHMChronoCross::classerArrivee (
            QStringList informationCoureur )  [private], [slot]
```

SLOT classerArrivee() de la classe IHMChronoCross.

Paramètres

informationCoureur	

Permet de classer les arrivées dans le tableau classement, pour les 3 premiers temps on utilise la méthode personnaliserAffichage↩ Arrivee()

Références COLONNE_CLASSE, COLONNE_DOSSARD, COLONNE_NOM, COLONNE_PRENOM, COLONNE_TEMPS, INFO_↩ COUREUR_CLASSE, INFO_COUREUR_DOSSARD, INFO_COUREUR_NOM, INFO_COUREUR_PRENOM, INFO_COUREUR_↩ TEMPS, modeleClassement, nbLignesClassement, et personnaliserAffichageArrivee().

Référencé par IHMChronoCross().

```
00855 {
00856     qDebug() << Q_FUNC_INFO << informationCoureur;
00857     //informationCoureur[ temps, dossard, Nom, Prenom, Classe]
00858
00859     QStandardItem *temps = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_TEMPS));
00860     QStandardItem *dossard = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_DOSSARD));
00861     QStandardItem *nom = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_NOM));
00862     QStandardItem *prenom = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_PRENOM));
00863     QStandardItem *classe = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_CLASSE));
00864
00865     modeleClassement->setItem(nbLignesClassement,
      COLONNE_TEMPS, temps);
00866     modeleClassement->setItem(nbLignesClassement,
      COLONNE_DOSSARD, dossard);
00867     modeleClassement->setItem(nbLignesClassement,
      COLONNE_NOM, nom);
00868     modeleClassement->setItem(nbLignesClassement,
      COLONNE_PRENOM, prenom);
00869     modeleClassement->setItem(nbLignesClassement,
      COLONNE_CLASSE, classe);
00870
00871     qDebug() << Q_FUNC_INFO << nbLignesClassement;
00872
00873     if(nbLignesClassement < 3)
00874         personnaliserAffichageArrivee(nbLignesClassement);
00875
00876     nbLignesClassement += 1;
00877 }
```

8.6.3.7 commencerNouvelleCourse

```
void IHMChronoCross::commencerNouvelleCourse ( )  [private], [slot]
```

SLOT commencerNouvelleCourse() de la classe IHMChronoCross.

Reinitialise le QLCDChrono pour lancer une nouvelle course.

Références modeleClassement, nomColonnes, et vueTableauClassement.

Référencé par IHMChronoCross().

```
00741 {
00742     modeleClassement->clear();
00743     modeleClassement->setHorizontalHeaderLabels(nomColonnes);
00744     vueTableauClassement->setModel(modeleClassement);
00745 }
```

8.6.3.8 creerCourse

```
void IHMChronoCross::creerCourse (
            QString nomCourse )  [private], [slot]
```

Méthode creerCourse() de la classe IHMChronoCross.

Si l'utilisateur a selectionner une course valide, utilise la méthode creerChrono() de la classe Course

Références course, Course : :creerChrono(), et verifierCourseSelectionnee().

Référencé par IHMChronoCross().

```
00607 {
00608     qDebug() << Q_FUNC_INFO << verifierCourseSelectionnee(nomCourse);
00609     if(verifierCourseSelectionnee(nomCourse))
00610         course->creerChrono();
00611 }
```

8.6.3.9 getHeure()

```
long IHMChronoCross::getHeure ( )  [private]
```

Méthode getHeure() de la classe IHMChronoCross.

Renvoie

Retourne la valeur de m_valeur pour les heures

Références m_valeur.

Référencé par update().

```
00536 {
00537     return m_valeur / 36000;
00538 }
```

8.6.3.10 getMinute()

```
long IHMChronoCross::getMinute ( )  [private]
```

Méthode getMinute() de la classe IHMChronoCross.

Renvoie

Retourne la valeur de m_valeur pour les Minutes

Références m_valeur.

Référencé par update().

```
00526 {
00527     return (m_valeur % 36000) / 600;
00528 }
```

8.6.3.11 getSeconde()

```
long IHMChronoCross::getSeconde ( )  [private]
```

Méthode getSeconde() de la classe IHMChronoCross.

Renvoie

Retourne la valeur de m_valeur pour les secondes

Références m_valeur.

Référencé par update().

```
00516 {
00517     return (m_valeur / 10) % 60;
00518 }
```

8.6.3.12 initialiserConfirmationDialog()

```
void IHMChronoCross::initialiserConfirmationDialog ( )  [private]
```

Méthode initialiserConfirmationDialog() de la classe IHMChronoCross.

Initialise la page dialog de confirmation pour la suppression d'un temps

Références bAnnulerDialog, bConfirmationDialog, confirmationDialog, labelConfirmationDialog, quitterDialog(), supprimerPremierTemps(), et tempsArriveesNonClassees.

Référencé par associerArriveeDossard().

```
00439 {
00440     QString temps = tempsArriveesNonClassees.front();
00441     confirmationDialog = new QDialog(this);
00442     labelConfirmationDialog = new QLabel (tr("Etes vous sûr de vouloir supprimer le
       temps %1 ?").arg(temps));
00443     bConfirmationDialog = new QPushButton (QString::fromUtf8("Confirmer"));
00444     bAnnulerDialog = new QPushButton (QString::fromUtf8("Annuler"));
00445
00446     QHBoxLayout *boutonLayout = new QHBoxLayout;
00447     boutonLayout->addWidget(bConfirmationDialog);
00448     boutonLayout->addWidget(bAnnulerDialog);
00449     boutonLayout->setContentsMargins(0, 0, 0, 5); // G H D B
00450
00451     QVBoxLayout *mainDialogLayout = new QVBoxLayout;
00452     mainDialogLayout->addWidget(labelConfirmationDialog);
00453     mainDialogLayout->addLayout(boutonLayout);
00454     mainDialogLayout->setContentsMargins(10, 10, 10, 10); // G H D B
00455
00456     setWindowTitle(tr("Confirmation"));
00457
00458     connect(bAnnulerDialog, SIGNAL(clicked()), this, SLOT(
      quitterDialog()));
00459     connect(bConfirmationDialog, SIGNAL(clicked()), this, SLOT(
      supprimerPremierTemps()));
00460     qDebug() << Q_FUNC_INFO;
00461
00462     confirmationDialog->setLayout(mainDialogLayout);
00463     confirmationDialog->exec();
00464 }
```

8.6.3.13 initialiserCourse

```
void IHMChronoCross::initialiserCourse ( )  [private], [slot]
```

SLOT initialiserCourse() de la classe IHMChrono-Cross.

Si le chronomètre est pret alors on initialse l'état de la course dans la BDD à "prete", on change la led et on affiche le bouton Synchroniser

Références bSynchroniser, course, Course : :estChronometragePret(), labelLedChrono, Course : :setEtat(), et setOrange().

Référencé par IHMChronoCross().

```
00619 {
00620     if(course->estChronometragePret())
00621     {
00622         course->setEtat("Prete");
00623         setOrange(labelLedChrono);
00624         bSynchroniser->setEnabled(true);
00625     }
00626     else
00627         qDebug() << Q_FUNC_INFO << "Erreur";
00628 }
```

8.6.3.14 lancerChronoIHM

```
void IHMChronoCross::lancerChronoIHM ( )  [private], [slot]
```

SLOTS lancerChronoIHM() de la classe IHMChronoCross.

Permet d'initialiser le QLCDChrono, d'activer le bouton Arreter et de désactiver le bouton Chronometrer. On passe les leds à l'état vert.

Références bArreter, bLancer, labelLedChrono, labelLedCourse, m_timer, QLCDChrono, et setVert().

Référencé par IHMChronoCross().

```
00670 {
00671     QLCDChrono->display("00:00:00");
00672     m_timer->start(100);
00673     setVert(labelLedCourse);
00674     setVert(labelLedChrono);
00675     bLancer->setEnabled(false);
00676     bArreter->setEnabled(true);
00677 }
```

8.6.3.15 lancerCourse

```
void IHMChronoCross::lancerCourse ( )  [private], [slot]
```

SLOT lancerCourse() de la classe IHMChronoCross.

Change l'état de la course dans la BDD en passant en mode EnCours et utilise la méthode chronometrer de la classe Course.

Références Course : :chronometrer(), course, et Course : :setEtat().

Référencé par IHMChronoCross().

```
00659 {
00660     course->chronometrer();
00661     course->setEtat("EnCours");
00662 }
```

8.6.3.16 listerCourses

```
void IHMChronoCross::listerCourses (
            QString manifestation )  [private], [slot]
```

Méthode listerCourses() de la Classe IHMChronoCross.

Liste et d'affiche les courses à venir à partir d'une manifestation séléctionnée

Références cbListeCourses, course, et Course : :getListeCourses().

Référencé par IHMChronoCross().

```
00584 {
00585     QVector<QString> listeCourses = course->getListeCourses(manifestation);
00586     int nbCourses = listeCourses.size();
00587
00588     //si l'utilisateur change de manifestation on vide la liste course
00589     if(cbListeCourses->count() != 0)
00590     {
00591         cbListeCourses->clear();
00592         cbListeCourses->addItem(tr("< Séléctionner une Course >"));
00593     }
00594
00595     for(int i = 0; i < nbCourses; i += 1)
00596     {
00597         cbListeCourses->addItem(listeCourses[i]);
00598     }
00599 }
```

8.6.3.17 listerManifestations()

```
void IHMChronoCross::listerManifestations ( )  [private]
```

Méthode listerManifestations() de la Classe IHMChronoCross.

Lister les manifestations à venir

Références cbListeManifestations, course, et Course : :getListeManifestations().

Référencé par IHMChronoCross().

```
00333 {
00334     QStringList listeManifestation = course->getListeManifestations();
00335     int nbManifestations = listeManifestation.size();
00336     for(int i = 0; i < nbManifestations; i += 1)
00337     {
00338         cbListeManifestations->addItem(listeManifestation[i]);
00339     }
00340 }
```

8.6.3.18 mettreAJourNbArriveesClassees

```
void IHMChronoCross::mettreAJourNbArriveesClassees (
            int nbArrivee )  [private], [slot]
```

SLOT mettreAJourNbArriveesClassees() de la classe IHMChronoCross.

Permet de mettre à jour le nombre d'arrivées non classées

Paramètres

nbArrivee	

Références nbArriveesClassees, nbCoureurArrive, et QLCDNbArriveesClassees.

Référencé par associerArriveeDossard().

```
00971 {
00972     nbCoureurArrive += nbArrivee;
00973     nbArriveesClassees = nbCoureurArrive;
00974     if(nbArriveesClassees < 10)
00975     {
00976         QString strNbArriveesClassees = QString::number(nbArriveesClassees);
00977         strNbArriveesClassees = "0" + strNbArriveesClassees;
00978         QLCDNbArriveesClassees->display(strNbArriveesClassees);
00979         qDebug() << Q_FUNC_INFO << "nbArriveesClassees : " << nbArriveesClassees;
00980     }
00981     else
00982         QLCDNbArriveesClassees->display(nbArriveesClassees);
00983         qDebug() << Q_FUNC_INFO << "nbArriveesClassees : " << nbArriveesClassees;
00984 }
```

8.6.3.19 mettreAJourNbArriveesNonClassees

```
void IHMChronoCross::mettreAJourNbArriveesNonClassees (
             int nbArrivee )  [private], [slot]
```

SOT mettreAJourNbArriveesNonClassees() de la classe IHMChronoCross.

Permet de mettre à jour le nombre d'arrivées non classées

Paramètres

nbArrivee	

Références nbArriveesNonClassees, et QLCDNbArriveesNonClassees.

Référencé par ajouterArriveeCoureur(), associerArriveeDossard(), et supprimerPremierTemps().

```
00952 {
00953     nbArriveesNonClassees += nbArrivee;
00954     if(nbArriveesNonClassees < 10)
00955     {
00956         QString strNbArriveesNonClassees = QString::number(nbArriveesNonClassees);
00957         strNbArriveesNonClassees = "0" + strNbArriveesNonClassees;
00958         QLCDNbArriveesNonClassees->display(strNbArriveesNonClassees);
00959     }
00960     else
00961         QLCDNbArriveesNonClassees->display(
      nbArriveesNonClassees);
00962 }
```

8.6.3.20 parer

```
void IHMChronoCross::parer ( )  [private], [slot]
```

Private SLOT parer() de la classe IHMChronoCross.

Affiche la led Orange pour la course, on désactive le bouton synchroniser et on active le bouton lancer

Références bLancer, bSynchroniser, labelLedCourse, et setOrange().

Référencé par IHMChronoCross().

```
00647 {
00648     setOrange(labelLedCourse);
00649     bSynchroniser->setEnabled(false);
00650     bLancer->setEnabled(true);
00651 }
```

8.6.3.21 personnaliserAffichageArrivee()

```
void IHMChronoCross::personnaliserAffichageArrivee (
              int nbLignesClassement )  [private]
```

Méthode personnaliserAffichageArrivee() de la classe IHMChronoCross.

Personnalise les trois premières lignes du classement en or argent et bronze

Paramètres

nbLignesClassement	

Références modeleClassement, et TAILLETEXTECLASSEMENT.

Référencé par classerArrivee().

```
00473 {
00474     QFont texteClassement;
00475     texteClassement.setPointSize(TAILLETEXTECLASSEMENT);
00476     texteClassement.setBold(true);
00477
00478     switch(nbLignesClassement)
00479     {
00480         case 0:
00481             for(int i = 0; i < 5; i+=1)
00482             {
00483                 QStandardItem *item = modeleClassement->item(
      nbLignesClassement, i);
00484                 item->setBackground(QColor(255,223,0));
00485                 item->setFont(texteClassement);
00486             }
00487         break;
00488
00489         case 1:
00490             for(int i = 0; i < 5; i+=1)
00491             {
00492                 QStandardItem *item = modeleClassement->item(
      nbLignesClassement, i);
00493                 item->setBackground(QColor(192,192,192));
00494                 item->setFont(texteClassement);
00495             }
00496         break;
00497
00498         case 2:
00499             for(int i = 0; i < 5; i+=1)
00500             {
00501                 QStandardItem *item = modeleClassement->item(
      nbLignesClassement, i);
00502                 item->setBackground(QColor(205,127,50));
00503                 item->setFont(texteClassement);
00504             }
00505         break;
00506
00507     }
00508 }
```

8.6.3.22 preparerCourse

```
void IHMChronoCross::preparerCourse ( )  [private], [slot]
```

Slots preparerCourse() de la classe IHMChronoCross.

Utilise la méthode preparerChrono() de la classe Course

Références course, et Course : :preparerChrono().

Référencé par IHMChronoCross().

```
00636 {
00637     qDebug() << Q_FUNC_INFO << "\nbDemarrer\n";
00638     course->preparerChrono();
00639 }
```

8.6.3.23 quitter

```
void IHMChronoCross::quitter ( )  [private], [slot]
```

SLOT quitter() de la classe IHMChronoCross.

Action permet de fermer la page

Référencé par IHMChronoCross().

```
00885 {
00886     close();
00887 }
```

8.6.3.24 quitterDialog

```
IHMChronoCross::quitterDialog ( )  [private], [slot]
```

SLOT quitterDialog() de la classe IHMChronoCross.

Permet de fermer la page dialog Confirmation

Références confirmationDialog.

Référencé par initialiserConfirmationDialog().

```
00992 {
00993     confirmationDialog->close();
00994     qDebug() << Q_FUNC_INFO;
00995 }
```

8.6.3.25 reinitialiserClassement()

```
void IHMChronoCross::reinitialiserClassement ( )  [private]
```

Méthode reinitialiserClassement() de la classe IHMChronoCross.

vide le classement mais conserve l'entête des colonnes

Références modeleClassement, et nomColonnes.

Référencé par terminerCourse().

```
00428 {
00429         modeleClassement->clear();
00430         modeleClassement->setHorizontalHeaderLabels(nomColonnes);
00431 }
```

8.6.3.26 reinitialiserInfoCourse()

```
void IHMChronoCross::reinitialiserInfoCourse ( )  [private]
```

Méthode reinitialiserInfoCourse() de la classe IHMChronoCross.

Permet de réinitialiser les informations de la course sur l'IHM

Références labelDistanceCourse, labelHeureCourse, labelNbInscrit, labelNomCourse, QLCDNbArriveesClassees, et QLCDNb↩ ArriveesNonClassees.

Référencé par terminerCourse().

```
00412 {
00413     labelNomCourse->setText("Nom :");
00414     labelDistanceCourse->setText("Distance :");
00415     labelHeureCourse->setText("Heure :");
00416     labelNbInscrit->setText("Inscrits :");
00417
00418     QLCDNbArriveesClassees->display("--");
00419     QLCDNbArriveesNonClassees->display("--");
00420 }
```

8.6.3.27 setOrange()

```
void IHMChronoCross::setOrange (
            QLabel * label )  [private]
```

Méthode setOrange() de la classe IHMChronoCross.

Initialise l'image ledOrange

Paramètres

label	de l'image

Référencé par initialiserCourse(), et parer().

```
00399 {
00400     QImage image;
00401     image.load(":/images/orange.png");
00402     QPixmap pixmap = QPixmap::fromImage(image);
00403     label->setPixmap(pixmap);
00404 }
```

8.6.3.28 setRouge()

```
void IHMChronoCross::setRouge (
            QLabel * label )  [private]
```

Méthode setRouge() de la classe IHMChronoCross.

Initialise l'image ledRouge

Paramètres

label	de l'image

Référencé par arreterCourse(), IHMChronoCross(), et terminerCourse().

```
00371 {
00372     QImage image;
00373     image.load(":/images/rouge.png");
00374     QPixmap pixmap = QPixmap::fromImage(image);
00375     label->setPixmap(pixmap);
00376 }
```

8.6.3.29 setVert()

```
void IHMChronoCross::setVert (
            QLabel * label )  [private]
```

Méthode setVert() de la classe IHMChronoCross.

Initialise l'image ledVerte

Paramètres

label	de l'image

Référencé par lancerChronoIHM().

```
00385 {
00386     QImage image;
00387     image.load(":/images/vert.png");
00388     QPixmap pixmap = QPixmap::fromImage(image);
00389     label->setPixmap(pixmap);
00390 }
```

8.6.3.30 supprimerPremierTemps

```
void IHMChronoCross::supprimerPremierTemps ( )  [private], [slot]
```

SLOT supprimerPremierTemps() de la classe IHMChronoCross.

Permet de supprimer le permier temps de la liste des temps non classées.

Références confirmationDialog, mettreAJourNbArriveesNonClassees(), modeleArriveesNonClassees, et tempsArriveesNonClassees.

Référencé par initialiserConfirmationDialog().

```
01003 {
01004     modeleArriveesNonClassees->removeRows(0,1);
01005     tempsArriveesNonClassees.pop_front();
01006     confirmationDialog->close();
01007     this->mettreAJourNbArriveesNonClassees(-1);
01008 }
```

8.6.3.31 terminerChrono

```
void IHMChronoCross::terminerChrono ( )  [private], [slot]
```

SLOT terminerChrono() de la classe IHMChronoCross.

Permet de désactiver le QLCD et d'activer le bouton terminer

Références bTerminer, m_timer, et QLCDChrono.

Référencé par IHMChronoCross().

```
00710 {
00711     QLCDChrono->display("--:--:--");
00712     m_timer->stop();
00713     bTerminer->setEnabled(true);
00714 }
```

8.6.3.32 terminerCourse

```
void IHMChronoCross::terminerCourse ( )  [private], [slot]
```

Private SLOT terminerCourse() de la classe IHMChronoCross.

Permet de passer la led chrono à l'état rouge et réinitialser les informations course affiché

Références course, labelLedChrono, m_valeur, nbArriveesClassees, nbArriveesNonClassees, nbCoureurArrive, nbLignes↩Classement, reinitialiserClassement(), reinitialiserInfoCourse(), Course : :setEtat(), et setRouge().

Référencé par IHMChronoCross().

```
00722 {
00723    setRouge(labelLedChrono);
00724    course->setEtat("Terminee");
00725    reinitialiserInfoCourse();
00726    reinitialiserClassement();
00727    nbArriveesNonClassees = 0;
00728    nbLignesClassement = 0;
00729    nbArriveesClassees = 0;
00730    nbCoureurArrive = 0;
00731    m_valeur = 0;
00732    qDebug() << Q_FUNC_INFO << "on recommence";
00733 }
```

8.6.3.33 tic

```
void IHMChronoCross::tic ( )  [private], [slot]
```

Méthode tic() de la classe IHMChronoCross.

Ajoute un au chrono interne de l'IHM et utilise la méthode update

Références m_valeur, et update().

Référencé par IHMChronoCross().

```
00895 {
00896     qDebug() << Q_FUNC_INFO << m_valeur;
00897     m_valeur++;
00898     update();
00899 }
```

8.6.3.34 update()

```
void IHMChronoCross::update ( )  [private]
```

Méthode update() de la classe IHMChronoCross.

Permet de mettre à jour l'horloge interne de l'IHM

Références getHeure(), getMinute(), getSeconde(), m_valeur, et QLCDChrono.

Référencé par tic().

```
00546 {
00547     QString heure, minute, seconde;
00548
00549     if (getHeure() < 10)
00550     {
00551         heure = "0" + QString::number(getHeure());
00552     }
00553     else heure = QString::number(getHeure());
00554
00555     if (getMinute() < 10)
00556     {
00557         minute = "0" + QString::number(getMinute());
00558     }
00559     else minute = QString::number(getMinute());
00560
00561     if (getSeconde() < 10)
00562     {
00563         seconde = "0" + QString::number(getSeconde());
00564     }
00565     else seconde = QString::number(getSeconde());
00566
00567     if (getHeure() == 24)
00568     {
00569         m_valeur = 0;
00570     }
00571
00572     QString text = heure + ":" + minute + ":" + seconde;
00573     QLCDChrono->display(text);
00574 }
```

8.6.3.35 verifierCourseSelectionnee()

```
bool IHMChronoCross::verifierCourseSelectionnee (
            QString courseSelectionnee )  [private]
```

Méthode verifierCourseSelectionnee() de la classe IHMChronoCross.

Paramètres

courseSelectionnee	

Renvoie

Référencé par creerCourse().

```
00350 {
00351     if(!courseSelectionnee.isEmpty())
00352     {
00353         if(courseSelectionnee != "< Séléctionner une Course >")
00354         {
00355             return true;
00356         }
00357         else
00358             return false;
00359     }
00360     else
00361         return false;
00362 }
```

8.6.4 Documentation des données membres

8.6.4.1 acquitement

```
bool IHMChronoCross::acquitement  [private]
```

8.6.4.2 bAnnulerDialog

```
QPushButton* IHMChronoCross::bAnnulerDialog  [private]
```

Référencé par initialiserConfirmationDialog().

8.6.4.3 bArreter

```
QPushButton* IHMChronoCross::bArreter  [private]
```

Bouton Arreter.

Référencé par arreterCourse(), IHMChronoCross(), et lancerChronoIHM().

8.6.4.4 bAssocier

```
QPushButton* IHMChronoCross::bAssocier  [private]
```

Bouton Associer.

Référencé par ajouterArriveeCoureur(), et IHMChronoCross().

8.6.4.5 bConfirmationDialog

```
QPushButton* IHMChronoCross::bConfirmationDialog  [private]
```

Référencé par initialiserConfirmationDialog().

8.6.4.6 bLancer

```
QPushButton* IHMChronoCross::bLancer  [private]
```

Bouton Lancer le chronomètre.

Référencé par IHMChronoCross(), lancerChronoIHM(), et parer().

8.6.4.7 bSynchroniser

```
QPushButton* IHMChronoCross::bSynchroniser  [private]
```

Bouton Synchroniser.

Référencé par IHMChronoCross(), initialiserCourse(), et parer().

8.6.4.8 bTerminer

```
QPushButton* IHMChronoCross::bTerminer  [private]
```

Bouton Terminer.

Référencé par IHMChronoCross(), et terminerChrono().

8.6.4.9 cbListeCourses

```
QComboBox* IHMChronoCross::cbListeCourses  [private]
```

Combobox contenant les différentes course de la base de donnée.

Référencé par associerArriveeDossard(), IHMChronoCross(), et listerCourses().

8.6.4.10 cbListeManifestations

```
QComboBox* IHMChronoCross::cbListeManifestations  [private]
```

Combobox contenant les différentes manifesations de la base de donnée.

Référencé par IHMChronoCross(), et listerManifestations().

8.6.4.11 classement

```
QStringList IHMChronoCross::classement  [private]
```

Liste du classement pour une course.

Référencé par IHMChronoCross().

8.6.4.12 confirmationDialog

```
QDialog* IHMChronoCross::confirmationDialog  [private]
```

Référencé par initialiserConfirmationDialog(), quitterDialog(), et supprimerPremierTemps().

8.6.4.13 course

```
Course* IHMChronoCross::course  [private]
```

agrégation d'une course

Référencé par afficherInformationsCourse(), arreterChrono(), arreterCourse(), associerArriveeDossard(), creerCourse(), IHM↩ ChronoCross(), initialiserCourse(), lancerCourse(), listerCourses(), listerManifestations(), preparerCourse(), et terminerCourse().

8.6.4.14 courses

```
QVector<QStringList> IHMChronoCross::courses  [private]
```

liste des courses pour une manifestation

8.6.4.15 labelConfirmationDialog

```
QLabel* IHMChronoCross::labelConfirmationDialog  [private]
```

Référencé par initialiserConfirmationDialog().

8.6.4.16 labelDistanceCourse

```
QLabel* IHMChronoCross::labelDistanceCourse  [private]
```

Référencé par afficherInformationsCourse(), IHMChronoCross(), et reinitialiserInfoCourse().

8.6.4.17 labelEtatChrono

```
QLabel* IHMChronoCross::labelEtatChrono  [private]
```

Label contenant le texte "Etat : " pour la led chrono.

Référencé par IHMChronoCross().

8.6.4.18 labelEtatCourse

```
QLabel* IHMChronoCross::labelEtatCourse  [private]
```

Label contenant le texte "Etat : " pour la led course.

Référencé par IHMChronoCross().

8.6.4.19 labelHeureCourse

```
QLabel* IHMChronoCross::labelHeureCourse  [private]
```

Référencé par afficherInformationsCourse(), IHMChronoCross(), et reinitialiserInfoCourse().

8.6.4.20 labelLedChrono

```
QLabel* IHMChronoCross::labelLedChrono  [private]
```

Label contenant la LED (état vert, orange ou rouge) qui représente l'état de la connexion avec le TAG HEUER.

Référencé par IHMChronoCross(), initialiserCourse(), lancerChronoIHM(), et terminerCourse().

8.6.4.21 labelLedCourse

```
QLabel* IHMChronoCross::labelLedCourse  [private]
```

Label contenant la LED (état vert, orange ou rouge) qui représente l'état du chrono, si il est prêt à faire une course.

Référencé par arreterCourse(), IHMChronoCross(), lancerChronoIHM(), et parer().

8.6.4.22 labelListeCourses

```
QLabel* IHMChronoCross::labelListeCourses  [private]
```

Label contenant le QString "Courses : " devant cbListeCourses.

Référencé par IHMChronoCross().

8.6.4.23 labelManifestations

```
QLabel* IHMChronoCross::labelManifestations  [private]
```

Label contenant le QString "Manifestations : " devant cbListeManifestations.

Référencé par IHMChronoCross().

8.6.4.24 labelMessageDossard

```
QLabel* IHMChronoCross::labelMessageDossard  [private]
```

Label du message d'érreur pour les numéros de dossards.

Référencé par arreterCourse(), associerArriveeDossard(), et IHMChronoCross().

8.6.4.25 labelMessageSupprimer

```
QLabel* IHMChronoCross::labelMessageSupprimer  [private]
```

label contenant le message d'information pour supprimer le premier temps non classée

Référencé par ajouterArriveeCoureur(), et IHMChronoCross().

8.6.4.26 labelNbArriveesClassees

```
QLabel* IHMChronoCross::labelNbArriveesClassees  [private]
```

Label contentant le texte "Nombre d'arrivées classées :".

Référencé par IHMChronoCross().

8.6.4.27 labelNbArriveesNonClassees

```
QLabel* IHMChronoCross::labelNbArriveesNonClassees  [private]
```

Label contentant le texte "Nombre d'arrivées non classées :".

Référencé par IHMChronoCross().

8.6.4.28 labelNbInscrit

```
QLabel* IHMChronoCross::labelNbInscrit  [private]
```

Label contentant le texte "Nombre d'inscrit :".

Référencé par afficherInformationsCourse(), IHMChronoCross(), et reinitialiserInfoCourse().

8.6.4.29 labelNomCourse

```
QLabel* IHMChronoCross::labelNomCourse  [private]
```

Référencé par afficherInformationsCourse(), IHMChronoCross(), et reinitialiserInfoCourse().

8.6.4.30 labelNumeroDossard

```
QLabel* IHMChronoCross::labelNumeroDossard  [private]
```

Label contenant le QString "N° dossard : ".

Référencé par IHMChronoCross().

8.6.4.31 labelZoneArrivees

```
QLabel* IHMChronoCross::labelZoneArrivees  [private]
```

Label contenant le QString "Arrivées : " symbolisant la zone des arrivées non classées.

Référencé par IHMChronoCross().

8.6.4.32 labelZoneChrono

```
QLabel* IHMChronoCross::labelZoneChrono  [private]
```

Label contenant le texte "Chrono : " symbolisant la zone chrono.

Référencé par IHMChronoCross().

8.6.4.33 labelZoneClassement

```
QLabel* IHMChronoCross::labelZoneClassement  [private]
```

Label contenant le texte : Classement symbolisant la zone classement.

Référencé par IHMChronoCross().

8.6.4.34 labelZoneCourse

```
QLabel* IHMChronoCross::labelZoneCourse  [private]
```

Label contenant le texte : Course symbolisant la zone course.

Référencé par IHMChronoCross().

8.6.4.35 lineEditNumeroDossard

```
QLineEdit* IHMChronoCross::lineEditNumeroDossard  [private]
```

LineEdit qui permet de rentrer un numéro de dossard.

Référencé par ajouterArriveeCoureur(), associerArriveeDossard(), et IHMChronoCross().

8.6.4.36 logoChronoCross

```
QLabel* IHMChronoCross::logoChronoCross  [private]
```

Label contenant l'image du logo Chrono-Cross.

Référencé par IHMChronoCross().

8.6.4.37 m_timer

```
QTimer* IHMChronoCross::m_timer  [private]
```

Chrono de l'ihm.

Référencé par arreterCourse(), IHMChronoCross(), lancerChronoIHM(), et terminerChrono().

8.6.4.38 m_valeur

```
long IHMChronoCross::m_valeur  [private]
```

Valeur du chrono.

Référencé par getHeure(), getMinute(), getSeconde(), IHMChronoCross(), terminerCourse(), tic(), et update().

8.6.4.39 modeleArriveesNonClassees

```
QStringListModel* IHMChronoCross::modeleArriveesNonClassees  [private]
```

Model des arrivées non classés.

Référencé par ajouterArriveeCoureur(), associerArriveeDossard(), IHMChronoCross(), et supprimerPremierTemps().

8.6.4.40 modeleClassement

```
QStandardItemModel* IHMChronoCross::modeleClassement  [private]
```

Model du classement.

Référencé par classerArrivee(), commencerNouvelleCourse(), IHMChronoCross(), personnaliserAffichageArrivee(), et reinitialiserClassement().

8.6.4.41 nbArriveesClassees

```
int IHMChronoCross::nbArriveesClassees  [private]
```

Référencé par IHMChronoCross(), mettreAJourNbArriveesClassees(), et terminerCourse().

8.6.4.42 nbArriveesNonClassees

```
int IHMChronoCross::nbArriveesNonClassees  [private]
```

Référencé par IHMChronoCross(), mettreAJourNbArriveesNonClassees(), et terminerCourse().

8.6.4.43 nbCoureurArrive

```
int IHMChronoCross::nbCoureurArrive  [private]
```

Nombre de coureurs arrivés.

Référencé par IHMChronoCross(), mettreAJourNbArriveesClassees(), et terminerCourse().

8.6.4.44 nbLignesClassement

```
int IHMChronoCross::nbLignesClassement  [private]
```

nombre de lignes du classement

Référencé par classerArrivee(), IHMChronoCross(), et terminerCourse().

8.6.4.45 nomColonnes

```
QStringList IHMChronoCross::nomColonnes  [private]
```

Liste de nom des colonnes du classement "Temps, dossard, nom...".

Référencé par commencerNouvelleCourse(), IHMChronoCross(), et reinitialiserClassement().

8.6.4.46 QLCDChrono

```
QLCDNumber* IHMChronoCross::QLCDChrono  [private]
```

QLCD contenant le chrono.

Référencé par IHMChronoCross(), lancerChronoIHM(), terminerChrono(), et update().

8.6.4.47 QLCDNbArriveesClassees

```
QLCDNumber* IHMChronoCross::QLCDNbArriveesClassees  [private]
```

QLCD qui affiche le nombre d'arrivées classées.

Référencé par afficherInformationsCourse(), IHMChronoCross(), mettreAJourNbArriveesClassees(), et reinitialiserInfoCourse().

8.6.4.48 QLCDNbArriveesNonClassees

```
QLCDNumber* IHMChronoCross::QLCDNbArriveesNonClassees  [private]
```

QLCD qui affiche le nombre d'arrivées non classées.

Référencé par afficherInformationsCourse(), IHMChronoCross(), mettreAJourNbArriveesNonClassees(), et reinitialiserInfoCourse().

8.6.4.49 tempsArriveesNonClassees

```
QStringList IHMChronoCross::tempsArriveesNonClassees  [private]
```

Liste des arrivées non classés.

Référencé par ajouterArriveeCoureur(), associerArriveeDossard(), initialiserConfirmationDialog(), et supprimerPremierTemps().

8.6.4.50 vueListeTempsArriveesNonClassees

```
QListView* IHMChronoCross::vueListeTempsArriveesNonClassees  [private]
```

Vue en tableau des temps non classés.

Référencé par IHMChronoCross().

8.6.4.51 vueTableauClassement

```
QTableView* IHMChronoCross::vueTableauClassement  [private]
```

Vue en liste classement.

Référencé par commencerNouvelleCourse(), et IHMChronoCross().

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

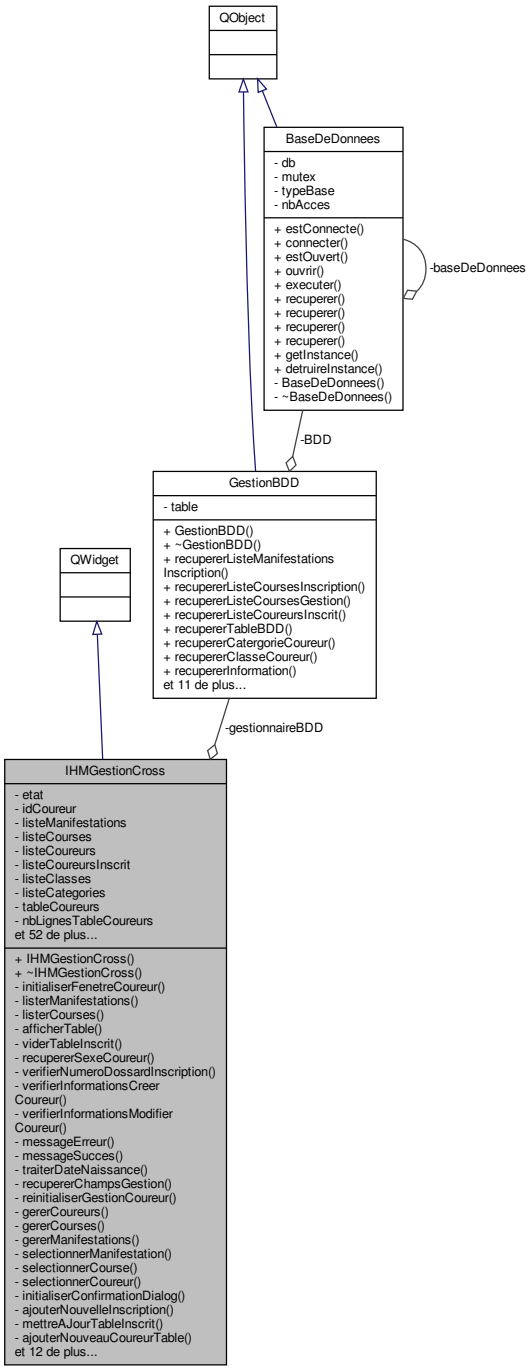
- ihmchronocross.h
- ihmchronocross.cpp
- ihmgestioncross.cpp

8.7 Référence de la classe IHMGestionCross

La fenêtre principale de l'application Gestion-Cross.

`#include <ihmgestioncross.h>`

Graphe de collaboration de IHMGestionCross :



Fonctions membres publiques

— IHMGestionCross (QWidget *parent=nullptr)
Constructeur de la fenêtre principale.
— ~IHMGestionCross ()

Connecteurs privés

- void gererCoureurs ()
  SLOT gererCoureurs() de la classe IHMGestionCross.
- void gererCourses ()
  SLOT gererCourses() de la classe IHMGestionCross.
- void gererManifestations ()
  SLOT gererManifestations() de la classe IHMGestionCross.
- void selectionnerManifestation (QString nom)
- void selectionnerCourse (QString nom)
- void selectionnerCoureur (QModelIndex index)
- void initialiserConfirmationDialog (QString nomTache)
- void ajouterNouvelleInscription ()
- void mettreAJourTableInscrit (QStringList inscription)
- void ajouterNouveauCoureurTable (QStringList informationCoureur)
- void supprimerCoureurTable ()
- void modifierCoureurTable ()
- void mettreAJourTableCoureur ()
- void passerModeNouveauCoureur ()
- void creerCoureur ()
- void supprimerCoureur ()
- void modifierCoureur ()
- void initialiserConfirmationDialog ()
- void annulerNouveauCoureur ()
- void confirmerDialog ()
- void quitterDialog ()
- void quitter ()
  SLOT quitter() de la classe IHMGestionCross.

Fonctions membres privées

- void initialiserFenetreCoureur ()
- void listerManifestations ()
- void listerCourses (QString idManifestation)
- void afficherTable (QString nomTable)
  Méthode afficherTable() de la classe IHMGestionCross.
- void viderTableInscrit ()
- QString recupererSexeCoureur ()
- bool verifierNumeroDossardInscription (QString numeroDossard)
- bool verifierInformationsCreerCoureur (QStringList informations)
- bool verifierInformationsModifierCoureur (QStringList informations)
- void messageErreur (QString message)
- void messageSucces (QString message)
- QVector< int > traiterDateNaissance (QString dateNaissance)
- QStringList recupererChampsGestion ()
- void reinitialiserGestionCoureur ()

Attributs privés

- QString etat
- QString idCoureur
- GestionBDD * gestionnaireBDD
- QVector< QString > listeManifestations
- QVector< QString > listeCourses
- QVector< QStringList > listeCoureurs
- QVector< QStringList > listeCoureursInscrit
- QVector< QStringList > listeClasses
- QVector< QStringList > listeCategories
- QVector< QStringList > tableCoureurs
- int nbLignesTableCoureurs
- int nbLignesTableInscrit
- QStringList nomColonnesInscrit
- QDate dateDefault
- QStackedWidget * fenetreGestionCross
- QWidget * fenetreManifestation
- QWidget * fenetreCourse
- QWidget * fenetreCoureur
- QPushButton * bManifestations
- QPushButton * bCourses
- QPushButton * bCoureurs
- QLabel * labelGestion
- QLabel * labelInscription
- QLabel * logoChronoCross

— QLabel * labelGestionNom
— QLabel * labelGestionPrenom
— QLabel * labelGestionDateNaissance
— QLabel * labelGestionClasse
— QLabel * labelGestionCategorie
— QLabel * labelGestionINE
— QLabel * labelGestionSexe
— QLabel * labelGestionParticipe
— QPushButton * bGestionNouveau
— QPushButton * bCreationConfirmer
— QPushButton * bCreationAnnuler
— QPushButton * bGestionModifier
— QPushButton * bGestionSupprimer
— QPushButton * bInscrire
— QDateEdit * deDateNaissance
— QLineEdit * lineEditNom
— QLineEdit * lineEditPrenom
— QComboBox * cbGestionClasse
— QComboBox * cbGestionCategorie
— QComboBox * cbGestionParticipe
— QLineEdit * lineEditINE
— QRadioButton * rbGestionSexeF
— QRadioButton * rbGestionSexeM
— QLabel * labelInscriptionManifestation
— QComboBox * cbInscriptionListeManifestation
— QLabel * labelInscriptionCourse
— QComboBox * cbInscriptionListeCourse
— QLabel * labelNumeroDossard
— QLineEdit * lineEditNumeroDossard
— QTableView * vueTableCoureurs
— QStandardItemModel * modeleTableCoureurs
— QStringList nomColonnesCoureur
— QTableView * vueTableInscrits
— QStandardItemModel * modeleTableInscrits
— QLabel * labelMessageInscription
— QDialog * confirmationDialog
— QLabel * labelConfirmationDialog
— QPushButton * bConfirmationDialog
— QPushButton * bAnnulerDialog

8.7.1 Description détaillée

La fenêtre principale de l'application Gestion-Cross.

Auteur

ANDREO Michaël

Version

1.0

8.7.2 Documentation des constructeurs et destructeur

8.7.2.1 IHMGestionCross()

```
IHMGestionCross::IHMGestionCross (
            QWidget * parent = nullptr )
```

Constructeur de la fenêtre principale.

Paramètres

parent	

Références ajouterNouveauCoureurTable(), ajouterNouvelleInscription(), annulerNouveauCoureur(), bCoureurs, bCourses, bCreationAnnuler, bCreationConfirmer, bGestionModifier, bGestionNouveau, bGestionSupprimer, bInscrire, bManifestations, cbInscriptionListeCourse, cbInscriptionListeManifestation, creerCoureur(), dateDefault, fenetreCoureur, fenetreCourse, fenetreGestionCross, fenetreManifestation, gererCoureurs(), gererCourses(), gererManifestations(), gestionnaireBDD, IMAGECHRONOCROSS, initialiserFenetreCoureur(), logoChronoCross, mettreAJourTableInscrit(), modifierCoureur(), modifierCoureurTable(), nbLignesTableCoureurs, passerModeNouveauCoureur(), quitter(), selectionnerCoureur(), selectionnerCourse(), selectionnerManifestation(), supprimerCoureur(), supprimerCoureurTable(), TAILLETEXTEBOUTONTITRE, et vueTableCoureurs.

```
00015                                                 : QWidget(parent)
00016 {
00017     qDebug() << Q_FUNC_INFO;
00018
00019     gestionnaireBDD = new GestionBDD(this);
00020     nbLignesTableCoureurs = 0;
00021     QVector<int> valeurDateDefault;
00022     valeurDateDefault << 2000 << 01 << 01;
00023     dateDefault.setDate(valeurDateDefault[0], valeurDateDefault[1], valeurDateDefault[2]);
00024     qDebug() << Q_FUNC_INFO << dateDefault;
00025
00026     setStyleSheet(QLatin1String("QWidget\n"
00027     "{\n"
00028     "   background-color: white;\n"
00029     "}\n"
00030     "QPushButton {\n"
00031     "    background-color: lightgray;\n"
00032     "    border-width: 1px;\n"
00033     "    border-color: black;\n"
00034     "    border-style: solid;\n"
00035     "    border-radius: 1;\n"
00036     "    padding: 5px;\n"
00037     "    min-width: 9ex;\n"
00038     "    min-height: 2.5ex;\n"
00039     "}\n"
00040     "\n"
00041     "QPushButton:hover:enabled {\n"
00042     "   background-color: gray;\n"
00043     "}\n"
00044     "\n"
00045     "QPushButton:pressed {\n"
00046     "    padding-left: 5px;\n"
00047     "    padding-top: 5px;\n"
00048     "    background-color: white;\n"
00049     "}\n"
00050     "\n"
00051     "QPushButton:checked {\n"
00052     "background-color: white;\n"
00053     "}\n"
00054     "QComboBox, QLineEdit {\n"
00055     "    background-color: white;\n"
00056     "    selection-color: #0a214c; \n"
00057     "    selection-background-color: white;\n"
00058     "    border-width: 1px;\n"
00059     "    padding: 3px;\n"
00060     "    border-style: solid;\n"
00061     "    border-color: gray;\n"
00062     "    border-radius: 5px;\n"
00063     "}\n"
00064     "\n"
00065     "QLineEdit:focus {\n"
00066     "    border-width: 2px;\n"
00067     "    padding: 0px;\n"
00068     "}\n"
00069     "\n"
00070     "QComboBox::item:hover {\n"
00071     "    background-color: white;\n"
00072     "}\n"
00073     ""));
00074
00075     QFont texteButton;
00076     texteButton.setPointSize(TAILLETEXTEBOUTONTITRE);
00077
00078     fenetreGestionCross = new QStackedWidget(this);
00079     fenetreManifestation = new QWidget(this);
00080     fenetreCourse = new QWidget(this);
00081     fenetreCoureur = new QWidget(this);
00082
00083     fenetreGestionCross->addWidget(fenetreManifestation);
00084     fenetreGestionCross->addWidget(fenetreCourse);
00085     fenetreGestionCross->addWidget(fenetreCoureur);
00086
00087     initialiserFenetreCoureur();
00088
00089     bManifestations = new QPushButton(QString::fromUtf8("Manifestations"), this);
00090     bManifestations->setCheckable(true);
00091     bManifestations->setFont(texteButton);
00092
```

```
00093     bCourses = new QPushButton(QString::fromUtf8("Courses"), this);
00094     bCourses->setCheckable(true);
00095     bCourses->setFont(texteButton);
00096
00097     bCoureurs = new QPushButton(QString::fromUtf8("Coureurs"), this);
00098     bCoureurs->setCheckable(true);
00099     bCoureurs->setFont(texteButton);
00100
00101     logoChronoCross = new QLabel(this);
00102     QPixmap pixmap_img(IMAGECHRONOCROSS);
00103     logoChronoCross->setPixmap(pixmap_img);
00104
00105     QHBoxLayout *hBoutons = new QHBoxLayout;
00106     QHBoxLayout *hLayoutFenetre = new QHBoxLayout;
00107     QVBoxLayout *mainLayout = new QVBoxLayout;
00108
00109     hBoutons->addWidget(bManifestations);
00110     hBoutons->addWidget(bCourses);
00111     hBoutons->addWidget(bCoureurs);
00112     hBoutons->addStretch();
00113     hBoutons->addWidget(logoChronoCross);
00114     hBoutons->setContentsMargins(0, 0, 0, 40); // G H D B
00115     hLayoutFenetre->addWidget(fenetreGestionCross);
00116     mainLayout->addLayout(hBoutons);
00117     mainLayout->addLayout(hLayoutFenetre);
00118
00119     setLayout(mainLayout);
00120     setWindowTitle("Gestion-Cross");
00121     setContextMenuPolicy(Qt::ActionsContextMenu);
00122
00123     QAction *actionQuitter = new QAction("&Quitter", this);
00124     actionQuitter->setShortcut(QKeySequence::Quit);
00125     addAction(actionQuitter);
00126
00127     QAction *actionEntrerPAD = new QAction("&Entrer1", this);
00128     actionEntrerPAD->setShortcut(QKeySequence(Qt::Key_Enter));
00129     addAction(actionEntrerPAD);
00130
00131     QAction *actionEntrerRETURN = new QAction("&Entrer2", this);
00132     actionEntrerRETURN->setShortcut(QKeySequence(Qt::Key_Return));
00133     addAction(actionEntrerRETURN);
00134
00135     //Connect
00136     connect(actionQuitter, SIGNAL(triggered()), this, SLOT(quitter()));
00137     connect(actionEntrerPAD, SIGNAL(triggered()), this, SLOT(
    ajouterNouvelleInscription()));
00138     connect(actionEntrerRETURN, SIGNAL(triggered()), this, SLOT(
    ajouterNouvelleInscription()));
00139
00140     connect(bCoureurs, SIGNAL(clicked()), this, SLOT(gererCoureurs()));
00141     connect(bCourses, SIGNAL(clicked()), this, SLOT(gererCourses()));
00142     connect(bManifestations, SIGNAL(clicked()), this, SLOT(
    gererManifestations()));
00143     connect(bInscrire, SIGNAL(clicked()), this, SLOT(
    ajouterNouvelleInscription()));
00144     connect(bGestionNouveau, SIGNAL(clicked()), this, SLOT(
    passerModeNouveauCoureur()));
00145     connect(bCreationConfirmer, SIGNAL(clicked()), this, SLOT(
    creerCoureur()));
00146     connect(bCreationAnnuler, SIGNAL(clicked()), this, SLOT(
    annulerNouveauCoureur()));
00147     connect(bGestionSupprimer, SIGNAL(clicked()), this, SLOT(
    supprimerCoureur()));
00148     connect(bGestionModifier, SIGNAL(clicked()), this, SLOT(
    modifierCoureur()));
00149
00150     connect(vueTableCoureurs, SIGNAL(clicked(QModelIndex)), this, SLOT(
    selectionnerCoureur(QModelIndex)));
00151     connect(cbInscriptionListeManifestation, SIGNAL(currentIndexChanged(
    QString)), this, SLOT(selectionnerManifestation(QString)));
00152     connect(cbInscriptionListeCourse, SIGNAL(currentIndexChanged(QString)), this,
    SLOT(selectionnerCourse(QString)));
00153     connect(gestionnaireBDD, SIGNAL(nouvelInscrit(QStringList)), this, SLOT(
    mettreAJourTableInscrit(QStringList)));
00154     connect(gestionnaireBDD, SIGNAL(nouveauCoureur(QStringList)), this, SLOT(
    ajouterNouveauCoureurTable(QStringList)));
00155     connect(gestionnaireBDD, SIGNAL(coureurSupprime()), this, SLOT(
    supprimerCoureurTable()));
00156     connect(gestionnaireBDD, SIGNAL(coureurModifie()), this, SLOT(
    modifierCoureurTable()));
00157
00158     setWindowFlags(Qt::Window | Qt::WindowCloseButtonHint);
00159     showMaximized();
00160
00161     gererCoureurs();
00162 }
```

8.7.2.2 ~IHMGestionCross()

```
IHMGestionCross::~IHMGestionCross ( )
```

Références BaseDeDonnees : :detruireInstance().

```
00165 {
00166     BaseDeDonnees::detruireInstance();
00167     qDebug() << Q_FUNC_INFO;
00168 }
```

8.7.3 Documentation des fonctions membres

8.7.3.1 afficherTable()

```
void IHMGestionCross::afficherTable (
            QString nomTable )  [private]
```

Méthode afficherTable() de la classe IHMGestionCross.

Récupère la table Coureur de la base de données et l'affiche dans le tableau

Paramètres

nomTable	QString

Références COLONNE_CATEGORIE, COLONNE_CLASSE, COLONNE_DATENAISSANCE, COLONNE_INE, COLONNE_NOM, COLONNE_PRENOM, COLONNE_SEXE, gestionnaireBDD, INFO_COUREUR_CATEGORIE, INFO_COUREUR_CLASSE, INFO↩_COUREUR_DATENAISSANCE, INFO_COUREUR_INE, INFO_COUREUR_NOM, INFO_COUREUR_PRENOM, INFO_COURE↩UR_SEXE, modeleTableCoureurs, nbLignesTableCoureurs, GestionBDD : :recupererInformation(), GestionBDD : :recupererTableB↩DD(), et tableCoureurs.

Référencé par gererCoureurs(), modifierCoureurTable(), et supprimerCoureurTable().

```
00457 {
00458     // enregistrement [ idCoureur, idCatégorie, idClasse, INE, Nom, Prenom, DateNaissance, Sexe ]
00459     // tableau [ nom prenom classe categorie INE datenaissace sexe ]
00460     tableCoureurs.clear();
00461     tableCoureurs = gestionnaireBDD->
      recupererTableBDD(nomTable);
00462     if(tableCoureurs.size() == 0)
00463         return;
00464     QStringList enregistrement;
00465     int nbEnregistrements = tableCoureurs.count();
00466
00467     qDebug() << Q_FUNC_INFO << nbEnregistrements;
00468
00469     for(int i = 0; i < nbEnregistrements; i += 1)
00470     {
00471         enregistrement = tableCoureurs[i];
00472         QStandardItem *nom = new QStandardItem(enregistrement.at(
      INFO_COUREUR_NOM));
00473         QStandardItem *prenom = new QStandardItem(enregistrement.at(
      INFO_COUREUR_PRENOM));
00474         QStandardItem *classe = new QStandardItem(gestionnaireBDD->
      recupererInformation("Nom", "Classe", QString("idClasse = %1;").arg(enregistrement[
      INFO_COUREUR_CLASSE])));
00475
00476         QStandardItem *categorie = new QStandardItem(gestionnaireBDD->
      recupererInformation("Nom", "Categorie", QString("idCategorie = %1;").arg(
      enregistrement[INFO_COUREUR_CATEGORIE])));
00477         QStandardItem *ine = new QStandardItem(enregistrement.at(
      INFO_COUREUR_INE));
```

```
00478         QStandardItem *dateNaissance = new QStandardItem(enregistrement.at(
      INFO_COUREUR_DATENAISSANCE));
00479         QStandardItem *sexe = new QStandardItem(enregistrement.at(
      INFO_COUREUR_SEXE));
00480
00481         modeleTableCoureurs->setItem(i, COLONNE_NOM, nom);
00482         modeleTableCoureurs->setItem(i, COLONNE_PRENOM, prenom);
00483         modeleTableCoureurs->setItem(i, COLONNE_CLASSE, classe);
00484         modeleTableCoureurs->setItem(i, COLONNE_CATEGORIE, categorie);
00485         modeleTableCoureurs->setItem(i, COLONNE_INE, ine);
00486         modeleTableCoureurs->setItem(i, COLONNE_DATENAISSANCE,
      dateNaissance);
00487         modeleTableCoureurs->setItem(i, COLONNE_SEXE, sexe);
00488
00489         enregistrement.clear();
00490     }
00491     nbLignesTableCoureurs = modeleTableCoureurs->rowCount();
00492 }
```

8.7.3.2 ajouterNouveauCoureurTable

```
void IHMGestionCross::ajouterNouveauCoureurTable (
            QStringList informationCoureur )  [private], [slot]
```

Références COLONNE_CATEGORIE, COLONNE_CLASSE, COLONNE_DATENAISSANCE, COLONNE_INE, COLONNE_NOM, COLONNE_PRENOM, COLONNE_SEXE, messageSucces(), mettreAJourTableCoureur(), modeleTableCoureurs, et nbLignes↩ TableCoureurs.

Référencé par IHMGestionCross().

```
01012 {
01013     qDebug() << Q_FUNC_INFO << informationsCoureur;
01014     // informationsCoureur [ 0 categorie , 1 classe , 2 ine , 3 nom , 4 prenom , 5 datenaissance , 6 sexe ]
01015     if(nbLignesTableCoureurs != modeleTableCoureurs->rowCount())
01016         nbLignesTableCoureurs = modeleTableCoureurs->rowCount();
01017     modeleTableCoureurs->insertRow(nbLignesTableCoureurs);
01018
01019     messageSucces("créé(e)");
01020
01021     QStandardItem *categorie = new QStandardItem(informationsCoureur.at(0));
01022     QStandardItem *classe  = new QStandardItem(informationsCoureur.at(1));
01023     QStandardItem *ine  = new QStandardItem(informationsCoureur.at(2));
01024     QStandardItem *nom  = new QStandardItem(informationsCoureur.at(3));
01025     QStandardItem *prenom  = new QStandardItem(informationsCoureur.at(4));
01026     QStandardItem *dateNaissance  = new QStandardItem(informationsCoureur.at(5));
01027     QStandardItem *sexe  = new QStandardItem(informationsCoureur.at(6));
01028
01029     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_NOM, nom);
01030     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_PRENOM, prenom);
01031     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_CLASSE, classe);
01032     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_CATEGORIE, categorie);
01033     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_INE, ine);
01034     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_DATENAISSANCE, dateNaissance);
01035     modeleTableCoureurs->setItem(nbLignesTableCoureurs,
      COLONNE_SEXE, sexe);
01036
01037     modeleTableCoureurs->sort(4,Qt::SortOrder::AscendingOrder);
01038     mettreAJourTableCoureur();
01039
01040 }
```

8.7.3.3 ajouterNouvelleInscription

```
void IHMGestionCross::ajouterNouvelleInscription ( )  [private], [slot]
```

Références GestionBDD : :ajouterNouvelInscrit(), cbInscriptionListeCourse, gestionnaireBDD, idCoureur, lineEditNom, lineEdit↩ NumeroDossard, listerManifestations(), GestionBDD : :recupererInformation(), et verifierNumeroDossardInscription().

Référencé par IHMGestionCross().

```
00977 {
00978
00979     QString numeroDossard = lineEditNumeroDossard->text();
00980     if(verifierNumeroDossardInscription(numeroDossard))
00981     {
00982         qDebug() << Q_FUNC_INFO << "Dossad valide";
00983         QStringList inscription;
00984         QString nomCourse = cbInscriptionListeCourse->currentText();
00985         QString idCourse = gestionnaireBDD->recupererInformation("
      idCourse", "Course", QString("Nom = '%1'").arg(nomCourse));
00986         QString idCoureur = gestionnaireBDD->
      recupererInformation("idCoureur", "Coureur", QString("Nom = '%1'").arg(
      lineEditNom->text()));
00987         qDebug() << Q_FUNC_INFO << "Inscription :\tidCoureur : " << idCoureur << "\tidCourse : " <<
      idCourse << "\tnumero : " << numeroDossard;
00988         inscription << idCoureur << idCourse << numeroDossard;
00989         gestionnaireBDD->ajouterNouvelInscrit(inscription);
00990         lineEditNumeroDossard->clear();
00991         listerManifestations();
00992     }
00993     else
00994         qDebug() << Q_FUNC_INFO << "Dossard invalide";
00995 }
```

8.7.3.4 annulerNouveauCoureur

```
void IHMGestionCross::annulerNouveauCoureur ( )  [private], [slot]
```

Références reinitialiserGestionCoureur().

Référencé par IHMGestionCross().

```
01128 {
01129     reinitialiserGestionCoureur();
01130 }
```

8.7.3.5 confirmerDialog

```
void IHMGestionCross::confirmerDialog ( )  [private], [slot]
```

Références confirmationDialog, etat, gestionnaireBDD, idCoureur, lineEditINE, GestionBDD : :modifierCoureur(), recupererChamps↩ Gestion(), GestionBDD : :supprimerCoureur(), et verifierInformationsModifierCoureur().

Référencé par initialiserConfirmationDialog().

```
01177 {
01178     if(etat == "Suppression Coureur")
01179     {
01180         QString INE = lineEditINE->text();
01181         qDebug() << Q_FUNC_INFO << "SUPPRESSION DEMANDÉ" << INE;
01182         gestionnaireBDD->supprimerCoureur(INE);
01183     }
01184     else if(etat == "Modification Coureur")
01185     {
01186         QStringList informationsCoureur = recupererChampsGestion();
01187         informationsCoureur << idCoureur;
01188         if(verifierInformationsModifierCoureur(informationsCoureur))
01189         {
01190             informationsCoureur << idCoureur;
01191             qDebug() << Q_FUNC_INFO << "MODIFICATION DEMANDÉ";
01192             gestionnaireBDD->modifierCoureur(informationsCoureur);
01193         }
01194     }
01195     confirmationDialog->close();
01196 }
```

8.7.3.6 creerCoureur

```
void IHMGestionCross::creerCoureur ( )  [private], [slot]
```

Références GestionBDD : :ajouterNouveauCoureur(), gestionnaireBDD, recupererChampsGestion(), reinitialiserGestionCoureur(), et verifierInformationsCreerCoureur().

Référencé par IHMGestionCross().

```
01110 {
01111     QStringList informationsCoureur; // informationsCoureur [ 0 Categorie , 1 classe , 2 INE , 3 nom ,  4
       prenom , 5 dateNaissance , 6 sexe ]
01112     informationsCoureur = recupererChampsGestion();
01113     qDebug() << Q_FUNC_INFO << informationsCoureur;
01114
01115     if(verifierInformationsCreerCoureur(informationsCoureur))
01116     {
01117         qDebug() << Q_FUNC_INFO << "OK";
01118         gestionnaireBDD->ajouterNouveauCoureur(informationsCoureur);
01119         reinitialiserGestionCoureur();
01120     }
01121     else
01122     {
01123         qDebug() << Q_FUNC_INFO << "ERREUR";
01124     }
01125 }
```

8.7.3.7 gererCoureurs

```
void IHMGestionCross::gererCoureurs ( )  [private], [slot]
```

SLOT gererCoureurs() de la classe IHMGestionCross.

Permet de changer l'état de l'IHM pour passer en mode "Gestion Coureur" et de changer l'aspect des boutons

Références afficherTable(), bCoureurs, bCourses, bManifestations, etat, FENETRE_COUREUR, fenetreGestionCross, modeleTable↩Coureurs, nomColonnesCoureur, et vueTableCoureurs.

Référencé par IHMGestionCross().

```
00755 {
00756     etat.clear();
00757     etat = "Mode Coureurs";
00758     if(fenetreGestionCross->currentIndex() == FENETRE_COUREUR)
00759     {
00760         bCoureurs->setChecked(true);
00761         return;
00762     }
00763     qDebug() << Q_FUNC_INFO;
00764     fenetreGestionCross->setCurrentIndex(FENETRE_COUREUR);
00765     bManifestations->setChecked(false);
00766     bCourses->setChecked(false);
00767     bCoureurs->setChecked(true);
00768
00769     if(nomColonnesCoureur.isEmpty())
00770         nomColonnesCoureur << "Nom" << QString::fromUtf8("Prénom") << "Classe" <<
      QString::fromUtf8("Catégorie") << "INE" << "Date de Naissance" << "Sexe";
00771     modeleTableCoureurs->clear();
00772     vueTableCoureurs->setEditTriggers(QAbstractItemView::NoEditTriggers);
00773     vueTableCoureurs->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
00774     vueTableCoureurs->verticalHeader()->setHidden(true);
00775     modeleTableCoureurs->setHorizontalHeaderLabels(
      nomColonnesCoureur);
00776     afficherTable("Coureur");
00777 }
```

8.7.3.8 gererCourses

```
void IHMGestionCross::gererCourses ( )  [private], [slot]
```

SLOT gererCourses() de la classe IHMGestionCross.

Permet de changer l'état de l'IHM pour passer en mode "Gestion Course" et de changer l'aspect des boutons

Références bCoureurs, bCourses, bManifestations, etat, FENETRE_COURSE, et fenetreGestionCross.

Référencé par IHMGestionCross().

```
00785 {
00786     etat.clear();
00787     etat = "Gerer Courses";
00788     if(fenetreGestionCross->currentIndex() == FENETRE_COURSE)
00789     {
00790         bCourses->setChecked(true);
00791         return;
00792     }
00793     qDebug() << Q_FUNC_INFO;
00794     fenetreGestionCross->setCurrentIndex(FENETRE_COURSE);
00795     bManifestations->setChecked(false);
00796     bCourses->setChecked(true);
00797     bCoureurs->setChecked(false);
00798 }
```

8.7.3.9 gererManifestations

```
void IHMGestionCross::gererManifestations ( )  [private], [slot]
```

SLOT gererManifestations() de la classe IHMGestionCross.

Permet de changer l'état de l'IHM pour passer en mode "Gestion Manifestation" et de changer l'aspect des boutons

Références bCoureurs, bCourses, bManifestations, etat, FENETRE_MANIFESTATION, et fenetreGestionCross.

Référencé par IHMGestionCross().

```
00806 {
00807     etat.clear();
00808     etat = "Gerer Manifestations";
00809     if(fenetreGestionCross->currentIndex() ==
      FENETRE_MANIFESTATION)
00810     {
00811         bManifestations->setChecked(true);
00812         return;
00813     }
00814     qDebug() << Q_FUNC_INFO;
00815     fenetreGestionCross->setCurrentIndex(
      FENETRE_MANIFESTATION);
00816     bManifestations->setChecked(true);
00817     bCourses->setChecked(false);
00818     bCoureurs->setChecked(false);
00819 }
```

8.7.3.10 initialiserConfirmationDialog [1/2]

```
void IHMGestionCross::initialiserConfirmationDialog (
            QString nomTache )  [private], [slot]
```

Références bAnnulerDialog, bConfirmationDialog, confirmationDialog, gestionnaireBDD, labelConfirmationDialog, lineEditINE, quitterDialog(), et supprimerCoureur().

```
00876 {
00877     confirmationDialog = new QDialog(this);
00878     labelConfirmationDialog = new QLabel (tr("Etes vous sûr de vouloir %1 ce(cette)
       coureur(coureuse) ?").arg(nomTache));
00879     bConfirmationDialog = new QPushButton (QString::fromUtf8("Confirmer"));
00880     bAnnulerDialog = new QPushButton (QString::fromUtf8("Annuler"));
00881
00882     QHBoxLayout *boutonLayout = new QHBoxLayout;
00883     boutonLayout->addWidget(bConfirmationDialog);
00884     boutonLayout->addWidget(bAnnulerDialog);
00885     boutonLayout->setContentsMargins(0, 0, 0, 5); // G H D B
00886
00887     QVBoxLayout *mainDialogLayout = new QVBoxLayout;
00888     mainDialogLayout->addWidget(labelConfirmationDialog);
00889     mainDialogLayout->addLayout(boutonLayout);
00890     mainDialogLayout->setContentsMargins(10, 10, 10, 10); // G H D B
00891
00892     setWindowTitle(tr("Confirmation"));
00893
00894     connect(bAnnulerDialog, SIGNAL(clicked()), this, SLOT(
      quitterDialog()));
00895     if(nomTache == "supprimer")
00896     {
00897         QString INE = lineEditINE->text();
00898         connect(bConfirmationDialog, SIGNAL(clicked()),
      gestionnaireBDD, SLOT(supprimerCoureur()));
00899     }
00900 /*    else if(nomTache == "modifier")
00901         connect(bConfirmationDialog, SIGNAL(clicked()), this, SLOT();
00902   */
00903     confirmationDialog->setLayout(mainDialogLayout);
00904     confirmationDialog->exec();
00905 }
```

8.7.3.11 initialiserConfirmationDialog [2/2]

```
void IHMGestionCross::initialiserConfirmationDialog ( )  [private], [slot]
```

Références bAnnulerDialog, bConfirmationDialog, confirmationDialog, confirmerDialog(), etat, labelConfirmationDialog, et quitterDialog().

Référencé par modifierCoureur(), et supprimerCoureur().

```
01147 {
01148     confirmationDialog = new QDialog(this);
01149     if(etat == "Suppression Coureur")
01150         labelConfirmationDialog = new QLabel (tr("Etes vous sûr de vouloir supprimer
       le Coureur selectionné ?"));
01151     else if(etat == "Modification Coureur")
01152         labelConfirmationDialog = new QLabel(tr("Etes vous sûr de vouloir modifier
       le Coureur selectionné ?"));
01153     bConfirmationDialog = new QPushButton (QString::fromUtf8("Confirmer"));
01154     bAnnulerDialog = new QPushButton (QString::fromUtf8("Annuler"));
01155
01156     QHBoxLayout *boutonLayout = new QHBoxLayout;
01157     boutonLayout->addWidget(bConfirmationDialog);
01158     boutonLayout->addWidget(bAnnulerDialog);
01159     boutonLayout->setContentsMargins(0, 0, 0, 5); // G H D B
01160
01161     QVBoxLayout *mainDialogLayout = new QVBoxLayout;
01162     mainDialogLayout->addWidget(labelConfirmationDialog);
01163     mainDialogLayout->addLayout(boutonLayout);
01164     mainDialogLayout->setContentsMargins(10, 10, 10, 10); // G H D B
01165
01166     setWindowTitle(tr("Confirmation"));
01167
01168     connect(bAnnulerDialog, SIGNAL(clicked()), this, SLOT(
```

```
    quitterDialog()));
01169     connect(bConfirmationDialog, SIGNAL(clicked()), this, SLOT(
    confirmerDialog()));
01170     qDebug() << Q_FUNC_INFO;
01171
01172     confirmationDialog->setLayout(mainDialogLayout);
01173     confirmationDialog->exec();
01174 }
```

8.7.3.12 initialiserFenetreCoureur()

```
void IHMGestionCross::initialiserFenetreCoureur ( )  [private]
```

Références bCreationAnnuler, bCreationConfirmer, bGestionModifier, bGestionNouveau, bGestionSupprimer, bInscrire, cbGestionCategorie, cbGestionClasse, cbGestionParticipe, cbInscriptionListeCourse, cbInscriptionListeManifestation, deDateNaissance, fenetreCoureur, labelGestion, labelGestionCategorie, labelGestionClasse, labelGestionDateNaissance, labelGestionINE, labelGestionNom, labelGestionParticipe, labelGestionPrenom, labelGestionSexe, labelInscription, labelInscriptionCourse, labelInscriptionManifestation, labelMessageInscription, labelNumeroDossard, lineEditINE, lineEditNom, lineEditNumeroDossard, lineEditPrenom, modeleTableCoureurs, modeleTableInscrits, nomColonnesInscrit, rbGestionSexeF, rbGestionSexeM, TAILLETEXTEBOUTONGESTION, TAILLETEXTEGESTION, TAILLETEXTEINSCRIPTION, TAILLETEXTETITRE, vueTableCoureurs, et vueTableInscrits.

Référencé par IHMGestionCross().

```
00171 {
00172     QFont texteLabelTitre;
00173     texteLabelTitre.setPointSize(TAILLETEXTETITRE);
00174
00175     QFont texteLabelGestion;
00176     texteLabelGestion.setPointSize(TAILLETEXTEGESTION);
00177
00178     QFont texteLabelInscription;
00179     texteLabelInscription.setPointSize(TAILLETEXTEINSCRIPTION);
00180
00181     QFont texteLabelInscriptionMessage;
00182     texteLabelInscriptionMessage.setPointSize(TAILLETEXTEINSCRIPTION);
00183     texteLabelInscriptionMessage.setItalic(true);
00184
00185     QFont texteLabelGestionBouton;
00186     texteLabelGestionBouton.setPointSize(TAILLETEXTEBOUTONGESTION);
00187
00188     //Gestion
00189     labelGestion = new QLabel(QString::fromUtf8("Coureur : "), this);
00190     labelGestion->setFont(texteLabelTitre);
00191
00192     labelGestionNom = new QLabel(QString::fromUtf8("Nom : "), this);
00193     labelGestionNom->setFont(texteLabelGestion);
00194     lineEditNom = new QLineEdit(this);
00195     lineEditNom->setFont(texteLabelGestion);
00196     lineEditNom->setEnabled(false);
00197
00198     labelGestionPrenom = new QLabel(QString::fromUtf8("Prénom : "), this);
00199     labelGestionPrenom->setFont(texteLabelGestion);
00200     lineEditPrenom = new QLineEdit(this);
00201     lineEditPrenom->setFont(texteLabelGestion);
00202     lineEditPrenom->setEnabled(false);
00203
00204     labelGestionDateNaissance = new QLabel(QString::fromUtf8("Date de Naissance :
    "), this);
00205     labelGestionDateNaissance->setFont(texteLabelGestion);
00206     deDateNaissance = new QDateEdit(this);
00207     deDateNaissance->setFont(texteLabelGestion);
00208     deDateNaissance->setEnabled(false);
00209
00210     labelGestionClasse = new QLabel(QString::fromUtf8("Classe : "), this);
00211     labelGestionClasse->setFont(texteLabelGestion);
00212     cbGestionClasse = new QComboBox(this);
00213     cbGestionClasse->setFont(texteLabelGestion);
00214     cbGestionClasse->setStyleSheet("text-align : center");
00215     cbGestionClasse->addItem("< Classes >");
00216     cbGestionClasse->setEnabled(false);
00217
00218     labelGestionCategorie = new QLabel(QString::fromUtf8("Catégorie : "), this);
00219     labelGestionCategorie->setFont(texteLabelGestion);
00220     cbGestionCategorie = new QComboBox(this);
00221     cbGestionCategorie->addItem("< Catégories >");
00222     cbGestionCategorie->setFont(texteLabelGestion);
```

```
00223     cbGestionCategorie->setEnabled(false);
00224
00225     labelGestionINE = new QLabel(QString::fromUtf8("INE : "), this);
00226     labelGestionINE->setFont(texteLabelGestion);
00227     lineEditINE = new QLineEdit(this);
00228     lineEditINE->setFont(texteLabelGestion);
00229     lineEditINE->setEnabled(false);
00230
00231     labelGestionSexe = new QLabel(QString::fromUtf8("Sexe : "), this);
00232     labelGestionSexe->setFont(texteLabelGestion);
00233     rbGestionSexeF = new QRadioButton("F", this);
00234     rbGestionSexeF->setFont(texteLabelGestion);
00235     rbGestionSexeF->setEnabled(false);
00236     rbGestionSexeM = new QRadioButton("M", this);
00237     rbGestionSexeM->setFont(texteLabelGestion);
00238     rbGestionSexeM->setEnabled(false);
00239
00240     labelGestionParticipe = new QLabel(QString::fromUtf8("Participe à :"), this);
00241     labelGestionParticipe->setFont(texteLabelGestion);
00242     cbGestionParticipe = new QComboBox(this);
00243     cbGestionParticipe->setFont(texteLabelGestion);
00244     cbGestionParticipe->setEnabled(false);
00245
00246     //Creation
00247     bCreationConfirmer = new QPushButton(QString::fromUtf8("Confirmer : "), this);
00248     bCreationConfirmer->setFont(texteLabelGestionBouton);
00249     bCreationConfirmer->setVisible(false);
00250     bCreationAnnuler = new QPushButton(QString::fromUtf8("Annuler : "), this);
00251     bCreationAnnuler->setFont(texteLabelGestionBouton);
00252     bCreationAnnuler->setVisible(false);
00253
00254     //Inscription
00255     labelInscription = new QLabel(QString::fromUtf8("Inscription : "), this);
00256     labelInscription->setFont(texteLabelTitre);
00257
00258     labelInscriptionManifestation = new QLabel(QString::fromUtf8("
      Manifestation : "), this);
00259     labelInscriptionManifestation->setFont(texteLabelInscription);
00260     cbInscriptionListeManifestation = new QComboBox(this);
00261     cbInscriptionListeManifestation->addItem("< Liste des manifestations >")
     ;
00262     cbInscriptionListeManifestation->setFont(texteLabelInscription);
00263     cbInscriptionListeManifestation->setEnabled(false);
00264
00265     bInscrire = new QPushButton(QString::fromUtf8("Inscrire"), this);
00266     bInscrire->setEnabled(false);
00267     bInscrire->setFont(texteLabelGestionBouton);
00268
00269     labelInscriptionCourse = new QLabel(QString::fromUtf8("Course : "), this);
00270     labelInscriptionCourse->setFont(texteLabelInscription);
00271     cbInscriptionListeCourse = new QComboBox(this);
00272     cbInscriptionListeCourse->addItem("< Liste des courses >");
00273     cbInscriptionListeCourse->setFont(texteLabelInscription);
00274     cbInscriptionListeCourse->setEnabled(false);
00275
00276     labelNumeroDossard = new QLabel(QString::fromUtf8("Numéro de dossard : "), this);
00277     labelNumeroDossard->setFont(texteLabelGestion);
00278     lineEditNumeroDossard = new QLineEdit(this);
00279     lineEditNumeroDossard->setFont(texteLabelGestion);
00280     lineEditNumeroDossard->setEnabled(false);
00281
00282     labelMessageInscription = new QLabel(this);
00283     labelMessageInscription->setFont(texteLabelInscriptionMessage);
00284
00285     //Tableau
00286     // enregistrement [idCoureur, idCatégorie, idClasse, INE, Nom, Prenom, DateNaissance, Sexe]
00287     // tableau [nom prenom classe categorie INE datenaissace sexe]
00288
00289     vueTableCoureurs = new QTableView(this);
00290     modeleTableCoureurs = new QStandardItemModel();
00291     vueTableCoureurs->setModel(modeleTableCoureurs);
00292
00293     QVBoxLayout *tableauLayout = new QVBoxLayout;
00294     tableauLayout->addWidget(vueTableCoureurs);
00295     tableauLayout->setContentsMargins(0, 0, 20, 0); // G H D B
00296
00297     vueTableInscrits = new QTableView(this);
00298     modeleTableInscrits = new QStandardItemModel();
00299     vueTableInscrits->setModel(modeleTableInscrits);
00300     nomColonnesInscrit << "Nom" << QString::fromUtf8("Prénom") << "Dossard";
00301     modeleTableInscrits->clear();
00302     vueTableInscrits->setEditTriggers(QAbstractItemView::NoEditTriggers);
00303     vueTableInscrits->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
00304     vueTableInscrits->verticalHeader()->setHidden(true);
00305     modeleTableInscrits->setHorizontalHeaderLabels(
     nomColonnesInscrit);
00306
00307     QVBoxLayout *tableauInscritsLayout = new QVBoxLayout;
00308     tableauInscritsLayout->addWidget(vueTableInscrits);
00309
00310     QHBoxLayout *sexeFMGestionLayout = new QHBoxLayout;
```

```
00311     sexeFMGestionLayout->addWidget(rbGestionSexeF);
00312     sexeFMGestionLayout->addWidget(rbGestionSexeM);
00313     sexeFMGestionLayout->addStretch();
00314
00315     QVBoxLayout *editGestionLayout = new QVBoxLayout;
00316     editGestionLayout->addWidget(lineEditNom);
00317     editGestionLayout->addWidget(lineEditPrenom);
00318     editGestionLayout->addWidget(deDateNaissance);
00319     editGestionLayout->addWidget(cbGestionClasse);
00320     editGestionLayout->addWidget(cbGestionCategorie);
00321     editGestionLayout->addWidget(lineEditINE);
00322     editGestionLayout->addLayout(sexeFMGestionLayout);
00323     editGestionLayout->addWidget(cbGestionParticipe);
00324     editGestionLayout->setContentsMargins(15, 0, 0, 0); // G H D B
00325
00326     bGestionNouveau = new QPushButton(QString::fromUtf8("Nouveau"), this);
00327     bGestionNouveau->setEnabled(true);
00328     bGestionNouveau->setFont(texteLabelGestionBouton);
00329     bGestionModifier = new QPushButton(QString::fromUtf8("Modifier"), this);
00330     bGestionModifier->setEnabled(false);
00331     bGestionModifier->setFont(texteLabelGestionBouton);
00332     bGestionSupprimer = new QPushButton(QString::fromUtf8("Supprimer"), this);
00333     bGestionSupprimer->setEnabled(false);
00334     bGestionSupprimer->setFont(texteLabelGestionBouton);
00335
00336     QHBoxLayout *gestionLayoutBoutons = new QHBoxLayout;
00337     gestionLayoutBoutons->addWidget(bGestionNouveau);
00338     gestionLayoutBoutons->addWidget(bGestionModifier);
00339     gestionLayoutBoutons->addWidget(bGestionSupprimer);
00340     gestionLayoutBoutons->addStretch(10);
00341     gestionLayoutBoutons->addWidget(bCreationConfirmer);
00342     gestionLayoutBoutons->addWidget(bCreationAnnuler);
00343     gestionLayoutBoutons->addStretch();
00344
00345     QVBoxLayout *labelGestionLayout = new QVBoxLayout;
00346     labelGestionLayout->addWidget(labelGestionNom);
00347     labelGestionLayout->addWidget(labelGestionPrenom);
00348     labelGestionLayout->addWidget(labelGestionDateNaissance);
00349     labelGestionLayout->addWidget(labelGestionClasse);
00350     labelGestionLayout->addWidget(labelGestionCategorie);
00351     labelGestionLayout->addWidget(labelGestionINE);
00352     labelGestionLayout->addWidget(labelGestionSexe);
00353     labelGestionLayout->addWidget(labelGestionParticipe);
00354     labelGestionLayout->setContentsMargins(10, 0, 15, 0); // G H D B
00355
00356     QHBoxLayout *gestionLayout = new QHBoxLayout;
00357     gestionLayout->addLayout(labelGestionLayout);
00358     gestionLayout->addLayout(editGestionLayout);
00359     gestionLayout->setContentsMargins(0, 0, 0, 10);
00360
00361     QHBoxLayout *inscriptionDossardLayout = new QHBoxLayout;
00362     inscriptionDossardLayout->addWidget(labelNumeroDossard);
00363     inscriptionDossardLayout->addWidget(lineEditNumeroDossard);
00364     inscriptionDossardLayout->addStretch();
00365
00366     QHBoxLayout *gestionLayoutBouton = new QHBoxLayout;
00367     gestionLayoutBouton->addWidget(bInscrire);
00368     gestionLayoutBouton->addStretch();
00369
00370     QHBoxLayout *inscriptionManifestationLayout = new QHBoxLayout;
00371     inscriptionManifestationLayout->addWidget(labelInscriptionManifestation);
00372     inscriptionManifestationLayout->addWidget(cbInscriptionListeManifestation
      );
00373     inscriptionManifestationLayout->addStretch();
00374
00375     QHBoxLayout *inscriptionCourseLayout = new QHBoxLayout;
00376     inscriptionCourseLayout->addWidget(labelInscriptionCourse);
00377     inscriptionCourseLayout->addWidget(cbInscriptionListeCourse);
00378     inscriptionCourseLayout->addStretch();
00379
00380     QVBoxLayout *inscriptionLayout = new QVBoxLayout;
00381     inscriptionLayout->addWidget(labelInscription);
00382     inscriptionLayout->addLayout(inscriptionManifestationLayout);
00383     inscriptionLayout->addLayout(inscriptionCourseLayout);
00384     inscriptionLayout->addLayout(inscriptionDossardLayout);
00385     inscriptionLayout->addLayout(gestionLayoutBouton);
00386     inscriptionLayout->addLayout(tableauInscritsLayout);
00387     inscriptionLayout->addWidget(labelMessageInscription);
00388
00389     QVBoxLayout *gestionInscriptionLayout = new QVBoxLayout;
00390     gestionInscriptionLayout->addWidget(labelGestion);
00391     gestionInscriptionLayout->addLayout(gestionLayout);
00392     gestionInscriptionLayout->addLayout(gestionLayoutBoutons);
00393     gestionInscriptionLayout->addSpacing(20);
00394     gestionInscriptionLayout->addLayout(inscriptionLayout);
00395     gestionInscriptionLayout->setContentsMargins(20, 0, 0, 0); // G H D B
00396
00397     QHBoxLayout *panneauLayout = new QHBoxLayout;
00398     panneauLayout->addLayout(tableauLayout);
00399     panneauLayout->addLayout(gestionInscriptionLayout);
00400     panneauLayout->setContentsMargins(5, 0, 5, 5); // G H D B
```

```
00401     fenetreCoureur->setLayout(panneauLayout);
00402 }
```

8.7.3.13 listerCourses()

```
void IHMGestionCross::listerCourses (
            QString idManifestation )  [private]
```

Références cbInscriptionListeCourse, gestionnaireBDD, listeCourses, listeManifestations, GestionBDD : :recupererListeCourses↩ Inscription(), et recupererSexeCoureur().

Référencé par selectionnerManifestation().

```
00431 {
00432     if(listeManifestations[0] != "Inscrit(e) à toute les courses disponibles")
00433     {
00434         QString sexe = recupererSexeCoureur();
00435         listeCourses = gestionnaireBDD->
      recupererListeCoursesInscription(nom, sexe);
00436         qDebug() << Q_FUNC_INFO << listeCourses.size() << nom <<
      listeCourses;
00437         cbInscriptionListeCourse->clear();
00438         cbInscriptionListeCourse->addItem("< Liste des courses >");
00439         for(int i=0; i < listeCourses.size(); i++)
00440         {
00441             qDebug() << Q_FUNC_INFO << listeCourses.at(i);
00442             cbInscriptionListeCourse->addItem(listeCourses.at(i));
00443         }
00444         cbInscriptionListeCourse->setEnabled(true);
00445     }
00446     else
00447         qDebug() << Q_FUNC_INFO << "Aucune course disponible";
00448 }
```

8.7.3.14 listerManifestations()

```
void IHMGestionCross::listerManifestations ( )  [private]
```

Références bInscrire, cbInscriptionListeCourse, cbInscriptionListeManifestation, gestionnaireBDD, lineEditINE, listeManifestations, et GestionBDD : :recupererListeManifestationsInscription().

Référencé par ajouterNouvelleInscription(), et selectionnerCoureur().

```
00405 {
00406     QString INE = lineEditINE->text();
00407     listeManifestations = gestionnaireBDD->
      recupererListeManifestationsInscription(INE);
00408     cbInscriptionListeManifestation->clear();
00409
00410     if(listeManifestations[0] == "Inscrit(e) à toute les courses disponibles")
00411     {
00412         cbInscriptionListeManifestation->addItem("Inscrit(e) à toute les
       courses disponibles");
00413         cbInscriptionListeManifestation->setEditable(false);
00414         cbInscriptionListeCourse->addItem("Inscrit(e) à toute les courses
       disponibles");
00415         cbInscriptionListeCourse->setEnabled(true);
00416         cbInscriptionListeCourse->setEditable(false);
00417         bInscrire->setEnabled(false);
00418     }
00419     else
00420     {
00421         cbInscriptionListeManifestation->clear();
00422         cbInscriptionListeManifestation->addItem("< Liste des manifestations
       >");
00423         for(int i=0; i < listeManifestations.size(); i++)
00424         {
00425             cbInscriptionListeManifestation->addItem(
      listeManifestations.at(i));
00426         }
00427     }
00428 }
```

8.7.3.15 messageErreur()

```
void IHMGestionCross::messageErreur (
            QString message )  [private]
```

Références labelMessageInscription, et lineEditNumeroDossard.

Référencé par verifierInformationsCreerCoureur(), verifierInformationsModifierCoureur(), et verifierNumeroDossardInscription().

```
00675 {
00676     labelMessageInscription->setText(message);
00677     labelMessageInscription->setStyleSheet("color : #FF0000");
00678     lineEditNumeroDossard->setStyleSheet("color : #FF0000");
00679 }
```

8.7.3.16 messageSucces()

```
void IHMGestionCross::messageSucces (
            QString message )  [private]
```

Références labelMessageInscription.

Référencé par ajouterNouveauCoureurTable(), modifierCoureurTable(), et supprimerCoureurTable().

```
00682 {
00683     labelMessageInscription->setStyleSheet("color : #000000");
00684     labelMessageInscription->setText(QString("Le(La) coureur(euse) a été %1 avec
      succés !").arg(message));
00685 }
```

8.7.3.17 mettreAJourTableCoureur

```
void IHMGestionCross::mettreAJourTableCoureur ( )  [private], [slot]
```

Références gestionnaireBDD, GestionBDD : :recupererTableBDD(), et tableCoureurs.

Référencé par ajouterNouveauCoureurTable(), modifierCoureurTable(), et supprimerCoureurTable().

```
01043 {
01044     tableCoureurs.clear();
01045     tableCoureurs = gestionnaireBDD->
     recupererTableBDD("Coureur");
01046     qDebug() << Q_FUNC_INFO;
01047 }
```

8.7.3.18 mettreAJourTableInscrit

```
void IHMGestionCross::mettreAJourTableInscrit (
            QStringList inscription )  [private], [slot]
```

Références COLONNE_NOM, COLONNE_NUMERODOSSARD, COLONNE_PRENOM, INFO_COUREURINSCRIT_NOM, INFO_COUREURINSCRIT_NUMERODOSSAD, INFO_COUREURINSCRIT_PRENOM, modeleTableInscrits, et nbLignesTableInscrit.

Référencé par IHMGestionCross().

```
00998 {
00999     QStandardItem *nom = new QStandardItem(inscription.at(
      INFO_COUREURINSCRIT_NOM));
01000     QStandardItem *prenom = new QStandardItem(inscription.at(
      INFO_COUREURINSCRIT_PRENOM));
01001     QStandardItem *numeroDossard = new QStandardItem(inscription.at(
      INFO_COUREURINSCRIT_NUMERODOSSAD));
01002
01003     nbLignesTableInscrit = modeleTableInscrits->rowCount();
01004     qDebug() << Q_FUNC_INFO << nbLignesTableInscrit;
01005
01006     modeleTableInscrits->setItem(nbLignesTableInscrit,
      COLONNE_NOM, nom);
01007     modeleTableInscrits->setItem(nbLignesTableInscrit,
      COLONNE_PRENOM, prenom);
01008     modeleTableInscrits->setItem(nbLignesTableInscrit,
      COLONNE_NUMERODOSSARD, numeroDossard);
01009 }
```

8.7.3.19 modifierCoureur

```
void IHMGestionCross::modifierCoureur ( )  [private], [slot]
```

Références etat, et initialiserConfirmationDialog().

Référencé par IHMGestionCross().

```
01140 {
01141     etat.clear();
01142     etat = "Modification Coureur";
01143     initialiserConfirmationDialog();
01144 }
```

8.7.3.20 modifierCoureurTable

```
void IHMGestionCross::modifierCoureurTable ( )  [private], [slot]
```

Références afficherTable(), messageSucces(), et mettreAJourTableCoureur().

Référencé par IHMGestionCross().

```
01058 {
01059     messageSucces("modifié(e)");
01060     afficherTable("Coureur");
01061     mettreAJourTableCoureur();
01062 }
```

8.7.3.21 passerModeNouveauCoureur

```
void IHMGestionCross::passerModeNouveauCoureur ( )  [private], [slot]
```

Références bCreationAnnuler, bCreationConfirmer, bGestionModifier, bGestionNouveau, bGestionSupprimer, cbGestionCategorie, cbGestionClasse, cbGestionParticipe, cbInscriptionListeCourse, cbInscriptionListeManifestation, dateDefault, deDateNaissance, etat, gestionnaireBDD, lineEditINE, lineEditNom, lineEditPrenom, rbGestionSexeF, rbGestionSexeM, GestionBDD : :recuperer↩ CategoriesCreation(), GestionBDD : :recupererClassesCreation(), et vueTableCoureurs.

Référencé par IHMGestionCross().

```
01065 {
01066     etat.clear();
01067     etat = "Nouveau Coureur";
01068     vueTableCoureurs->setEnabled(false);
01069     bGestionModifier->setEnabled(false);
01070     bGestionNouveau->setEnabled(false);
01071     cbInscriptionListeCourse->setEnabled(false);
01072     cbInscriptionListeManifestation->setEnabled(false);
01073     bGestionSupprimer->setEnabled(false);
01074     bCreationConfirmer->setVisible(true);
01075     bCreationAnnuler->setVisible(true);
01076     rbGestionSexeF->setEnabled(true);
01077     rbGestionSexeM->setEnabled(true);
01078     cbGestionParticipe->clear();
01079     lineEditNom->setEnabled(true);
01080     lineEditNom->clear();
01081     lineEditPrenom->setEnabled(true);
01082     lineEditPrenom->clear();
01083     lineEditINE->setEnabled(true);
01084     lineEditINE->clear();
01085     deDateNaissance->setEnabled(true);
01086     deDateNaissance->setDate(dateDefault);
01087     cbGestionClasse->setEnabled(true);
01088     cbGestionClasse->clear();
01089     cbGestionClasse->addItem("< Classes >");
01090     cbGestionCategorie->setEnabled(true);
01091     cbGestionCategorie->clear();
01092     cbGestionCategorie->addItem("< Catégories >");
01093     QVector<QString> categories;
01094     categories = gestionnaireBDD->recupererCategoriesCreation();
01095     QVector<QString> classes;
01096     classes = gestionnaireBDD->recupererClassesCreation();
01097
01098     for(int i = 0; i < classes.size(); i += 1)
01099     {
01100         cbGestionClasse->addItem(classes[i]);
01101     }
01102
01103     for(int i = 0; i < categories.size(); i += 1)
01104     {
01105         cbGestionCategorie->addItem(categories[i]);
01106     }
01107 }
```

8.7.3.22 quitter

```
void IHMGestionCross::quitter ( )  [private], [slot]
```

SLOT quitter() de la classe IHMGestionCross.

Action qui ferme la page

Référencé par IHMGestionCross().

```
01216 {
01217     close();
01218 }
```

8.7.3.23 quitterDialog

```
void IHMGestionCross::quitterDialog ( )  [private], [slot]
```

Références confirmationDialog, et reinitialiserGestionCoureur().

Référencé par initialiserConfirmationDialog().

```
01204 {
01205     confirmationDialog->close();
01206     reinitialiserGestionCoureur();
01207     qDebug() << Q_FUNC_INFO;
01208 }
```

8.7.3.24 recupererChampsGestion()

```
QStringList IHMGestionCross::recupererChampsGestion ( )  [private]
```

Références cbGestionCategorie, cbGestionClasse, deDateNaissance, lineEditINE, lineEditNom, lineEditPrenom, rbGestionSexeF, et rbGestionSexeM.

Référencé par confirmerDialog(), et creerCoureur().

```
00698 {
00699     QStringList informationsCoureur;
00700     informationsCoureur << cbGestionCategorie->currentText();
00701     informationsCoureur << cbGestionClasse->currentText();
00702     informationsCoureur << lineEditINE->text();
00703     informationsCoureur << lineEditNom->text();
00704     informationsCoureur << lineEditPrenom->text();
00705     QDate date = deDateNaissance->date();
00706     QString dateNaissance = date.toString("yyyy-MM-d");
00707     informationsCoureur << dateNaissance;
00708     if(rbGestionSexeF->isChecked())
00709         informationsCoureur << "F";
00710     if(rbGestionSexeM->isChecked())
00711         informationsCoureur << "M";
00712     informationsCoureur[3] = informationsCoureur[3].toUpper();
00713     informationsCoureur[4] = informationsCoureur[4].toLower();
00714     informationsCoureur[4][0] = informationsCoureur[4][0].toUpper();
00715     return informationsCoureur;
00716 }
```

8.7.3.25 recupererSexeCoureur()

```
QString IHMGestionCross::recupererSexeCoureur ( )  [private]
```

Références rbGestionSexeF, et rbGestionSexeM.

Référencé par listerCourses().

```
00501 {
00502     if(rbGestionSexeF->isChecked())
00503         return "F";
00504     else if(rbGestionSexeM->isChecked())
00505         return "M";
00506     else
00507         return "ERREUR";
00508 }
```

8.7.3.26 reinitialiserGestionCoureur()

```
void IHMGestionCross::reinitialiserGestionCoureur ( )  [private]
```

Références bCreationAnnuler, bCreationConfirmer, bGestionNouveau, cbGestionCategorie, cbGestionClasse, cbGestionParticipe, dateDefault, deDateNaissance, etat, lineEditINE, lineEditNom, lineEditPrenom, rbGestionSexeF, rbGestionSexeM, et vueTableCoureurs.

Référencé par annulerNouveauCoureur(), creerCoureur(), quitterDialog(), et supprimerCoureurTable().

```
00719 {
00720     etat.clear();
00721     etat = "Gestion Coureurs";
00722     vueTableCoureurs->setEnabled(true);
00723     bGestionNouveau->setEnabled(true);
00724     bCreationConfirmer->setVisible(false);
00725     bCreationAnnuler->setVisible(false);
00726     lineEditNom->clear();
00727     lineEditNom->setEnabled(false);
00728     lineEditPrenom->clear();
00729     lineEditPrenom->setEnabled(false);
00730     deDateNaissance->setDate(dateDefault);
00731     deDateNaissance->setEnabled(false);
00732     cbGestionClasse->clear();
00733     cbGestionClasse->addItem("< Classes >");
00734     cbGestionClasse->setEnabled(false);
00735     cbGestionCategorie->clear();
00736     cbGestionCategorie->addItem("< Categories >");
00737     cbGestionCategorie->setEnabled(false);
00738     cbGestionParticipe->clear();
00739     cbGestionParticipe->addItem("< Categories >");
00740     cbGestionParticipe->setEnabled(false);
00741     lineEditINE->clear();
00742     lineEditINE->setEnabled(false);
00743     rbGestionSexeF->setEnabled(false);
00744     rbGestionSexeM->setEnabled(false);
00745 }
```

8.7.3.27 selectionnerCoureur

```
void IHMGestionCross::selectionnerCoureur (
            QModelIndex index )  [private], [slot]
```

Références bGestionModifier, bGestionSupprimer, cbGestionCategorie, cbGestionClasse, cbGestionParticipe, cbInscriptionListeCourse, cbInscriptionListeManifestation, deDateNaissance, etat, gestionnaireBDD, idCoureur, INFO_COUREUR_CATEGORIE, INFO_COUREUR_CLASSE, INFO_COUREUR_DATENAISSANCE, INFO_COUREUR_INE, INFO_COUREUR_NOM, INFO_COUREUR_PRENOM, INFO_COUREUR_SEXE, labelMessageInscription, lineEditINE, lineEditNom, lineEditNumeroDossard, lineEditPrenom, listerManifestations(), modeleTableInscrits, rbGestionSexeF, rbGestionSexeM, GestionBDD : :recupererInformation(), GestionBDD : :recupererListeCoursesGestion(), tableCoureurs, et traiterDateNaissance().

Référencé par IHMGestionCross().

```
00908 {
00909     etat.clear();
00910     etat = "Gestion Coureurs";
00911
00912     qDebug() << Q_FUNC_INFO << tableCoureurs;
00913
00914     if(cbInscriptionListeCourse->currentText() != "< Liste des courses >")
00915     {
00916         modeleTableInscrits->clear();
00917         cbInscriptionListeCourse->clear();
00918         cbInscriptionListeCourse->addItem("< Liste des courses >");
00919         lineEditNumeroDossard->clear();
00920         lineEditNumeroDossard->setEnabled(false);
00921     }
00922     labelMessageInscription->clear();
00923     bGestionModifier->setEnabled(true);
00924     bGestionSupprimer->setEnabled(true);
00925     lineEditNom->setText(tableCoureurs.at(index.row()).at(
      INFO_COUREUR_NOM));
```

```
00926     lineEditNom->setEnabled(true);
00927     lineEditPrenom->setText(tableCoureurs.at(index.row()).at(
      INFO_COUREUR_PRENOM));
00928     lineEditPrenom->setEnabled(true);
00929     lineEditINE->setText(tableCoureurs.at(index.row()).at(
      INFO_COUREUR_INE));
00930     lineEditINE->setEnabled(true);
00931     deDateNaissance->setEnabled(true);
00932
00933     idCoureur = gestionnaireBDD->recupererInformation("
      idCoureur", "Coureur", QString("INE = '%1';").arg(tableCoureurs.at(index.row()).at(
      INFO_COUREUR_INE)));
00934
00935     qDebug() << Q_FUNC_INFO << "id : " << idCoureur;
00936
00937     QString dateNaissance = tableCoureurs.at(index.row()).at(
      INFO_COUREUR_DATENAISSANCE);
00938     QVector<int> dates = traiterDateNaissance(dateNaissance);
00939     QDate date(dates[0], dates[1], dates[2]);
00940     qDebug() << Q_FUNC_INFO << date;
00941     deDateNaissance->setDate(date);
00942
00943     QString categorie = gestionnaireBDD->recupererInformation("Nom", "
      Categorie", QString("idCategorie = %1;").arg(tableCoureurs.at(index.row()).at(
      INFO_COUREUR_CATEGORIE)));
00944     cbGestionCategorie->clear();
00945     cbGestionCategorie->addItem(categorie);
00946     cbGestionCategorie->setEnabled(true);
00947
00948     QString classe = gestionnaireBDD->recupererInformation("Nom", "
      Classe", QString("idClasse = %1;").arg(tableCoureurs.at(index.row()).at(
      INFO_COUREUR_CLASSE)));
00949     cbGestionClasse->setEnabled(true);
00950     cbGestionClasse->clear();
00951     cbGestionClasse->addItem(classe);
00952
00953     rbGestionSexeF->setEnabled(true);
00954     rbGestionSexeM->setEnabled(true);
00955     cbGestionParticipe->setEnabled(true);
00956     cbGestionParticipe->setEditable(false);
00957     cbGestionParticipe->clear();
00958     cbInscriptionListeManifestation->setEnabled(true);
00959
00960     if(tableCoureurs.at(index.row()).at(INFO_COUREUR_SEXE) == "F")
00961         rbGestionSexeF->setChecked(true);
00962     else
00963         rbGestionSexeM->setChecked(true);
00964
00965     QString INE = lineEditINE->text();
00966     QStringList nomCoursesInscrit = gestionnaireBDD->
      recupererListeCoursesGestion(INE);
00967
00968     for(int i = 0; i < nomCoursesInscrit.size(); i += 1)
00969     {
00970         cbGestionParticipe->addItem(nomCoursesInscrit[i]);
00971     }
00972
00973     listerManifestations();
00974 }
```

8.7.3.28 selectionnerCourse

```
void IHMGestionCross::selectionnerCourse (
            QString nom )  [private], [slot]
```

Références bInscrire, cbInscriptionListeCourse, COLONNE_NOM, COLONNE_NUMERODOSSARD, COLONNE_PRENOM, gestionnaireBDD, INFO_COUREURINSCRIT_NOM, INFO_COUREURINSCRIT_NUMERODOSSAD, INFO_COUREURINSCRIT_PRENOM, labelMessageInscription, lineEditNumeroDossard, listeCoureursInscrit, listeManifestations, modeleTableInscrits, GestionBDD : :recupererInformation(), GestionBDD : :recupererListeCoureursInscrit(), et viderTableInscrit().

Référencé par IHMGestionCross().

```
00836 {
00837     if(nom!="< Liste des courses >" && nom!="")
00838     {
00839         if(listeManifestations[0] != "Inscrit(e) à toute les courses disponibles")
00840         {
00841             if(!listeCoureursInscrit.isEmpty())
```

```
00842                 viderTableInscrit();
00843             lineEditNumeroDossard->setEnabled(true);
00844             listeCoureursInscrit = gestionnaireBDD->
      recupererListeCoureursInscrit(nom);
00845             int nbCoureursInscrits = listeCoureursInscrit.size();
00846             for(int i = 0; i < nbCoureursInscrits; i += 1)
00847             {
00848                 QStringList enregistrement = listeCoureursInscrit[i];
00849                 QStandardItem *nom = new QStandardItem(enregistrement.at(
      INFO_COUREURINSCRIT_NOM));
00850                 QStandardItem *prenom = new QStandardItem(enregistrement.at(
      INFO_COUREURINSCRIT_PRENOM));
00851                 QStandardItem *numeroDossard = new QStandardItem(enregistrement.at(
      INFO_COUREURINSCRIT_NUMERODOSSAD));
00852
00853                 modeleTableInscrits->setItem(i, COLONNE_NOM, nom);
00854                 modeleTableInscrits->setItem(i,
      COLONNE_PRENOM, prenom);
00855                 modeleTableInscrits->setItem(i,
      COLONNE_NUMERODOSSARD, numeroDossard);
00856
00857                 QString idCourse = gestionnaireBDD->
      recupererInformation("idCourse", "Course", QString("Nom = '%1'").arg(
      cbInscriptionListeCourse->currentText()));
00858
00859                 labelMessageInscription->setText(QString("Pour inscrire un(e)
       coureur(euse)\nVeuillez entrer un numéro de dossard disponible entre %1001 et %1099").arg(idCourse));
00860             }
00861             bInscrire->setEnabled(true);
00862             QString idCourse = gestionnaireBDD->
      recupererInformation("idCourse", "Course", QString("Nom = '%1'").arg(
      cbInscriptionListeCourse->currentText()));
00863
00864             qDebug() << Q_FUNC_INFO << idCourse;
00865
00866             lineEditNumeroDossard->setText(idCourse);
00867         }
00868         else
00869         {
00870             lineEditNumeroDossard->setEnabled(false);
00871         }
00872     }
00873 }
```

8.7.3.29 selectionnerManifestation

```
void IHMGestionCross::selectionnerManifestation (
            QString nom )  [private], [slot]
```

Références cbInscriptionListeCourse, etat, listeCourses, et listerCourses().

Référencé par IHMGestionCross().

```
00822 {
00823     etat.clear();
00824     etat = "Inscription Coureur";
00825     if(nom != "< Liste des manifestations >" && nom != "")
00826     {
00827         listeCourses.clear();
00828         cbInscriptionListeCourse->clear();
00829         listerCourses(nom);
00830         return;
00831     }
00832     return;
00833 }
```

8.7.3.30 supprimerCoureur

```
void IHMGestionCross::supprimerCoureur ( )  [private], [slot]
```

Références etat, et initialiserConfirmationDialog().

Référencé par IHMGestionCross(), et initialiserConfirmationDialog().

```
01133 {
01134     etat.clear();
01135     etat = "Suppression Coureur";
01136     initialiserConfirmationDialog();
01137 }
```

8.7.3.31 supprimerCoureurTable

```
void IHMGestionCross::supprimerCoureurTable ( )  [private], [slot]
```

Références afficherTable(), messageSucces(), mettreAJourTableCoureur(), et reinitialiserGestionCoureur().

Référencé par IHMGestionCross().

```
01050 {
01051     messageSucces("supprimé(e)");
01052     reinitialiserGestionCoureur();
01053     afficherTable("Coureur");
01054     mettreAJourTableCoureur();
01055 }
```

8.7.3.32 traiterDateNaissance()

```
QVector< int > IHMGestionCross::traiterDateNaissance (
            QString dateNaissance )  [private]
```

Référencé par selectionnerCoureur().

```
00688 {
00689     //date : YYYY-MM-DD
00690     QStringList listDateNaissance = date.split("-", QString::SkipEmptyParts);
00691     QVector<int> retourDateNaissanceINT;
00692     retourDateNaissanceINT << listDateNaissance[0].toInt() << listDateNaissance[1].toInt() <<
      listDateNaissance[2].toInt();
00693     qDebug() << Q_FUNC_INFO << retourDateNaissanceINT;
00694     return retourDateNaissanceINT;
00695 }
```

8.7.3.33 verifierInformationsCreerCoureur()

```
bool IHMGestionCross::verifierInformationsCreerCoureur (
            QStringList informations )  [private]
```

Références cbGestionCategorie, cbGestionClasse, gestionnaireBDD, labelMessageInscription, messageErreur(), et GestionBDD↩ : :verifierInformation().

Référencé par creerCoureur().

```
00554 {
00555     // informationsCoureur [ 0 Categorie , 1 classe , 2 INE , 3 nom , 4 prenom , 5 dateNaissance , 6 sexe ]
00556     if(informations.size() == 7)
00557     {
00558         labelMessageInscription->clear();
00559         for(int i = 0;  i < 7; i += 1)
00560         {
00561             if(informations[i].isEmpty())
00562             {
00563                 qDebug() << Q_FUNC_INFO << "CHAMPS VIDE";
00564                 messageErreur("Creation impossible:\nUn ou plusieurs champs sont vides.");
00565                 return false;
00566             }
00567         }
00568         bool verificationINE = gestionnaireBDD->
      verifierInformation(QString("INE = '%1'").arg(informations[2]), "Coureur");
00569         qDebug() << Q_FUNC_INFO << "idCoureur : " << verificationINE;
00570         if(verificationINE)
00571         {
00572             qDebug() << Q_FUNC_INFO << "INE INVALIDE";
00573             messageErreur("Creation impossible:\nINE invalide.");
00574             return false;
```

```
00575         }
00576         qDebug() << Q_FUNC_INFO << "INE VALIDE";
00577         bool doublon = gestionnaireBDD->verifierInformation(QString("Nom
       = '%1' AND Prenom = '%2' AND DateNaissance = '%3';").arg(informations[3]).arg(informations[4]).arg(
      informations[5]), "Coureur");
00578         if(doublon)
00579         {
00580             qDebug() << Q_FUNC_INFO << "INFORMATION INVALIDE";
00581             messageErreur("Creation impossible :\nNom prenom ou DateNaissance invalide.");
00582             return false;
00583         }
00584         qDebug() << Q_FUNC_INFO << "INFORMATION VALIDE";
00585         if(cbGestionClasse->currentText() == "< Classes >")
00586         {
00587             qDebug() << Q_FUNC_INFO << "CLASSE INVALIDE";
00588             messageErreur("Creation impossible :\nClasse invalide.");
00589             return false;
00590         }
00591         if(cbGestionCategorie->currentText() == "< Catégories >")
00592         {
00593             qDebug() << Q_FUNC_INFO << "CATEGORIE INVALIDE";
00594             messageErreur("Création impossible:\nCategorie invalide.");
00595             return false;
00596         }
00597         else
00598         {
00599             labelMessageInscription->setStyleSheet("color : #000000");
00600             qDebug() << Q_FUNC_INFO << "CHAMPS OK";
00601             labelMessageInscription->setText("Création en cours...");
00602             return true;
00603         }
00604     }
00605     else
00606     {
00607         messageErreur("Creation impossible :\nVeuillez remplir le formulaire.");
00608         return false;
00609     }
00610 }
```

8.7.3.34 verifierInformationsModifierCoureur()

```
bool IHMGestionCross::verifierInformationsModifierCoureur (
            QStringList informations )  [private]
```

Références cbGestionCategorie, cbGestionClasse, gestionnaireBDD, labelMessageInscription, messageErreur(), et GestionBDD↩ : :verifierInformation().

Référencé par confirmerDialog().

```
00613 {
00614     // informationsCoureur [ 0 Categorie , 1 classe , 2 INE , 3 nom , 4 prenom , 5 dateNaissance , 6 sexe ,
       7 idCoureur]
00615     qDebug() << Q_FUNC_INFO << informations << informations.size();
00616     if(informations.size() == 8)
00617     {
00618         labelMessageInscription->clear();
00619         for(int i = 0;  i < 7; i += 1)
00620         {
00621             if(informations[i].isEmpty())
00622             {
00623                 qDebug() << Q_FUNC_INFO << "CHAMPS VIDE";
00624                 messageErreur("Creation impossible:\nUn ou plusieurs champs sont vides.");
00625                 return false;
00626             }
00627         }
00628         bool verificationINE = gestionnaireBDD->
      verifierInformation(QString("INE = '%1' AND idCoureur != '%2';").arg(informations[2]).
      arg(informations[7]), "Coureur");
00629         qDebug() << Q_FUNC_INFO << "idCoureur : " << verificationINE;
00630         if(verificationINE)
00631         {
00632             qDebug() << Q_FUNC_INFO << "INE INVALIDE";
00633             messageErreur("Creation impossible:\nINE invalide.");
00634             return false;
00635         }
00636         qDebug() << Q_FUNC_INFO << "INE VALIDE";
00637         bool doublon = gestionnaireBDD->verifierInformation(QString("Nom
       = '%1' AND Prenom = '%2' AND DateNaissance = '%3' AND idCoureur != '%4';").arg(informations[3]).arg(
      informations[4]).arg(informations[5]).arg(informations[7]), "Coureur");
```

```
00638         if(doublon)
00639         {
00640             qDebug() << Q_FUNC_INFO << "INFORMATION INVALIDE";
00641             messageErreur("Creation impossible :\nNom prenom ou DateNaissance invalide.");
00642             return false;
00643         }
00644         qDebug() << Q_FUNC_INFO << "INFORMATION VALIDE";
00645         if(cbGestionClasse->currentText() == "< Classes >")
00646         {
00647             qDebug() << Q_FUNC_INFO << "CLASSE INVALIDE";
00648             messageErreur("Creation impossible :\nClasse invalide.");
00649             return false;
00650         }
00651         qDebug() << Q_FUNC_INFO << "CLASSE VALIDE";
00652         if(cbGestionCategorie->currentText() == "< Catégories >")
00653         {
00654             qDebug() << Q_FUNC_INFO << "CATEGORIE INVALIDE";
00655             messageErreur("Création impossible:\nCategorie invalide.");
00656             return false;
00657         }
00658         else
00659         {
00660             qDebug() << Q_FUNC_INFO << "CATEGORIE VALIDE";
00661             labelMessageInscription->setStyleSheet("color : #000000");
00662             qDebug() << Q_FUNC_INFO << "CHAMPS OK";
00663             labelMessageInscription->setText("Modification en cours...");
00664             return true;
00665         }
00666     }
00667     else
00668     {
00669         messageErreur("Creation impossible :\nVeuillez remplir le formulaire.");
00670         return false;
00671     }
00672 }
```

8.7.3.35 verifierNumeroDossardInscription()

```
bool IHMGestionCross::verifierNumeroDossardInscription (
            QString numeroDossard )  [private]
```

Références cbInscriptionListeCourse, gestionnaireBDD, labelMessageInscription, lineEditNumeroDossard, messageErreur(), GestionBDD : :recupererInformation(), et GestionBDD : :verifierDossard().

Référencé par ajouterNouvelleInscription().

```
00511 {
00512     if(!numeroDossard.isEmpty())
00513     {
00514         QString numeroDossard = lineEditNumeroDossard->text();
00515         if(numeroDossard.length() ==3)
00516         {
00517             QString idCourse = gestionnaireBDD->
      recupererInformation("idCourse", "Course", QString("Nom = '%1'").arg(
      cbInscriptionListeCourse->currentText()));
00518             QString conditionMin = QString("%1000").arg(idCourse);
00519             QString conditionMax = QString("%1099").arg(idCourse);
00520             if(numeroDossard > conditionMin && numeroDossard < conditionMax)
00521             {
00522                 if(gestionnaireBDD->verifierDossard(numeroDossard))
00523                 {
00524                     labelMessageInscription->setStyleSheet("color : #000000");
00525                     lineEditNumeroDossard->setStyleSheet("color : #000000");
00526                     labelMessageInscription->clear();
00527                     return true;
00528                 }
00529                 else
00530                 {
00531                     messageErreur("Le numéro de dossard n'est pas disponible");
00532                     return false;
00533                 }
00534             }
00535             else
00536             {
00537                 messageErreur(QString("Le numéro de dossard est érroné, il doit commencer par
       : %1 ").arg(idCourse));
00538                 return false;
00539             }
00540         }
```

```
00541         else
00542         {
00543             messageErreur("Le numéro de dossard ne doit contenir que 3 chiffres");
00544             return false;
00545         }
00546     }
00547     else
00548     {
00549         return false;
00550     }
00551 }
```

8.7.3.36 viderTableInscrit()

```
void IHMGestionCross::viderTableInscrit ( )  [private]
```

Références modeleTableInscrits, et nomColonnesInscrit.

Référencé par selectionnerCourse().

```
00495 {
00496     modeleTableInscrits->clear();
00497     modeleTableInscrits->setHorizontalHeaderLabels(
      nomColonnesInscrit);
00498 }
```

8.7.4 Documentation des données membres

8.7.4.1 bAnnulerDialog

```
QPushButton* IHMGestionCross::bAnnulerDialog  [private]
```

Référencé par initialiserConfirmationDialog().

8.7.4.2 bConfirmationDialog

```
QPushButton* IHMGestionCross::bConfirmationDialog  [private]
```

Référencé par initialiserConfirmationDialog().

8.7.4.3 bCoureurs

```
QPushButton* IHMGestionCross::bCoureurs  [private]
```

Référencé par gererCoureurs(), gererCourses(), gererManifestations(), et IHMGestionCross().

8.7.4.4 bCourses

```
QPushButton* IHMGestionCross::bCourses  [private]
```

Référencé par gererCoureurs(), gererCourses(), gererManifestations(), et IHMGestionCross().

8.7.4.5 bCreationAnnuler

```
QPushButton* IHMGestionCross::bCreationAnnuler  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), et reinitialiserGestionCoureur().

8.7.4.6 bCreationConfirmer

```
QPushButton* IHMGestionCross::bCreationConfirmer  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), et reinitialiserGestionCoureur().

8.7.4.7 bGestionModifier

```
QPushButton* IHMGestionCross::bGestionModifier  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), et selectionnerCoureur().

8.7.4.8 bGestionNouveau

```
QPushButton* IHMGestionCross::bGestionNouveau  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), et reinitialiserGestionCoureur().

8.7.4.9 bGestionSupprimer

```
QPushButton* IHMGestionCross::bGestionSupprimer  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), et selectionnerCoureur().

8.7.4.10 bInscrire

```
QPushButton* IHMGestionCross::bInscrire  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), listerManifestations(), et selectionnerCourse().

8.7.4.11 bManifestations

```
QPushButton* IHMGestionCross::bManifestations  [private]
```

Référencé par gererCoureurs(), gererCourses(), gererManifestations(), et IHMGestionCross().

8.7.4.12 cbGestionCategorie

```
QComboBox* IHMGestionCross::cbGestionCategorie  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), reinitialiserGestionCoureur(), selectionnerCoureur(), verifierInformationsCreerCoureur(), et verifierInformationsModifierCoureur().

8.7.4.13 cbGestionClasse

```
QComboBox* IHMGestionCross::cbGestionClasse  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), reinitialiserGestionCoureur(), selectionnerCoureur(), verifierInformationsCreerCoureur(), et verifierInformationsModifierCoureur().

8.7.4.14 cbGestionParticipe

```
QComboBox* IHMGestionCross::cbGestionParticipe  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.15 cbInscriptionListeCourse

```
QComboBox* IHMGestionCross::cbInscriptionListeCourse  [private]
```

Référencé par ajouterNouvelleInscription(), IHMGestionCross(), initialiserFenetreCoureur(), listerCourses(), listerManifestations(), passerModeNouveauCoureur(), selectionnerCoureur(), selectionnerCourse(), selectionnerManifestation(), et verifierNumero↩DossardInscription().

8.7.4.16 cbInscriptionListeManifestation

```
QComboBox* IHMGestionCross::cbInscriptionListeManifestation  [private]
```

Référencé par IHMGestionCross(), initialiserFenetreCoureur(), listerManifestations(), passerModeNouveauCoureur(), et selectionner↩Coureur().

8.7.4.17 confirmationDialog

```
QDialog* IHMGestionCross::confirmationDialog  [private]
```

Référencé par confirmerDialog(), initialiserConfirmationDialog(), et quitterDialog().

8.7.4.18 dateDefault

```
QDate IHMGestionCross::dateDefault  [private]
```

Référencé par IHMGestionCross(), passerModeNouveauCoureur(), et reinitialiserGestionCoureur().

8.7.4.19 deDateNaissance

```
QDateEdit* IHMGestionCross::deDateNaissance  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.20 etat

```
QString IHMGestionCross::etat  [private]
```

Référencé par confirmerDialog(), gererCoureurs(), gererCourses(), gererManifestations(), initialiserConfirmationDialog(), modifierCoureur(), passerModeNouveauCoureur(), reinitialiserGestionCoureur(), selectionnerCoureur(), selectionnerManifestation(), et supprimerCoureur().

8.7.4.21 fenetreCoureur

```
QWidget* IHMGestionCross::fenetreCoureur  [private]
```

Référencé par IHMGestionCross(), et initialiserFenetreCoureur().

8.7.4.22 fenetreCourse

```
QWidget* IHMGestionCross::fenetreCourse  [private]
```

Référencé par IHMGestionCross().

8.7.4.23 fenetreGestionCross

```
QStackedWidget* IHMGestionCross::fenetreGestionCross  [private]
```

Référencé par gererCoureurs(), gererCourses(), gererManifestations(), et IHMGestionCross().

8.7.4.24 fenetreManifestation

```
QWidget* IHMGestionCross::fenetreManifestation  [private]
```

Référencé par IHMGestionCross().

8.7.4.25 gestionnaireBDD

```
GestionBDD* IHMGestionCross::gestionnaireBDD  [private]
```

Référencé par afficherTable(), ajouterNouvelleInscription(), confirmerDialog(), creerCoureur(), IHMGestionCross(), initialiserConfirmationDialog(), listerCourses(), listerManifestations(), mettreAJourTableCoureur(), passerModeNouveauCoureur(), selectionnerCoureur(), selectionnerCourse(), verifierInformationsCreerCoureur(), verifierInformationsModifierCoureur(), et verifierNumeroDossardInscription().

8.7.4.26 idCoureur

```
QString IHMGestionCross::idCoureur  [private]
```

Référencé par ajouterNouvelleInscription(), confirmerDialog(), et selectionnerCoureur().

8.7.4.27 labelConfirmationDialog

```
QLabel* IHMGestionCross::labelConfirmationDialog  [private]
```

Référencé par initialiserConfirmationDialog().

8.7.4.28 labelGestion

```
QLabel* IHMGestionCross::labelGestion  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.29 labelGestionCategorie

```
QLabel* IHMGestionCross::labelGestionCategorie  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.30 labelGestionClasse

```
QLabel* IHMGestionCross::labelGestionClasse  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.31 labelGestionDateNaissance

```
QLabel* IHMGestionCross::labelGestionDateNaissance  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.32 labelGestionINE

```
QLabel* IHMGestionCross::labelGestionINE  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.33 labelGestionNom

```
QLabel* IHMGestionCross::labelGestionNom  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.34 labelGestionParticipe

```
QLabel* IHMGestionCross::labelGestionParticipe  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.35 labelGestionPrenom

```
QLabel* IHMGestionCross::labelGestionPrenom  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.36 labelGestionSexe

```
QLabel* IHMGestionCross::labelGestionSexe  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.37 labelInscription

```
QLabel* IHMGestionCross::labelInscription  [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.38 labelInscriptionCourse

```
QLabel* IHMGestionCross::labelInscriptionCourse [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.39 labelInscriptionManifestation

```
QLabel* IHMGestionCross::labelInscriptionManifestation [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.40 labelMessageInscription

```
QLabel* IHMGestionCross::labelMessageInscription [private]
```

Référencé par initialiserFenetreCoureur(), messageErreur(), messageSucces(), selectionnerCoureur(), selectionnerCourse(), verifierInformationsCreerCoureur(), verifierInformationsModifierCoureur(), et verifierNumeroDossardInscription().

8.7.4.41 labelNumeroDossard

```
QLabel* IHMGestionCross::labelNumeroDossard [private]
```

Référencé par initialiserFenetreCoureur().

8.7.4.42 lineEditINE

```
QLineEdit* IHMGestionCross::lineEditINE [private]
```

Référencé par confirmerDialog(), initialiserConfirmationDialog(), initialiserFenetreCoureur(), listerManifestations(), passerModeNouveauCoureur(), recupererChampsGestion(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.43 lineEditNom

```
QLineEdit* IHMGestionCross::lineEditNom [private]
```

Référencé par ajouterNouvelleInscription(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.44 lineEditNumeroDossard

```
QLineEdit* IHMGestionCross::lineEditNumeroDossard [private]
```

Référencé par ajouterNouvelleInscription(), initialiserFenetreCoureur(), messageErreur(), selectionnerCoureur(), selectionnerCourse(), et verifierNumeroDossardInscription().

8.7.4.45 lineEditPrenom

```
QLineEdit* IHMGestionCross::lineEditPrenom  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.46 listeCategories

```
QVector<QStringList> IHMGestionCross::listeCategories  [private]
```

8.7.4.47 listeClasses

```
QVector<QStringList> IHMGestionCross::listeClasses  [private]
```

8.7.4.48 listeCoureurs

```
QVector<QStringList> IHMGestionCross::listeCoureurs  [private]
```

8.7.4.49 listeCoureursInscrit

```
QVector<QStringList> IHMGestionCross::listeCoureursInscrit  [private]
```

Référencé par selectionnerCourse().

8.7.4.50 listeCourses

```
QVector<QString> IHMGestionCross::listeCourses  [private]
```

Référencé par listerCourses(), et selectionnerManifestation().

8.7.4.51 listeManifestations

```
QVector<QString> IHMGestionCross::listeManifestations  [private]
```

Référencé par listerCourses(), listerManifestations(), et selectionnerCourse().

8.7.4.52 logoChronoCross

```
QLabel* IHMGestionCross::logoChronoCross  [private]
```

Référencé par IHMGestionCross().

8.7.4.53 modeleTableCoureurs

```
QStandardItemModel* IHMGestionCross::modeleTableCoureurs  [private]
```

Référencé par afficherTable(), ajouterNouveauCoureurTable(), gererCoureurs(), et initialiserFenetreCoureur().

8.7.4.54 modeleTableInscrits

```
QStandardItemModel* IHMGestionCross::modeleTableInscrits  [private]
```

Référencé par initialiserFenetreCoureur(), mettreAJourTableInscrit(), selectionnerCoureur(), selectionnerCourse(), et viderTable↩Inscrit().

8.7.4.55 nbLignesTableCoureurs

```
int IHMGestionCross::nbLignesTableCoureurs  [private]
```

Référencé par afficherTable(), ajouterNouveauCoureurTable(), et IHMGestionCross().

8.7.4.56 nbLignesTableInscrit

```
int IHMGestionCross::nbLignesTableInscrit  [private]
```

Référencé par mettreAJourTableInscrit().

8.7.4.57 nomColonnesCoureur

```
QStringList IHMGestionCross::nomColonnesCoureur  [private]
```

Référencé par gererCoureurs().

8.7.4.58 nomColonnesInscrit

```
QStringList IHMGestionCross::nomColonnesInscrit  [private]
```

Référencé par initialiserFenetreCoureur(), et viderTableInscrit().

8.7.4.59 rbGestionSexeF

```
QRadioButton* IHMGestionCross::rbGestionSexeF  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), recupererSexeCoureur(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.60 rbGestionSexeM

```
QRadioButton* IHMGestionCross::rbGestionSexeM  [private]
```

Référencé par initialiserFenetreCoureur(), passerModeNouveauCoureur(), recupererChampsGestion(), recupererSexeCoureur(), reinitialiserGestionCoureur(), et selectionnerCoureur().

8.7.4.61 tableCoureurs

```
QVector<QStringList> IHMGestionCross::tableCoureurs  [private]
```

Référencé par afficherTable(), mettreAJourTableCoureur(), et selectionnerCoureur().

8.7.4.62 vueTableCoureurs

```
QTableView* IHMGestionCross::vueTableCoureurs  [private]
```

Référencé par gererCoureurs(), IHMGestionCross(), initialiserFenetreCoureur(), passerModeNouveauCoureur(), et reinitialiserGestionCoureur().

8.7.4.63 vueTableInscrits

```
QTableView* IHMGestionCross::vueTableInscrits  [private]
```

Référencé par initialiserFenetreCoureur().

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

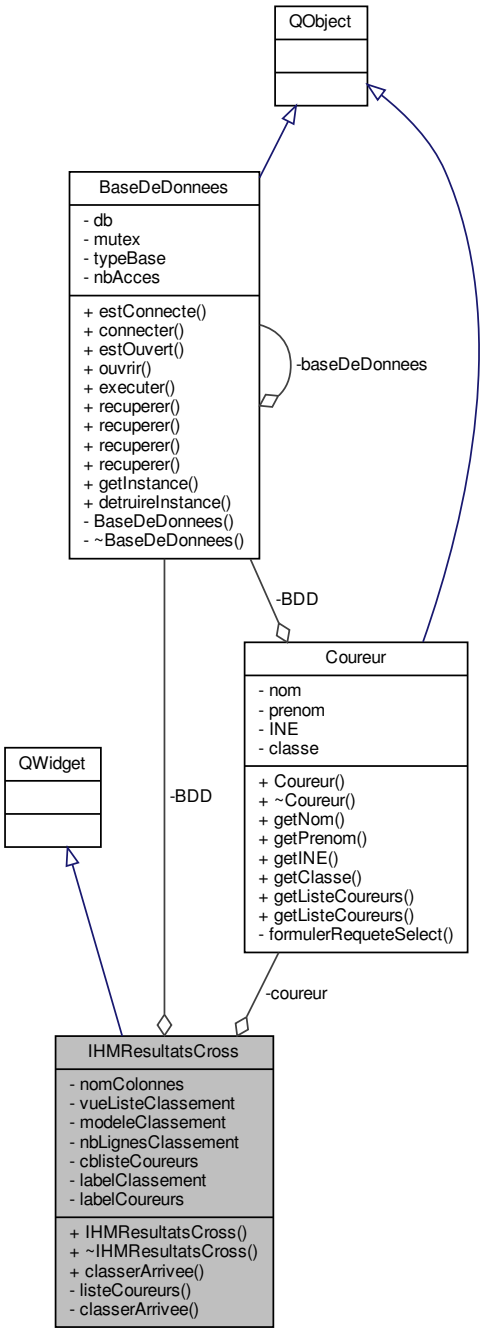
- ihmgestioncross.h
- ihmgestioncross.cpp

8.8 Référence de la classe IHMResultatsCross

La fenêtre principale de l'application Resultats-Cross.

```
#include <ihmresultatscross.h>
```

Graphe de collaboration de IHMResultatsCross :



Fonctions membres publiques

- IHMResultatsCross (QWidget *parent=nullptr)
  Constructeur de la fenêtre principale.
- ~IHMResultatsCross ()
  Destructeur de la fenêtre principale.
- void classerArrivee ()

Connecteurs privés

- void classerArrivee (QStringList classementCoureurs)

Fonctions membres privées

— void listeCoureurs ()

Attributs privés

— QStringList nomColonnes
— BaseDeDonnees ∗ BDD
agrégation BaseDeDonnee
— QTableView ∗ vueListeClassement
Tableau classement.
— QStandardItemModel ∗ modeleClassement
— int nbLignesClassement
— Coureur ∗ coureur
agrégation coureur
— QComboBox ∗ cblisteCoureurs
Combobox contenant les différentes coureurs dans la base de données.
— QLabel ∗ labelClassement
— QLabel ∗ labelCoureurs

8.8.1 Description détaillée

La fenêtre principale de l'application Resultats-Cross.

Auteur

Suzie Turlin

Version

0.1

8.8.2 Documentation des constructeurs et destructeur

8.8.2.1 IHMResultatsCross()

```
IHMResultatsCross::IHMResultatsCross (
            QWidget * parent = nullptr )
```

Constructeur de la fenêtre principale.

Paramètres

parent	QObject Adresse de l'objet Qt parent (0 = pas de parent car c'est la fenêtre principale)

A faire Définir le contenu de l'IHM

Références BDD, cblisteCoureurs, BaseDeDonnees : :connecter(), coureur, BaseDeDonnees : :estConnecte(), BaseDeDonnees↩ : :getInstance(), labelClassement, labelCoureurs, modeleClassement, nomColonnes, TAILLETEXTELABEL, et vueListeClassement.

```
00025                                                   : QWidget(parent)
00026 {
00031     coureur = new Coureur(this);
```

```
00032
00033     BDD = BaseDeDonnees::getInstance();
00034     if(!BDD->estConnecte())
00035         BDD->connecter("Resultats-Cross");
00036
00037     vueListeClassement = new QTableView(this);
00038     modeleClassement = new QStandardItemModel(1, 4);
00039     nomColonnes << "Nom" << "Prénom" << "Classe" << "INE";
00040     modeleClassement->setHorizontalHeaderLabels(nomColonnes);
00041     vueListeClassement->setModel(modeleClassement);
00042     vueListeClassement->setEditTriggers(QAbstractItemView::NoEditTriggers);
00043     vueListeClassement->setFixedSize(this->width(), this->height());
00044
00045     vueListeClassement->show();
00046
00047     // les widgets
00048
00049     // défini la taille du text dans les QPushButtons
00050     /*QFont texteBouton;
00051     texteBouton.setPointSize(TAILLETEXTEBUTON);
00052
00053     bAjouter = new QPushButton(QString::fromUtf8("Démarrer"), this);
00054     bAjouter->setDefault(false);
00055     bAjouter->setEnabled(false);
00056     bAjouter->setFont(texteBouton);*/
00057
00058    /* cblisteCoureurs = new QComboBox(this);
00059     cblisteCoureurs->setFixedSize(this->width()*0.66, this->height()*0.08);
00060     cblisteCoureurs->addItem(("< Séléctionner Coureur >"));*/
00061
00062
00063     // défini la taille du text des labels
00064     QFont texteLabel;
00065     texteLabel.setPointSize(TAILLETEXTELABEL);
00066
00067     labelClassement = new QLabel(tr("Classement : "), this);
00068     labelClassement->setFont(texteLabel);
00069     labelCoureurs = new QLabel(tr("Coureurs : "), this);
00070     labelCoureurs->setFont(texteLabel);
00071
00072     QVBoxLayout *classementLayout = new QVBoxLayout;
00073     classementLayout->addWidget(labelClassement);
00074     classementLayout->addWidget(vueListeClassement);
00075
00076     // le positionnement des widgets
00077     QHBoxLayout *listesLayout = new QHBoxLayout;
00078     //QHBoxLayout *boutonsLayout = new QHBoxLayout;
00079
00080     listesLayout->addWidget(cblisteCoureurs);
00081     listesLayout->addWidget(labelCoureurs);
00082
00083     QVBoxLayout *mainLayout = new QVBoxLayout;
00084     mainLayout->addLayout(listesLayout);
00085
00086     setLayout(mainLayout);
00087     setWindowTitle(tr("Résultat-Cross"));
00088     setContextMenuPolicy(Qt::ActionsContextMenu);
00089
00090
00091     //boutonsLayout->addWidget(bAjouter);
00092     //boutonsLayout->setContentsMargins(0, 0, 0, 20); // G H D B
00093
00094     // Les labels
00095
00096     // Les connexions
00097
00098     // connect(bDemarrer, SIGNAL(clicked()), this, SLOT(coureur()));
00099
00100     showMaximized(); // Fenêtre d'ouverture maximale
00101 }
```

8.8.2.2 ~IHMResultatsCross()

`IHMResultatsCross::~IHMResultatsCross ( )`

Destructeur de la fenêtre principale.

Références BaseDeDonnees : :detruireInstance().

```
00110 {
00111     BaseDeDonnees::detruireInstance();
00112     qDebug() << Q_FUNC_INFO;
00113 }
```

8.8.3 Documentation des fonctions membres

8.8.3.1 classerArrivee() [1/2]

```
void IHMResultatsCross::classerArrivee ( )
```

8.8.3.2 classerArrivee [2/2]

```
void IHMResultatsCross::classerArrivee (
            QStringList classementCoureurs )  [private], [slot]
```

Références COLONNE_CLASSE, COLONNE_INE, COLONNE_NOM, COLONNE_PRENOM, INFO_COUREUR_CLASSE, INFO_COUREUR_INE, INFO_COUREUR_NOM, INFO_COUREUR_PRENOM, modeleClassement, nbLignesClassement, et vueListeClassement.

```
00117 {
00118     qDebug() << Q_FUNC_INFO << informationCoureur;
00119     //informationCoureur[Nom, Prenom, Classe, INE]
00120     // Redimensionner automatiquement la colonne pour occuper l'espace disponible
00121
00122     vueListeClassement->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
00123     nbLignesClassement = 1 - modeleClassement->rowCount();
00124
00125
00126     QStandardItem *nom = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_NOM));
00127     QStandardItem *prenom = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_PRENOM));
00128     QStandardItem *classe = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_CLASSE));
00129     QStandardItem *INE = new QStandardItem(informationCoureur.at(
      INFO_COUREUR_INE));
00130
00131
00132     modeleClassement->setItem(nbLignesClassement,
      COLONNE_NOM, nom);
00133     modeleClassement->setItem(nbLignesClassement,
      COLONNE_PRENOM, prenom);
00134     modeleClassement->setItem(nbLignesClassement,
      COLONNE_CLASSE, classe);
00135     modeleClassement->setItem(nbLignesClassement,
      COLONNE_INE, INE);
00136     nbLignesClassement += 1;
00137 }
```

8.8.3.3 listeCoureurs()

```
void IHMResultatsCross::listeCoureurs ( )  [private]
```

8.8.4 Documentation des données membres

8.8.4.1 BDD

```
BaseDeDonnees* IHMResultatsCross::BDD  [private]
```

agrégation BaseDeDonnee

Référencé par IHMResultatsCross().

8.8.4.2 cblisteCoureurs

```
QComboBox* IHMResultatsCross::cblisteCoureurs  [private]
```

Combobox contenant les différentes coureurs dans la base de données.

Référencé par IHMResultatsCross().

8.8.4.3 coureur

```
Coureur* IHMResultatsCross::coureur  [private]
```

agrégation coureur

Référencé par IHMResultatsCross().

8.8.4.4 labelClassement

```
QLabel* IHMResultatsCross::labelClassement  [private]
```

Référencé par IHMResultatsCross().

8.8.4.5 labelCoureurs

```
QLabel* IHMResultatsCross::labelCoureurs  [private]
```

Référencé par IHMResultatsCross().

8.8.4.6 modeleClassement

```
QStandardItemModel* IHMResultatsCross::modeleClassement  [private]
```

Référencé par classerArrivee(), et IHMResultatsCross().

8.8.4.7 nbLignesClassement

```
int IHMResultatsCross::nbLignesClassement  [private]
```

Référencé par classerArrivee().

8.8.4.8 nomColonnes

```
QStringList IHMResultatsCross::nomColonnes  [private]
```

Référencé par IHMResultatsCross().

8.8.4.9 vueListeClassement

```
QTableView* IHMResultatsCross::vueListeClassement  [private]
```

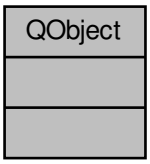
Tableau classement.

Référencé par classerArrivee(), et IHMResultatsCross().

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

8.9 Référence de la classe QObject

Graphe de collaboration de QObject :



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

8.10 Référence de la classe QWidget

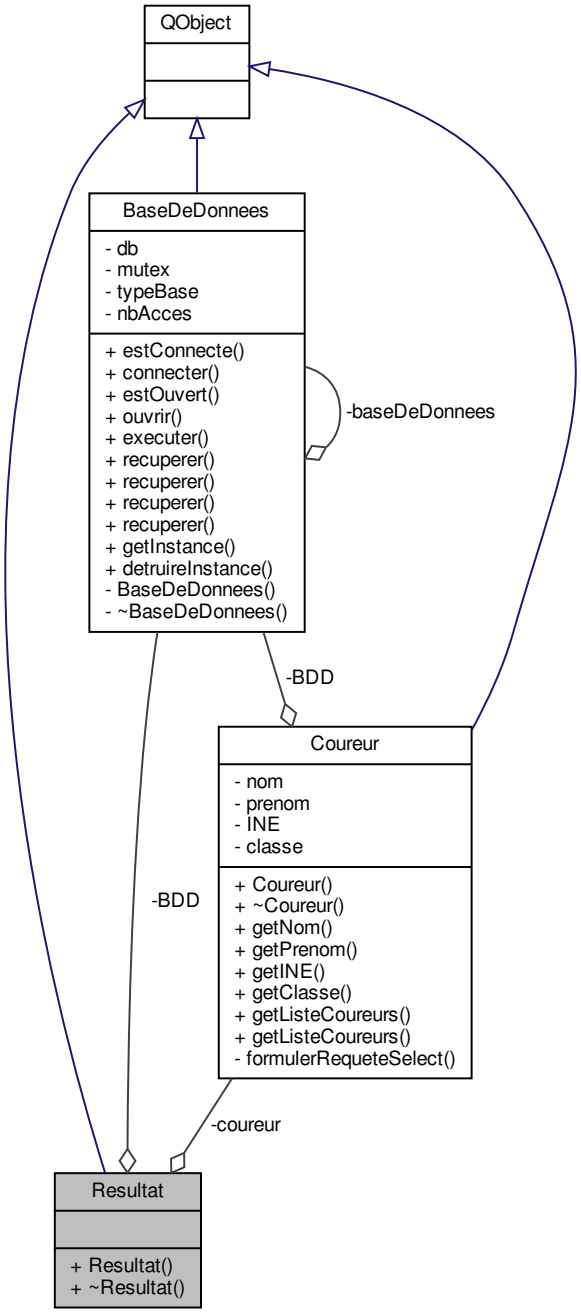
Graphe de collaboration de QWidget :

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

8.11 Référence de la classe Resultat

`#include <resultat.h>`

Graphe de collaboration de Resultat :



Fonctions membres publiques

- Resultat (QObject *parent=nullptr)
  Resultat.
- ~Resultat ()

Attributs privés

- BaseDeDonnees * BDD
  agrégation BaseDeDonnees
- Coureur * coureur
  association Coureur

8.11.1 Documentation des constructeurs et destructeur

8.11.1.1 Resultat()

```
Resultat::Resultat (
            QObject * parent = nullptr )  [explicit]
```

Resultat.

Paramètres

parent	QObject Adresse de l'objet Qt parent

Références BDD, BaseDeDonnees : :connecter(), BaseDeDonnees : :estConnecte(), et BaseDeDonnees : :getInstance().

```
00022                                   : QObject(parent)
00023 {
00024     BDD = BaseDeDonnees::getInstance();
00025     if(!BDD->estConnecte())
00026         BDD->connecter("Resultats-Cross");
00027     qDebug() << Q_FUNC_INFO << "Etat connection BDD : " << BDD->estConnecte();
00028 }
```

8.11.1.2 ~Resultat()

```
Resultat::~Resultat ( )
```

Références BaseDeDonnees : :detruireInstance().

```
00031 {
00032     BaseDeDonnees::detruireInstance();
00033     qDebug() << Q_FUNC_INFO;
00034 }
```

8.11.2 Documentation des données membres

8.11.2.1 BDD

```
BaseDeDonnees* Resultat::BDD  [private]
```

agrégation BaseDeDonnees

Référencé par Resultat().

8.11.2.2 coureur

```
Coureur* Resultat::coureur  [private]
```

association Coureur

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

9 Documentation des fichiers

9.1 Référence du fichier basededonnees.cpp

Définition de la classe BaseDeDonnees.

```
#include "basededonnees.h"
#include <QDebug>
#include <QMessageBox>
```

9.1.1 Description détaillée

Définition de la classe BaseDeDonnees.

Auteur

Thierry Vaira

Version

1.1

9.2 Référence du fichier basededonnees.h

Déclaration de la classe BaseDeDonnees.

```
#include <QObject>
#include <QtSql/QtSql>
#include <QSqlDatabase>
#include <QMutex>
#include <QString>
```

Classes

— class BaseDeDonnees

Macros

— #define BDD_HOSTNAME "192.168.52.149"
— #define BDD_USERNAME "organisateur"
— #define BDD_PASSWORD "password"

9.2.1 Description détaillée

Déclaration de la classe BaseDeDonnees.

Auteur

Thierry VAIRA

Version

1.1

9.2.2 Documentation des macros

9.2.2.1 BDD_HOSTNAME

```
#define BDD_HOSTNAME "192.168.52.149"
```

9.2.2.2 BDD_PASSWORD

```
#define BDD_PASSWORD "password"
```

9.2.2.3 BDD_USERNAME

```
#define BDD_USERNAME "organisateur"
```

9.3 Référence du fichier Changelog.md

9.4 Référence du fichier chrono.cpp

Définition de la classe Chrono.

```
#include "chrono.h"
#include "unistd.h"
```

9.4.1 Description détaillée

Définition de la classe Chrono.

Auteur

Michael Andreo

Version

1.1

9.5 Référence du fichier chrono.h

Déclaration de la classe Chrono.

```
#include <QObject>
#include <QDebug>
#include <QtSerialPort/QSerialPort>
```

Classes

— class Chrono
Déclaration de la classe Chrono.

Macros

— #define PORT "/dev/hl975"
— #define MODECLOCK "#WP 120 5\t01AF\r\n"
— #define NEWSYNCHRO "#WC 007 02 00 :00 01/01/01\t048E\r\n"
— #define NEWRUN "#WC 002\t014C\r\n"
— #define STARTMANUALSYNCHRO "#WC 008 01\t01D3\r\n"
— #define CLOSERUN "#WC 001\t014B\r\n"
— #define MODEND "#WP 120 3\t01AD\r\n"
— #define TRAME_ACQUITTEMENT "AK"
— #define TRAME_SYNCHRO "TS"
— #define TRAME_PARAMETRES "&P"
— #define TRAME_TEMPS "TN"
— #define TRAME_COURSE_TERMINEE "CL"
— #define ETAT_NONSYNCHRO 0
— #define ETAT_MODECLOCK 1
— #define ETAT_NEWSYNCHRO 2
— #define ETAT_NEWRUN 3
— #define ETAT_STARTMANUALSYNCHRO 4
— #define ETAT_CLOSERUN 5
— #define ETAT_MODEND 6
— #define CHAMPS_TRAME_TEMPS 3
— #define TIMEOUT 100000

9.5.1 Description détaillée

Déclaration de la classe Chrono.

Auteur

Michael Andréo

Version

1.1

9.5.2 Documentation des macros

9.5.2.1 CHAMPS_TRAME_TEMPS

```
#define CHAMPS_TRAME_TEMPS 3
```

Référencé par Chrono : :decoderTrame().

9.5.2.2 CLOSERUN

```
#define CLOSERUN "#WC 001\t014B\r\n"
```

Référencé par Chrono : :arreterCourse().

9.5.2.3 ETAT_CLOSERUN

```
#define ETAT_CLOSERUN 5
```

Référencé par Chrono : :decoderTrame().

9.5.2.4 ETAT_MODECLOCK

```
#define ETAT_MODECLOCK 1
```

Référencé par Chrono : :decoderTrame().

9.5.2.5 ETAT_MODEND

```
#define ETAT_MODEND 6
```

Référencé par Chrono : :decoderTrame().

9.5.2.6 ETAT_NEWRUN

```
#define ETAT_NEWRUN 3
```

Référencé par Chrono : :decoderTrame().

9.5.2.7 ETAT_NEWSYNCHRO

```
#define ETAT_NEWSYNCHRO 2
```

Référencé par Chrono : :decoderTrame().

9.5.2.8 ETAT_NONSYNCHRO

```
#define ETAT_NONSYNCHRO 0
```

Référencé par Chrono : :decoderTrame().

9.5.2.9 ETAT_STARTMANUALSYNCHRO

```
#define ETAT_STARTMANUALSYNCHRO 4
```

Référencé par Chrono : :decoderTrame().

9.5.2.10 MODECLOCK

```
#define MODECLOCK "#WP 120 5\t01AF\r\n"
```

Référencé par Chrono : :creer().

9.5.2.11 MODEND

```
#define MODEND "#WP 120 3\t01AD\r\n"
```

Référencé par Chrono : :arreterChrono().

9.5.2.12 NEWRUN

```
#define NEWRUN "#WC 002\t014C\r\n"
```

Référencé par Chrono : :creerClassement().

9.5.2.13 NEWSYNCHRO

```
#define NEWSYNCHRO "#WC 007 02 00:00 01/01/01\t048E\r\n"
```

Référencé par Chrono : :synchroniser().

9.5.2.14 PORT

```
#define PORT "/dev/hl975"
```

Référencé par Chrono : :Chrono().

9.5.2.15 STARTMANUALSYNCHRO

```
#define STARTMANUALSYNCHRO "#WC 008 01\t01D3\r\n"
```

Référencé par Chrono : :demarrer().

9.5.2.16 TIMEOUT

```
#define TIMEOUT 100000
```

Référencé par Chrono : :lireTrame(), et Chrono : :synchroniser().

9.5.2.17 TRAME_ACQUITTEMENT

```
#define TRAME_ACQUITTEMENT "AK"
```

Référencé par Chrono : :decoderTrame().

9.5.2.18 TRAME_COURSE_TERMINEE

```
#define TRAME_COURSE_TERMINEE "CL"
```

Référencé par Chrono : :decoderTrame().

9.5.2.19 TRAME_PARAMETRES

```
#define TRAME_PARAMETRES "&P"
```

9.5.2.20 TRAME_SYNCHRO

```
#define TRAME_SYNCHRO "TS"
```

9.5.2.21 TRAME_TEMPS

```
#define TRAME_TEMPS "TN"
```

Référencé par Chrono : :decoderTrame().

9.6 Référence du fichier coureur.cpp

Définition de la classe Coureur.

```
#include "coureur.h"
#include "ihmresultatscross.h"
#include "../BaseDeDonnees/basededonnees.h"
```

9.6.1 Description détaillée

Définition de la classe Coureur.

Auteur

Suzie Turlin

Version

0.1

9.7 Référence du fichier coureur.h

```
#include <QObject>
```

Classes

— class Coureur
Gérer les coureurs.

9.8 Référence du fichier course.cpp

Définition de la classe Course.

```
#include "course.h"
```

9.8.1 Description détaillée

Définition de la classe Course.

Auteur

Michael Andréo

Version

1.1

9.9 Référence du fichier course.h

Déclaration de la classe Course.

```
#include <QObject>
#include <QDebug>
#include <QDate>
#include "chrono.h"
#include "../BaseDeDonnees/basededonnees.h"
```

Classes

— class Course
Déclaration de la classe Course.

Macros

— #define INFORMATION_COUREUR_ARRIVEE_CLASSE 4

9.9.1 Description détaillée

Déclaration de la classe Course.

Auteur

Michael Andréo

Version

1.1

9.9.2 Documentation des macros

9.9.2.1 INFORMATION_COUREUR_ARRIVEE_CLASSE

```
#define INFORMATION_COUREUR_ARRIVEE_CLASSE 4
```

Référencé par Course : :getInformationCoureur().

9.10 Référence du fichier gestionbdd.cpp

Définition de la classe GestionBDD.

```
#include "gestionbdd.h"
```

9.10.1 Description détaillée

Définition de la classe GestionBDD.

Auteur

ANDRÉO Michaël

Version

1.0

9.11 Référence du fichier gestionbdd.h

Déclaration de la classe GestionBDD.

```
#include <QObject>
#include <QDebug>
#include "../BaseDeDonnees/basededonnees.h"
```

Classes

— class GestionBDD
Déclaration de la classe GestionBDD.

9.11.1 Description détaillée

Déclaration de la classe GestionBDD.

Auteur

Michael Andréo

Version

1.0

9.12 Référence du fichier ihmchronocross.cpp

Définition de la classe IHMChronoCross.

```
#include "ihmchronocross.h"
#include "../BaseDeDonnees/basededonnees.h"
#include "course.h"
```

9.12.1 Description détaillée

Définition de la classe IHMChronoCross.

Auteur

ANDRÉO Michaël

Version

1.1

9.13 Référence du fichier ihmchronocross.h

Déclaration de la classe IHMChronoCross.

```
#include <QtWidgets>
#include <QMainWindow>
#include <QDebug>
#include <QTimer>
#include <QDialog>
```

Classes

— class IHMChronoCross
La fenêtre principale de l'application Chrono-Cross.

Macros

— #define TAILLETEXTELABEL 20
— #define TAILLETEXTEBUTON 20
— #define TAILLETEXTELISTE 16
— #define TAILLETEXTEINFO 18
— #define TAILLETEXTESUPPRIMER 15
— #define TAILLETEXTECLASSEMENT 12
— #define IMAGECHRONOCROSS "../image/icone-chrono-cross.png"
— #define CLASSEMENT 0
— #define TEMPS 1
— #define DOSSARD 2
— #define COLONNE_TEMPS 0
— #define COLONNE_DOSSARD 1
— #define COLONNE_NOM 2
— #define COLONNE_PRENOM 3
— #define COLONNE_CLASSE 4
— #define INFO_COUREUR_TEMPS 0
— #define INFO_COUREUR_DOSSARD 1
— #define INFO_COUREUR_NOM 2
— #define INFO_COUREUR_PRENOM 3
— #define INFO_COUREUR_CLASSE 4
— #define NUMERO_DOSSARD_INVALIDE 0
— #define DOSSARD_VALIDE_COURSE_INVALIDE 1
— #define DOSSARD_DEJA_ARRIVE 2
— #define DOSSARD_VALIDE 3

9.13.1 Description détaillée

Déclaration de la classe IHMChronoCross.

Auteur

Michael Andréo

Version

1.1

9.13.2 Documentation des macros

9.13.2.1 CLASSEMENT

```
#define CLASSEMENT 0
```

9.13.2.2 COLONNE_CLASSE

```
#define COLONNE_CLASSE 4
```

Référencé par IHMGestionCross : :afficherTable(), IHMGestionCross : :ajouterNouveauCoureurTable(), IHMResultatsCross↩ : :classerArrivee(), et IHMChronoCross : :classerArrivee().

9.13.2.3 COLONNE_DOSSARD

```
#define COLONNE_DOSSARD 1
```

Référencé par IHMChronoCross : :classerArrivee().

9.13.2.4 COLONNE_NOM

```
#define COLONNE_NOM 2
```

Référencé par IHMGestionCross : :afficherTable(), IHMGestionCross : :ajouterNouveauCoureurTable(), IHMResultatsCross↩ : :classerArrivee(), IHMChronoCross : :classerArrivee(), IHMGestionCross : :mettreAJourTableInscrit(), et IHMGestionCross↩ : :selectionnerCourse().

9.13.2.5 COLONNE_PRENOM

```
#define COLONNE_PRENOM 3
```

Référencé par IHMGestionCross : :afficherTable(), IHMGestionCross : :ajouterNouveauCoureurTable(), IHMResultatsCross↩ : :classerArrivee(), IHMChronoCross : :classerArrivee(), IHMGestionCross : :mettreAJourTableInscrit(), et IHMGestionCross↩ : :selectionnerCourse().

9.13.2.6 COLONNE_TEMPS

```
#define COLONNE_TEMPS 0
```

Référencé par IHMChronoCross : :classerArrivee().

9.13.2.7 DOSSARD

```
#define DOSSARD 2
```

9.13.2.8 DOSSARD_DEJA_ARRIVE

```
#define DOSSARD_DEJA_ARRIVE 2
```

Référencé par IHMChronoCross : :associerArriveeDossard().

9.13.2.9 DOSSARD_VALIDE

```
#define DOSSARD_VALIDE 3
```

Référencé par IHMChronoCross : :associerArriveeDossard().

9.13.2.10 DOSSARD_VALIDE_COURSE_INVALIDE

```
#define DOSSARD_VALIDE_COURSE_INVALIDE 1
```

Référencé par IHMChronoCross : :associerArriveeDossard().

9.13.2.11 IMAGECHRONOCROSS

```
#define IMAGECHRONOCROSS "../image/icone-chrono-cross.png"
```

Référencé par IHMChronoCross : :IHMChronoCross(), et IHMGestionCross : :IHMGestionCross().

9.13.2.12 INFO_COUREUR_CLASSE

```
#define INFO_COUREUR_CLASSE 4
```

Référencé par IHMGestionCross : :afficherTable(), IHMResultatsCross : :classerArrivee(), IHMChronoCross : :classerArrivee(), et IHMGestionCross : :selectionnerCoureur().

9.13.2.13 INFO_COUREUR_DOSSARD

```
#define INFO_COUREUR_DOSSARD 1
```

Référencé par IHMChronoCross : :classerArrivee().

9.13.2.14 INFO_COUREUR_NOM

```
#define INFO_COUREUR_NOM 2
```

Référencé par IHMGestionCross : :afficherTable(), IHMResultatsCross : :classerArrivee(), IHMChronoCross : :classerArrivee(), et IHMGestionCross : :selectionnerCoureur().

9.13.2.15 INFO_COUREUR_PRENOM

```
#define INFO_COUREUR_PRENOM 3
```

Référencé par IHMGestionCross : :afficherTable(), IHMResultatsCross : :classerArrivee(), IHMChronoCross : :classerArrivee(), et IHMGestionCross : :selectionnerCoureur().

9.13.2.16 INFO_COUREUR_TEMPS

```
#define INFO_COUREUR_TEMPS 0
```

Référencé par IHMChronoCross : :classerArrivee().

9.13.2.17 NUMERO_DOSSARD_INVALIDE

```
#define NUMERO_DOSSARD_INVALIDE 0
```

Référencé par IHMChronoCross : :associerArriveeDossard().

9.13.2.18 TAILLETEXTEBUTON

```
#define TAILLETEXTEBUTON 20
```

Référencé par IHMChronoCross : :IHMChronoCross().

9.13.2.19 TAILLETEXTECLASSEMENT

```
#define TAILLETEXTECLASSEMENT 12
```

Référencé par IHMChronoCross : :personnaliserAffichageArrivee().

9.13.2.20 TAILLETEXTEINFO

```
#define TAILLETEXTEINFO 18
```

Référencé par IHMChronoCross : :IHMChronoCross().

9.13.2.21 TAILLETEXTELABEL

```
#define TAILLETEXTELABEL 20
```

Référencé par IHMChronoCross : :IHMChronoCross(), et IHMResultatsCross : :IHMResultatsCross().

9.13.2.22 TAILLETEXTELISTE

```
#define TAILLETEXTELISTE 16
```

Référencé par IHMChronoCross : :IHMChronoCross().

9.13.2.23 TAILLETEXTESUPPRIMER

```
#define TAILLETEXTESUPPRIMER 15
```

Référencé par IHMChronoCross : :IHMChronoCross().

9.13.2.24 TEMPS

```
#define TEMPS 1
```

9.14 Référence du fichier ihmgestioncross.cpp

Définition de la classe IHMGestionCross.

```
#include <ihmgestioncross.h>
```

9.14.1 Description détaillée

Définition de la classe IHMGestionCross.

Auteur

ANDRÉO Michaël

Version

1.0

9.15 Référence du fichier ihmgestioncross.h

Déclaration de la classe IHMGestionCross.

```
#include <QtWidgets>
#include <QMainWindow>
#include <QDebug>
#include <QDialog>
#include <QLayout>
#include "gestionbdd.h"
```

Classes

— class IHMGestionCross
La fenêtre principale de l'application Gestion-Cross.

Macros

- #define FENETRE_MANIFESTATION 0
- #define FENETRE_COURSE 1
- #define FENETRE_COUREUR 2
- #define IMAGECHRONOCROSS "../image/icone-chrono-cross.png"
- #define TAILLETEXTETITRE 20
- #define TAILLETEXTEGESTION 15
- #define TAILLETEXTEINSCRIPTION 15
- #define TAILLETEXTEBOUTONTITRE 17
- #define TAILLETEXTEBOUTONGESTION 13
- #define INFO_COUREUR_ID 0
- #define INFO_COUREUR_CATEGORIE 1
- #define INFO_COUREUR_CLASSE 2
- #define INFO_COUREUR_INE 3
- #define INFO_COUREUR_NOM 4
- #define INFO_COUREUR_PRENOM 5
- #define INFO_COUREUR_DATENAISSANCE 6
- #define INFO_COUREUR_SEXE 7
- #define INFO_COUREURINSCRIT_NOM 0
- #define INFO_COUREURINSCRIT_PRENOM 1
- #define INFO_COUREURINSCRIT_NUMERODOSSAD 2
- #define COLONNE_NOM 0
- #define COLONNE_PRENOM 1
- #define COLONNE_NUMERODOSSARD 2
- #define COLONNE_CLASSE 2
- #define COLONNE_CATEGORIE 3
- #define COLONNE_INE 4
- #define COLONNE_DATENAISSANCE 5
- #define COLONNE_SEXE 6

9.15.1 Description détaillée

Déclaration de la classe IHMGestionCross.

Auteur

Michael Andréo

Version

1.0

9.15.2 Documentation des macros

9.15.2.1 COLONNE_CATEGORIE

```
#define COLONNE_CATEGORIE 3
```

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :ajouterNouveauCoureurTable().

9.15.2.2 COLONNE_CLASSE

```
#define COLONNE_CLASSE 2
```

9.15.2.3 COLONNE_DATENAISSANCE

```
#define COLONNE_DATENAISSANCE 5
```

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :ajouterNouveauCoureurTable().

9.15.2.4 COLONNE_INE

```
#define COLONNE_INE 4
```

Référencé par IHMGestionCross : :afficherTable(), IHMGestionCross : :ajouterNouveauCoureurTable(), et IHMResultatsCross↩ : :classerArrivee().

9.15.2.5 COLONNE_NOM

```
#define COLONNE_NOM 0
```

9.15.2.6 COLONNE_NUMERODOSSARD

```
#define COLONNE_NUMERODOSSARD 2
```

Référencé par IHMGestionCross : :mettreAJourTableInscrit(), et IHMGestionCross : :selectionnerCourse().

9.15.2.7 COLONNE_PRENOM

```
#define COLONNE_PRENOM 1
```

9.15.2.8 COLONNE_SEXE

```
#define COLONNE_SEXE 6
```

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :ajouterNouveauCoureurTable().

9.15.2.9 FENETRE_COUREUR

```
#define FENETRE_COUREUR 2
```

Référencé par IHMGestionCross : :gererCoureurs().

9.15.2.10 FENETRE_COURSE

```
#define FENETRE_COURSE 1
```

Référencé par IHMGestionCross : :gererCourses().

9.15.2.11 FENETRE_MANIFESTATION

```
#define FENETRE_MANIFESTATION 0
```

Référencé par IHMGestionCross : :gererManifestations().

9.15.2.12 IMAGECHRONOCROSS

```
#define IMAGECHRONOCROSS "../image/icone-chrono-cross.png"
```

9.15.2.13 INFO_COUREUR_CATEGORIE

```
#define INFO_COUREUR_CATEGORIE 1
```

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :selectionnerCoureur().

9.15.2.14 INFO_COUREUR_CLASSE

```
#define INFO_COUREUR_CLASSE 2
```

9.15.2.15 INFO_COUREUR_DATENAISSANCE

```
#define INFO_COUREUR_DATENAISSANCE 6
```

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :selectionnerCoureur().

9.15.2.16 INFO_COUREUR_ID

```
#define INFO_COUREUR_ID 0
```

9.15.2.17 INFO_COUREUR_INE

```
#define INFO_COUREUR_INE 3
```

Référencé par IHMGestionCross : :afficherTable(), IHMResultatsCross : :classerArrivee(), et IHMGestionCross : :selectionner↩ Coureur().

9.15.2.18 INFO_COUREUR_NOM

```
#define INFO_COUREUR_NOM 4
```

9.15.2.19 INFO_COUREUR_PRENOM

```
#define INFO_COUREUR_PRENOM 5
```

9.15.2.20 INFO_COUREUR_SEXE

```
#define INFO_COUREUR_SEXE 7
```

Référencé par IHMGestionCross : :afficherTable(), et IHMGestionCross : :selectionnerCoureur().

9.15.2.21 INFO_COUREURINSCRIT_NOM

```
#define INFO_COUREURINSCRIT_NOM 0
```

Référencé par IHMGestionCross : :mettreAJourTableInscrit(), et IHMGestionCross : :selectionnerCourse().

9.15.2.22 INFO_COUREURINSCRIT_NUMERODOSSAD

```
#define INFO_COUREURINSCRIT_NUMERODOSSAD 2
```

Référencé par IHMGestionCross : :mettreAJourTableInscrit(), et IHMGestionCross : :selectionnerCourse().

9.15.2.23 INFO_COUREURINSCRIT_PRENOM

```
#define INFO_COUREURINSCRIT_PRENOM 1
```

Référencé par IHMGestionCross : :mettreAJourTableInscrit(), et IHMGestionCross : :selectionnerCourse().

9.15.2.24 TAILLETEXTEBOUTONGESTION

```
#define TAILLETEXTEBOUTONGESTION 13
```

Référencé par IHMGestionCross : :initialiserFenetreCoureur().

9.15.2.25 TAILLETEXTEBOUTONTITRE

```
#define TAILLETEXTEBOUTONTITRE 17
```

Référencé par IHMGestionCross : :IHMGestionCross().

9.15.2.26 TAILLETEXTEGESTION

```
#define TAILLETEXTEGESTION 15
```

Référencé par IHMGestionCross : :initialiserFenetreCoureur().

9.15.2.27 TAILLETEXTEINSCRIPTION

```
#define TAILLETEXTEINSCRIPTION 15
```

Référencé par IHMGestionCross : :initialiserFenetreCoureur().

9.15.2.28 TAILLETEXTETITRE

```
#define TAILLETEXTETITRE 20
```

Référencé par IHMGestionCross : :initialiserFenetreCoureur().

9.16 Référence du fichier ihmresultatscross.cpp

Définition de la classe IHMResultatsCross.

```
#include "ihmresultatscross.h"
#include "../BaseDeDonnees/basededonnees.h"
#include "coureur.h"
```

9.16.1 Description détaillée

Définition de la classe IHMResultatsCross.

Définition de la classe Resultat.

Auteur

Suzie Turlin

Version

0.1

9.17 Référence du fichier ihmresultatscross.h

Déclaration de la classe IHMResultatsCross.

```
#include <QtWidgets>
#include <QMainWindow>
#include <QDebug>
```

Classes

— class IHMResultatsCross
La fenêtre principale de l'application Resultats-Cross.

Macros

— #define TAILLETEXTELABEL 20
— #define TAILLETEXTEBUTON 20
— #define HAUTEUR_TABLEAU 1
— #define COLONNE_INE 0
— #define COLONNE_NOM 1
— #define COLONNE_PRENOM 2
— #define COLONNE_CLASSE 3
— #define INFO_COUREUR_INE 0
— #define INFO_COUREUR_NOM 1
— #define INFO_COUREUR_PRENOM 2
— #define INFO_COUREUR_CLASSE 3

9.17.1 Description détaillée

Déclaration de la classe IHMResultatsCross.

Auteur

Suzie Turlin

Version

0.1

9.17.2 Documentation des macros

9.17.2.1 COLONNE_CLASSE

```
#define COLONNE_CLASSE 3
```

9.17.2.2 COLONNE_INE

```
#define COLONNE_INE 0
```

9.17.2.3 COLONNE_NOM

```
#define COLONNE_NOM 1
```

9.17.2.4 COLONNE_PRENOM

```
#define COLONNE_PRENOM 2
```

9.17.2.5 HAUTEUR_TABLEAU

```
#define HAUTEUR_TABLEAU 1
```

9.17.2.6 INFO_COUREUR_CLASSE

```
#define INFO_COUREUR_CLASSE 3
```

9.17.2.7 INFO_COUREUR_INE

```
#define INFO_COUREUR_INE 0
```

9.17.2.8 INFO_COUREUR_NOM

```
#define INFO_COUREUR_NOM 1
```

9.17.2.9 INFO_COUREUR_PRENOM

```
#define INFO_COUREUR_PRENOM 2
```

9.17.2.10 TAILLETEXTEBUTON

```
#define TAILLETEXTEBUTON 20
```

9.17.2.11 TAILLETEXTELABEL

```
#define TAILLETEXTELABEL 20
```

9.18 Référence du fichier INSTALL.md

9.19 Référence du fichier main.cpp

Programme principal Chrono-Cross.

```
#include <QApplication>
#include "ihmchronocross.h"
```

Fonctions

— int main (int argc, char ∗argv[ ])

9.19.1 Description détaillée

Programme principal Chrono-Cross.

Chronomètre les courses et classe les coureurs à l'arrivée

Auteur

ANDREO Michaël andreo.michael@outlook.fr

Version

1.1

9.19.2 Documentation des fonctions

9.19.2.1 main()

```
main (
            int argc,
            char * argv[] )
```

Paramètres

argc	
argv[]	

Renvoie

int

```
00023 {
00024     QApplication a(argc, argv);
00025
00026     IHMChronoCross w; // instancie la fenêtre principale de l'application
00027     w.show(); // affiche la fenêtre principale de l'application
00028
00029     return a.exec(); // exécute l'application
00030 }
```

9.20 Référence du fichier main.cpp

Programme principal Gestion-Cross.

```
#include "ihmgestioncross.h"
#include <QApplication>
```

Fonctions

— int main (int argc, char ∗argv[ ])

9.20.1 Description détaillée

Programme principal Gestion-Cross.

Gère des manifestation de courses de Cross

Auteur

ANDREO Michaël andreo.michael@outlook.fr

Version

1.1

9.20.2 Documentation des fonctions

9.20.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

```
00023 {
00024     QApplication a(argc, argv);
00025     IHMGestionCross w;
00026     w.show();
00027
00028     return a.exec();
00029 }
```

9.21 Référence du fichier main.cpp

Programme principal Resultats-Cross (Raspberry Pi + Écran)

```
#include <QApplication>
#include "ihmresultatscross.h"
```

Fonctions

— int main (int argc, char ∗argv[ ])

9.21.1 Description détaillée

Programme principal Resultats-Cross (Raspberry Pi + Écran)

Affiche en temps-réel le classement à l'arrivée d'une course

Auteur

TURLIN Suzie suzie.turlin@gmail.com

Version

0.1

9.21.2 Documentation des fonctions

9.21.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

```
00023 {
00024     QApplication a(argc, argv);
00025
00026     IHMResultatsCross w; // instancie la fenêtre principale de l'application
00027     w.show(); // affiche la fenêtre principale de l'application
00028
00029     return a.exec(); // exécute l'application
00030 }
```

9.22 Référence du fichier README.md

9.23 Référence du fichier resultat.cpp

```
#include "resultat.h"
#include "../BaseDeDonnees/basededonnees.h"
#include "coureur.h"
```

9.24 Référence du fichier resultat.h

```
#include <QObject>
```

Classes

— class Resultat

Index