

Chrono-Cross

1.1

Généré par Doxygen 1.7.6.1

Jeudi Juin 7 2018 20 :33 :15

Table des matières

1	Page principale du projet Chrono-Cross	1
1.1	Introduction	1
1.2	Table des matières	1
2	Changelog	2
3	Recette IR	8
4	Base de données MySQL	8
5	A propos	9
6	Licence GPL	9
7	Liste des choses à faire	10
8	Documentation des classes	10
8.1	Référence de la classe BaseDeDonnees	10
8.1.1	Description détaillée	12
8.1.2	Documentation des constructeurs et destructeur	13
8.1.3	Documentation des fonctions membres	13
8.1.4	Documentation des données membres	20
8.2	Référence de la classe ChronoCrossClassement	20
8.2.1	Documentation des constructeurs et destructeur	23
8.2.2	Documentation des fonctions membres	24
8.2.3	Documentation des données membres	36
8.3	Référence de la classe EditionCourse	37
8.3.1	Description détaillée	38
8.3.2	Documentation des constructeurs et destructeur	38
8.3.3	Documentation des fonctions membres	38
8.4	Référence de la classe EditionManifestation	39
8.4.1	Description détaillée	40
8.4.2	Documentation des constructeurs et destructeur	40
8.4.3	Documentation des fonctions membres	40
8.5	Référence de la classe IHMManifestation	41

8.5.1	Description détaillée	45
8.5.2	Documentation des constructeurs et destructeur	45
8.5.3	Documentation des fonctions membres	47
8.5.4	Documentation des données membres	70
8.6	Référence de la classe IHMResultats	72
8.6.1	Description détaillée	75
8.6.2	Documentation des constructeurs et destructeur	75
8.6.3	Documentation des fonctions membres	75
8.6.4	Documentation des données membres	81
9	Documentation des fichiers	81
9.1	Référence du fichier basededonnees.cpp	81
9.2	Référence du fichier basededonnees.cpp	82
9.3	Référence du fichier basededonnees.cpp	82
9.4	Référence du fichier basededonnees.h	82
9.4.1	Documentation des macros	82
9.5	Référence du fichier basededonnees.h	82
9.5.1	Documentation des macros	83
9.6	Référence du fichier basededonnees.h	83
9.6.1	Documentation des macros	83
9.7	Référence du fichier Changelog.dox	84
9.8	Référence du fichier chronocrossclassement.cpp	84
9.8.1	Description détaillée	84
9.9	Référence du fichier chronocrossclassement.h	84
9.9.1	Description détaillée	85
9.9.2	Documentation des macros	85
9.10	Référence du fichier editioncourse.cpp	86
9.10.1	Description détaillée	86
9.11	Référence du fichier editioncourse.h	86
9.11.1	Description détaillée	86
9.12	Référence du fichier editionmanifestation.cpp	86
9.12.1	Description détaillée	87
9.13	Référence du fichier editionmanifestation.h	87
9.13.1	Description détaillée	87

9.14	Référence du fichier ihmmanifestation.cpp	87
9.14.1	Description détaillée	87
9.15	Référence du fichier ihmmanifestation.h	87
9.15.1	Description détaillée	88
9.15.2	Documentation des macros	88
9.16	Référence du fichier ihmresultats.cpp	91
9.16.1	Description détaillée	91
9.17	Référence du fichier ihmresultats.h	91
9.17.1	Description détaillée	92
9.17.2	Documentation des macros	92
9.18	Référence du fichier main.cpp	93
9.18.1	Description détaillée	93
9.18.2	Documentation des fonctions	93
9.19	Référence du fichier main.cpp	94
9.19.1	Description détaillée	94
9.19.2	Documentation des fonctions	94
9.20	Référence du fichier main.cpp	94
9.20.1	Description détaillée	95
9.20.2	Documentation des fonctions	95
9.21	Référence du fichier README.dox	95

1 Page principale du projet Chrono-Cross

1.1 Introduction

Système informatisé de gestion de bout en bout de courses à pied.

1.2 Table des matières

- [Changelog](#)
- [Recette IR](#)
- [Base de données MySQL](#)
- [A propos](#)
- [Licence GPL](#)

Dépôt SVN : <https://svn.riouxsvn.com/chrono-cross>

2 Changelog

r64 | jbulin | 2018-06-07 18 :22 :23 +0200 (jeu. 07 juin 2018) | 1 ligne

Release 1.1

r63 | jbulin | 2018-06-07 18 :12 :01 +0200 (jeu. 07 juin 2018) | 1 ligne

Modification de la version (passage en version 1.1)

r62 | cpellizzoni | 2018-06-07 16 :05 :22 +0200 (jeu. 07 juin 2018) | 1 ligne

ajustement IHM

r61 | cpellizzoni | 2018-06-05 14 :06 :45 +0200 (mar. 05 juin 2018) | 2 lignes

Modification version Windows

r60 | cpellizzoni | 2018-06-05 11 :01 :41 +0200 (mar. 05 juin 2018) | 1 ligne

Portage sous Windows et Qt 5.4

r59 | jbulin | 2018-05-30 14 :55 :31 +0200 (mer. 30 mai 2018) | 1 ligne

Ajustement des couleurs, le fond est devenu blanc pour la visibilité de loin sur grand écran. Suppression du layout 'onglet' pour rendre l'application plus épurée.

r58 | cpellizzoni | 2018-05-30 12 :12 :33 +0200 (mer. 30 mai 2018) | 1 ligne

bouton top depart et enregistrer fonctionnelle

r57 | jbulin | 2018-05-30 11 :28 :47 +0200 (mer. 30 mai 2018) | 1 ligne

Changement des couleurs et la police du tableau pour améliorer la visibilité sur grand écran

r56 | jbulin | 2018-05-25 16 :45 :30 +0200 (ven. 25 mai 2018) | 1 ligne

Tag de la version 1.0

r55 | jbulin | 2018-05-25 16 :41 :23 +0200 (ven. 25 mai 2018) | 1 ligne

Pour l'application Gestion-Cross ajout de doxygene. Pour l'application Resultats-Cross ajout de la méthode rafraichir(), ajout de doxygènes.

r54 | cpellizzoni | 2018-05-25 14 :56 :24 +0200 (ven. 25 mai 2018) | 1 ligne

ajout doxygen Gestion-Cross et Crono-Cross + ajout fonctionnalite Chrono-Cross et modification interface

r53 | cpellizzoni | 2018-05-24 10 :37 :34 +0200 (jeu. 24 mai 2018) | 1 ligne

ajout tableau pour visualiser classement temps et dossard

r52 | jbulin | 2018-05-23 17 :37 :32 +0200 (mer. 23 mai 2018) | 1 ligne

ajout de la fonction activerImpression et ajouterResultat ainsi que la doxygene relative. Correction d'une faute d'orthographe sur la fonction imprimerResultats. Suppression des classes inutile (editionManifestation, editionCourse, [IHMManifestation](#)).

r51 | cpellizzoni | 2018-05-23 17 :30 :03 +0200 (mer. 23 mai 2018) | 1 ligne

ajout programmation de l'interface (bouton) + decodage de trames

r50 | tvaira | 2018-04-20 11 :31 :56 +0200 (ven. 20 avril 2018) | 1 ligne

Amelioration IHM inscription coureurs

r49 | jbulin | 2018-04-19 21 :49 :55 +0200 (jeu. 19 avril 2018) | 1 ligne

Suppression de qDebug sous commentaires, ajout de commentaires Doxygene

r48 | cpellizzoni | 2018-04-19 17 :49 :29 +0200 (jeu. 19 avril 2018) | 1 ligne

ajout fonctionnalite selection course pour inscription avec dossard , bouton modifier un coureur fonctionnel

r47 | jbulin | 2018-04-18 17 :59 :15 +0200 (mer. 18 avril 2018) | 1 ligne

Modifications apportées sur le tableau résultats

r46 | cpellizzoni | 2018-04-18 16 :24 :57 +0200 (mer. 18 avril 2018) | 1 ligne

modification bouton création coureur et ajout de la suppression et ajout selection coureur pour modifier et supprimer

r45 | jbulin | 2018-04-18 15 :28 :44 +0200 (mer. 18 avril 2018) | 1 ligne

Programmation du bouton 'Imprimer', ajout de l'impression par défaut en format paysage, ajout de la colonne 'Numéro de Dossard' dans le tableau de résultatset ajout des

informations sur la manifestation et la course pour l'impression.

r44 | tvaira | 2018-04-16 11 :09 :28 +0200 (lun. 16 avril 2018) | 1 ligne

Verification des TODO

r43 | jbulin | 2018-04-13 19 :09 :36 +0200 (ven. 13 avril 2018) | 1 ligne

Initialisation Resultats-Cross avec le contenu de Gestion-Cross

r42 | jbulin | 2018-04-12 22 :13 :20 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout des couleurs et des classements en gras dans le tableau. Programmation pour l'actualisation des labels, ajout d'un bouton imprimer. Correction d'un bug sur la suppression d'une course

r41 | cpellizzoni | 2018-04-12 17 :29 :45 +0200 (jeu. 12 avril 2018) | 1 ligne

Modification IHM

r40 | cpellizzoni | 2018-04-12 12 :14 :52 +0200 (jeu. 12 avril 2018) | 1 ligne

Verification INE fonctionnel + suppression coureur fonctionnel

r39 | jbulin | 2018-04-12 12 :02 :30 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout et programmation des listes déroulantes dans l'onglet Resultats

r38 | jbulin | 2018-04-11 13 :24 :22 +0200 (mer. 11 avril 2018) | 1 ligne

Ajout de la fonction afficher les résultats, et ajout du tableau des classements

r37 | cpellizzoni | 2018-04-11 12 :13 :54 +0200 (mer. 11 avril 2018) | 1 ligne

Ajout d'une verification pour la suppression d'un coureur

r36 | cpellizzoni | 2018-04-11 12 :12 :09 +0200 (mer. 11 avril 2018) | 1 ligne

Bouton supprimer coureur fonctionnel

r35 | jbulin | 2018-04-06 12 :31 :57 +0200 (ven. 06 avril 2018) | 1 ligne

Ajout de doxygen

r34 | jbulin | 2018-04-06 11 :22 :09 +0200 (ven. 06 avril 2018) | 1 ligne

Ajout des des classes [EditionManifestation](#) et [EditionCourse](#), programmation des boutons pour l'edition de la course

r33 | jbulin | 2018-04-06 11 :19 :57 +0200 (ven. 06 avril 2018) | 1 ligne

rajout doxygen modifierManifestation

r32 | jbulin | 2018-04-05 17 :56 :14 +0200 (jeu. 05 avril 2018) | 1 ligne

Programmation des boutons modifier, supprimer, et inscrire de l'onglet manifestation.
Programmation de ligne de scroll.

r31 | cpellizzoni | 2018-04-05 12 :29 :56 +0200 (jeu. 05 avril 2018) | 1 ligne

Doxygen mis a jour, bouton et code pour creer un coureur fonctionnelle

r30 | cpellizzoni | 2018-04-04 14 :38 :44 +0200 (mer. 04 avril 2018) | 1 ligne

liste déroulante catégorie fonctionnelle

r29 | jbulin | 2018-04-04 14 :34 :59 +0200 (mer. 04 avril 2018) | 1 ligne

Programmation du bouton supprimer manifestation, et la programmation du bouton créer course

r28 | tvaira | 2018-03-30 19 :43 :07 +0200 (ven. 30 mars 2018) | 1 ligne

Restructuration en onglets de l'IHM principale

r27 | jbulin | 2018-03-30 12 :30 :16 +0200 (ven. 30 mars 2018) | 1 ligne

Ajout de la classe basededonnees

r26 | jbulin | 2018-03-30 11 :56 :06 +0200 (ven. 30 mars 2018) | 1 ligne

Ajout de la classe ServeurBDDGestionCross, et classe [IHMManifestation](#) renommé en IHMGestionCross

r25 | cpellizzoni | 2018-03-29 13 :52 :34 +0200 (jeu. 29 mars 2018) | 1 ligne

Modification IHM Resultat-Cross

r24 | cpellizzoni | 2018-03-29 11 :08 :43 +0200 (jeu. 29 mars 2018) | 1 ligne

Modification dossier base de données

r23 | cpellizzoni | 2018-03-29 08 :15 :10 +0200 (jeu. 29 mars 2018) | 1 ligne

Ajout de la base de donnees

r22 | tvaira | 2018-03-28 19 :15 :12 +0200 (mer. 28 mars 2018) | 1 ligne

Ajout du fichier SQL

r21 | jbulin | 2018-03-28 18 :01 :06 +0200 (mer. 28 mars 2018) | 1 ligne

Modifications apportées à l'IHM Gestrion-Cross Julien

r20 | jbulin | 2018-03-28 15 :10 :54 +0200 (mer. 28 mars 2018) | 1 ligne

Renommage projets Resultats-Cross et Chrono-Cross

r19 | jbulin | 2018-03-28 14 :57 :49 +0200 (mer. 28 mars 2018) | 1 ligne

Renommage projet Gestion-Cross

r18 | cpellizzoni | 2018-03-28 14 :49 :30 +0200 (mer. 28 mars 2018) | 1 ligne

Suppression fichiers inutiles

r17 | jbulin | 2018-03-28 14 :46 :40 +0200 (mer. 28 mars 2018) | 1 ligne

Modifications apporté a creation-evenement

r16 | cpellizzoni | 2018-03-28 14 :17 :25 +0200 (mer. 28 mars 2018) | 1 ligne

Ajout classe qextserialport

r15 | cpellizzoni | 2018-03-28 14 :12 :48 +0200 (mer. 28 mars 2018) | 1 ligne

Suppression fichiers inutiles

r14 | cpellizzoni | 2018-03-28 14 :12 :10 +0200 (mer. 28 mars 2018) | 1 ligne

Modification et ajout IHM corentin

r13 | tvaira | 2018-03-24 11 :48 :25 +0100 (sam. 24 mars 2018) | 1 ligne

Mise a jour parametrage Doxygen

r12 | tvaira | 2018-03-24 11 :44 :05 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout parametrage Doxygen

r11 | tvaira | 2018-03-24 11 :23 :34 +0100 (sam. 24 mars 2018) | 1 ligne

Nettoyage des fichiers indesirables

r10 | jbulin | 2018-03-21 17 :52 :15 +0100 (mer. 21 mars 2018) | 1 ligne

Ajout de la vérification de la saisie dans le LineEdit dateManifestation

r9 | jbulin | 2018-03-19 10 :21 :53 +0100 (lun. 19 mars 2018) | 1 ligne

Programmation des slots et bouton pour la gestion de la manifestation -Julien

r8 | jbulin | 2018-03-16 12 :29 :04 +0100 (ven. 16 mars 2018) | 1 ligne

Programmation des boutons julien

r7 | jbulin | 2018-03-15 16 :55 :09 +0100 (jeu. 15 mars 2018) | 1 ligne

Modification de l'IHMRaspberryPi

r6 | cpellizzoni | 2018-03-15 16 :48 :55 +0100 (jeu. 15 mars 2018) | 1 ligne

Programme de test panneau lumineux

r5 | cpellizzoni | 2018-03-15 11 :54 :53 +0100 (jeu. 15 mars 2018) | 1 ligne

Modification interface graphique

r4 | jbulin | 2018-03-15 11 :15 :14 +0100 (jeu. 15 mars 2018) | 1 ligne

Ajout des codes sources julien

r3 | cpellizzoni | 2018-03-14 11 :48 :56 +0100 (mer. 14 mars 2018) | 1 ligne

Ajout code

r2 | tvaira | 2018-02-03 11 :10 :48 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout initial (tv)

r1 | www-data | 2018-02-03 11 :01 :27 +0100 (sam. 03 févr. 2018) | 1 ligne

Creating initial repository structure

3 Recette IR

Étudiant 3 : BULIN Julien

- 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 : PELLIZZONI Corentin

- 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 stockés dans la base de données
- le démarrage d'une course est possible

4 Base de données MySQL

```
CREATE DATABASE IF NOT EXISTS 'Chrono-cross';
```

```
USE 'Chrono-cross';
```

```
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;
```

```
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;
```

```
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;
```

```
CREATE TABLE IF NOT EXISTS 'Manifestation' ( 'idManifestation' int(11) NOT NULL AUTO_INCREMENT, 'Nom' varchar(255) NOT NULL, 'Date' date NOT NULL, PRIMA-
```

```
RY KEY ('idManifestation') ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

```
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, PRIMARY KEY ('idCourse'), CONSTRAINT Course_fk_1 FOREIGN KEY ('idManifestation') REFERENCES Manifestation('idManifestation') ON DELETE CASCADE ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

```
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_Coureur-Course 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 ;
```

```
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 ;
```

5 A propos

Auteur

Bulin Julien <julien.bulin@hotmail.com>

Pellizzoni Corentin <pellizzoni.corentin@yahoo.com>

Version

1.1

Date

2018

6 Licence GPL

This program is free software ; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation ; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY ; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

7 Liste des choses à faire

Membre **ChronoCrossClassement** : **:numeroterDossard** (int l, int c)

vérifier si le numéro de dossard est valide

Membre **IHMManifestation** : **:creerCourse** ()

vérifier si la validité de l'heure

Membre **IHMManifestation** : **:creerManifestation** ()

vérifier si la date n'est pas passée

Membre **IHMManifestation** : **:modifierCourse** ()

vérifier si la validité de l'heure

Membre **IHMManifestation** : **:modifierManifestation** ()

vérifier si la date n'est pas passée

Membre **IHMManifestation** : **:verifierInformationsCoureur** ()

vérifier si la date de naissance n'est pas dans le futur

Membre **IHMResultats** : **:afficherCourse** ()

afficher les paramètres de la course

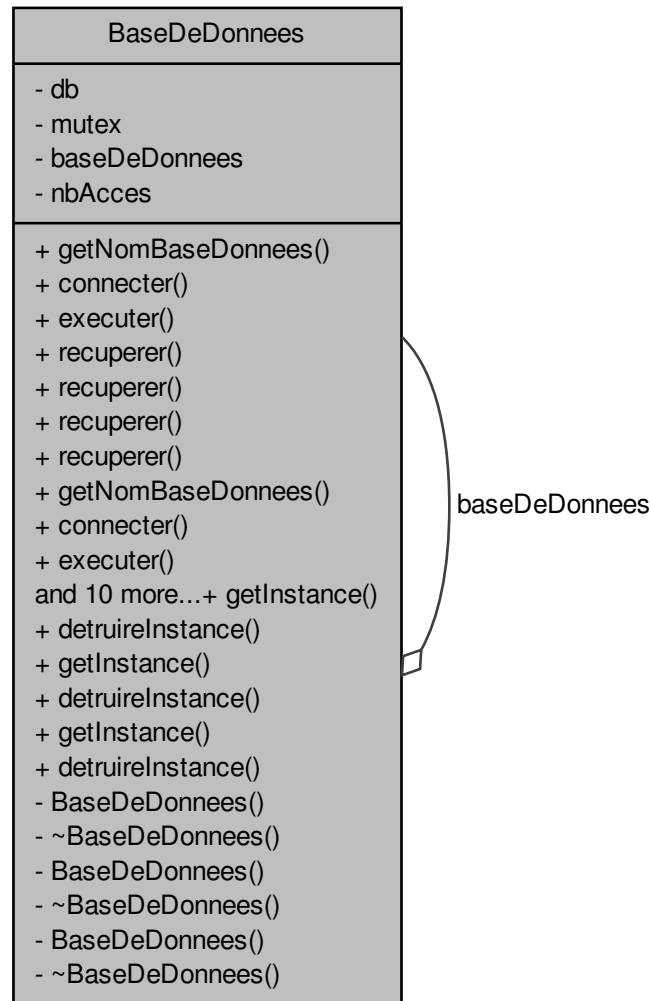
8 Documentation des classes

8.1 Référence de la classe BaseDeDonnees

Déclaration de la classe utilisant la base de données.

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

Graphe de collaboration de BaseDeDonnees :



Fonctions membres publiques

- QString **getNomBaseDonnees** () const
- bool **connecter** (QString serveur=HOSTNAME, QString nomBase=DATABASENAME)
- 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)
- QString [getNomBaseDonnees](#) () const
- bool [connecter](#) (QString serveur=[HOSTNAME](#), QString nomBase=[DATABASENAME](#))
- 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)
- QString [getNomBaseDonnees](#) () const
- bool [connecter](#) (QString serveur=[HOSTNAME](#), QString nomBase=[DATABASENAME](#))
- 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](#) ()
- static void [destruireInstance](#) ()
- static [BaseDeDonnees](#) * [getInstance](#) ()
- static void [destruireInstance](#) ()
- static [BaseDeDonnees](#) * [getInstance](#) ()
- static void [destruireInstance](#) ()

Fonctions membres privées

- [BaseDeDonnees](#) ()
- [~BaseDeDonnees](#) ()
- [BaseDeDonnees](#) ()
- [~BaseDeDonnees](#) ()
- [BaseDeDonnees](#) ()
- [~BaseDeDonnees](#) ()

Attributs privés

- QSqlDatabase [db](#)
- QMutex [mutex](#)

Attributs privés statiques

- static [BaseDeDonnees](#) * [baseDeDonnees](#) = NULL
- static int [nbAcces](#) = 0

8.1.1 Description détaillée

Auteur

Thierry VAIRA

Version

1.0

Date

Mercredi 30 mars 2018

8.1.2 Documentation des constructeurs et destructeur

8.1.2.1 BaseDeDonnees : :BaseDeDonnees () [private]

Références [db](#).

Référencé par [getInstance\(\)](#).

```
{
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::BaseDeDonnees()>";
    #endif
    db = QSqlDatabase::addDatabase("QMYSQL");
}
```

8.1.2.2 BaseDeDonnees : :~BaseDeDonnees () [private]

```
{
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::~~BaseDeDonnees()>";
    #endif
}
```

8.1.2.3 BaseDeDonnees : :BaseDeDonnees () [private]

8.1.2.4 BaseDeDonnees : :~BaseDeDonnees () [private]

8.1.2.5 BaseDeDonnees : :BaseDeDonnees () [private]

8.1.2.6 BaseDeDonnees : :~BaseDeDonnees () [private]

8.1.3 Documentation des fonctions membres

8.1.3.1 bool BaseDeDonnees : :connecter (QString serveur = HOSTNAME, QString nomBase = DATABASENAME)

8.1.3.2 bool BaseDeDonnees : :connecter (QString serveur = HOSTNAME, QString nomBase = DATABASENAME)

Références [db](#), [mutex](#), [PASSWORD](#), et [USERNAME](#).

Référencé par [IHMManifestation : :IHMManifestation\(\)](#), [IHMRésultats : :IHMRésultats\(\)](#), et [ChronoCrossClassement : :initialiserBD\(\)](#).

```
{
    QMutexLocker verrou(&mutex);
    if (!db.isOpen())
    {
        db.setHostName(serveur);
        db.setUserName(USERNAME);
        db.setPassword(PASSWORD);
        db.setDatabaseName(nomBase);
    }
}
```

```

#ifdef DEBUG_BASEDEDONNEES
qDebug() << "HostName : " << db.hostName();
qDebug() << "UserName : " << db.userName();
qDebug() << "DatabaseName : " << db.databaseName();
#endif
if(db.open())
{
#ifdef DEBUG_BASEDEDONNEES
qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()>
connexion réussie à %1").arg(db.hostName());
#endif

return true;
}
else
{
qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()> erreur :
impossible de se connecter à la base de données !");

QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
QString::fromUtf8("Impossible de se connecter à la base de données !"));

return false;
}
}
else
return true;
}

```

8.1.3.3 `bool BaseDeDonnees : :connecter (QString serveur = HOSTNAME, QString nomBase = DATABASENAME)`

8.1.3.4 `static void BaseDeDonnees : :destruireInstance () [static]`

8.1.3.5 `static void BaseDeDonnees : :destruireInstance () [static]`

8.1.3.6 `void BaseDeDonnees : :destruireInstance () [static]`

Références [baseDeDonnees](#), et [nbAcces](#).

Référéncé par [ChronoCrossClassement : :~ChronoCrossClassement\(\)](#), [IHM-Manifestation : :~IHMManifestation\(\)](#), et [IHMResultats : :~IHMResultats\(\)](#).

```

{
// instance ?
if(baseDeDonnees != NULL)
{
nbAcces--;
#ifdef DEBUG_BASEDEDONNEES
qDebug() << "<BaseDeDonnees::destruireInstance()> nbAcces restants = " <
< nbAcces;
#endif
// dernier ?
if(nbAcces == 0)
delete baseDeDonnees;
}
}

```

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

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

Références [db](#), et [mutex](#).

Référencé par [IHManifestation : :creerCoureur\(\)](#), [IHManifestation : :creerCourse\(\)](#), [IHManifestation : :creerManifestation\(\)](#), [ChronoCrossClassement : :enregistrer-Resultat\(\)](#), [IHManifestation : :modifierCoureur\(\)](#), [IHManifestation : :modifier-Course\(\)](#), [IHManifestation : :modifierManifestation\(\)](#), [IHManifestation : :supprimer-Coureur\(\)](#), [IHManifestation : :supprimerCourse\(\)](#), et [IHManifestation : :supprimer-Manifestation\(\)](#).

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::executer()> retour %1
pour la requete : %2").arg(QString::number(retour)).arg(requete);
        #endif
        if(retour)
        {
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::executer()> erreur :
%1 pour la requête %2").arg(r.lastError().text()).arg(requete);

            return false;
        }
    }
    else
        return false;
}
```

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

8.1.3.10 `static BaseDeDonnees* BaseDeDonnees : :getInstance () [static]`

8.1.3.11 `BaseDeDonnees * BaseDeDonnees : :getInstance () [static]`

Références [BaseDeDonnees\(\)](#), [baseDeDonnees](#), et [nbAcces](#).

Référencé par [IHManifestation : :IHManifestation\(\)](#), [IHMRésultats : :IHMRésultats\(\)](#), et [ChronoCrossClassement : :initialiserBD\(\)](#).

```
{
    if(baseDeDonnees == NULL)
        baseDeDonnees = new BaseDeDonnees();

    nbAcces++;
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::getInstance()> nbAcces = " << nbAcces;
    #endif

    return baseDeDonnees;
}
```

8.1.3.12 `static BaseDeDonnees* BaseDeDonnees : :getInstance () [static]`

8.1.3.13 `QString BaseDeDonnees : :getNomBaseDonnees () const`

8.1.3.14 QString BaseDeDonnees : :getNomBaseDonnees () const

Références [db](#).

Référencé par [IHManifestation : :IHManifestation\(\)](#), et [IHMRésultats : :IHMRésultats\(\)](#).

```
{
    return db.databaseName();
}
```

8.1.3.15 QString BaseDeDonnees : :getNomBaseDonnees () const

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

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

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

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

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

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

Références [db](#), et [mutex](#).

Référencé par [IHMRésultats : :afficherManifestation\(\)](#), [IHMRésultats : :afficherRésultats\(\)](#), [IHManifestation : :afficherRésultats\(\)](#), [ChronoCrossClassement : :chargerInscritsCourse\(\)](#), [IHManifestation : :chargerListeCategories\(\)](#), [IHManifestation : :chargerListeClasses\(\)](#), [IHManifestation : :chargerListeCoureurs\(\)](#), [ChronoCrossClassement : :chargerListeCourses\(\)](#), [IHManifestation : :chargerListeCourses\(\)](#), [IHManifestation : :chargerListeInscriptionCourses\(\)](#), [ChronoCrossClassement : :chargerListeManifestation\(\)](#), [IHManifestation : :chargerListeManifestations\(\)](#), [IHManifestation : :getNumeroDossard\(\)](#), et [IHMRésultats : :recupererCourse\(\)](#).

```
{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
#ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QString)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(requete);
#endif
        if(retour)
        {
            // on se positionne sur l'enregistrement
            r.first();

            // on vérifie l'état de l'enregistrement retourné
            if(!r.isValid())
```

```

    {
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("
<BaseDeDonnees::recuperer(QString, QString)> résultat non valide !");
        #endif
        return false;
    }

    // on récupère sous forme de QString la valeur du champ
    if(r.isNull(0))
    {
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("
<BaseDeDonnees::recuperer(QString, QString)> résultat vide !");
        #endif
        return false;
    }
    donnees = r.value(0).toString();
    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << "<BaseDeDonnees::recuperer(QString, QString)>
enregistrement -> " << donnees;
    #endif
    return true;
}
else
{
    qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QString)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg(requete)
;

    return false;
}
}
else
    return false;
}

```

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

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

Références [db](#), et [mutex](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QStringList)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(
requete);
        #endif
        if(retour)
        {
            // on se positionne sur l'enregistrement
            r.first();

            // on vérifie l'état de l'enregistrement retourné
            if(!r.isValid())
            {
                #ifdef DEBUG_BASEDEDONNEES
                qDebug() << QString::fromUtf8("
<BaseDeDonnees::recuperer(QString, QStringList)> résultat non valide !");
                #endif
                return false;
            }
        }
    }
}

```

```

        // on récupère sous forme de QString la valeur de tous les champs
sélectionnés
        // et on les stocke dans une liste de QString
        for(int i=0;i<r.record().count();i++)
            if(!r.isNull(i))
                donnees << r.value(i).toString();
#ifdef DEBUG_BASEDEDONNEES
        qDebug() << "<BaseDeDonnees::recuperer(QString, QStringList)>
enregistrement -> " << donnees;
#endif
        return true;
    }
    else
    {
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QStringList)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg(
requete);
        return false;
    }
}
else
    return false;
}

```

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

Références [db](#), et [mutex](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;
    QString data;

    if(db.isOpen())
    {
        retour = r.exec(requete);
#ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
QVector<QString>)> retour %1 pour la requete : %2").arg(QString::number(retour)).arg(
requete);
#endif
        if(retour)
        {
            // pour chaque enregistrement
            while ( r.next() )
            {
                // on récupère sous forme de QString la valeur du champs
sélectionné
                data = r.value(0).toString();

                #ifdef DEBUG_BASEDEDONNEES
                //qDebug() << "<BaseDeDonnees::recuperer(QString,
QVector<QString>)> enregistrement -> " << data;
                #endif

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);
            }
#ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString, QVector<QString>)>
enregistrement -> " << donnees;
#endif
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,

```

```

        QVector<QString>> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg
        (requete);

        return false;
    }
}
else
    return false;
}

```

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

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

Références [db](#), et [mutex](#).

```

{
    QMutexLocker verrou(&mutex);
    QSqlQuery r;
    bool retour;
    QStringList data;

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
        QVector<QStringList>> retour %1 pour la requete : %2").arg(QString::number(retour)).
        arg(requete);
        #endif
        if(retour)
        {
            // pour chaque enregistrement
            while ( r.next() )
            {
                // on récupère sous forme de QString la valeur de tous les
                champs sélectionnés
                // et on les stocke dans une liste de QString
                for(int i=0;i<r.record().count();i++)
                    data << r.value(i).toString();

                #ifdef DEBUG_BASEDEDONNEES
                //qDebug() << "<BaseDeDonnees::recuperer(QString,
                QVector<QStringList>> enregistrement -> " << data;
                /*for(int i=0;i<r.record().count();i++)
                    qDebug() << r.value(i).toString();*/
                #endif

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);

                // on efface la liste de QString pour le prochain
                enregistrement
                data.clear();
            }
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString,
            QVector<QStringList>> enregistrement -> " << donnees;
            #endif
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
            QVector<QStringList>> erreur : %1 pour la requête %2").arg(r.lastError().text())
            .arg(requete);
        }
    }
}

```

```
        return false;
    }
}
else
    return false;
}
```

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

8.1.4 Documentation des données membres

8.1.4.1 `static BaseDeDonnees * BaseDeDonnees : :baseDeDonnees = NULL`
[static, private]

Référencé par [destruireInstance\(\)](#), et [getInstance\(\)](#).

8.1.4.2 `QSqlDatabase BaseDeDonnees : :db` [private]

Référencé par [BaseDeDonnees\(\)](#), [connecter\(\)](#), [executer\(\)](#), [getNomBaseDonnees\(\)](#), et [recuperer\(\)](#).

8.1.4.3 `QMutex BaseDeDonnees : :mutex` [private]

Référencé par [connecter\(\)](#), [executer\(\)](#), et [recuperer\(\)](#).

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

Référencé par [destruireInstance\(\)](#), et [getInstance\(\)](#).

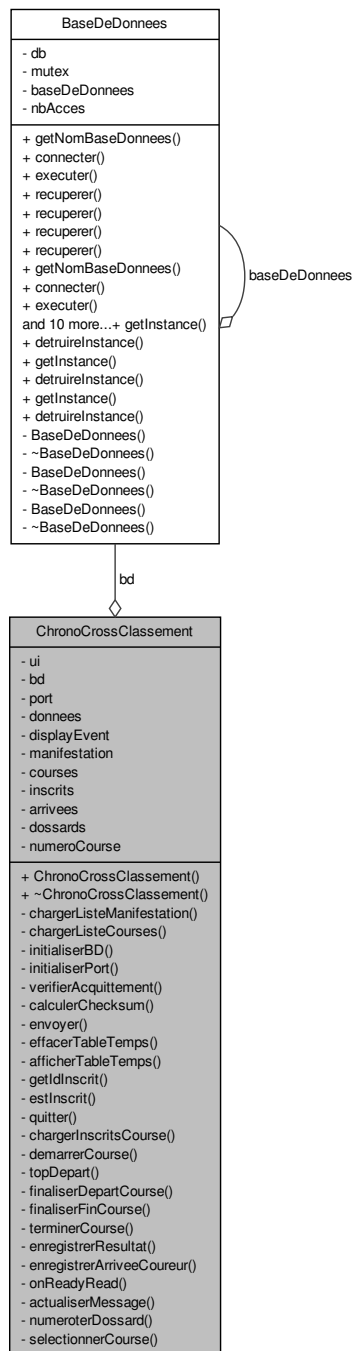
La documentation de cette classe a été générée à partir des fichiers suivants :

- [Chrono-Cross/basededonnees.h](#)
- [Gestion-Cross/basededonnees.h](#)
- [Resultats-Cross/basededonnees.h](#)
- [Chrono-Cross/basededonnees.cpp](#)
- [Gestion-Cross/basededonnees.cpp](#)
- [Resultats-Cross/basededonnees.cpp](#)

8.2 Référence de la classe ChronoCrossClassement

```
#include <chronocrossclassement.h>
```

Grphe de collaboration de ChronoCrossClassement :



Signaux

- void `courseTerminee` (QString numero)
- void `courseDemarree` (QString numero)
- void `tempsArrivee` (QString ordre, QString temps)

Fonctions membres publiques

- `ChronoCrossClassement` (QWidget *parent=0)
Constructeur.
- `~ChronoCrossClassement` ()
Destructeur.

Connecteurs privés

- void `quitter` ()
Ferme l'application principale.
- void `chargerInscritsCourse` ()
Charge la liste des inscrits pour la course sélectionnée.
- void `demarrerCourse` ()
Démarre une course.
- void `topDepart` ()
Démarre une course logiciellement.
- void `finaliserDepartCourse` (QString numero)
Affiche le nombre d'arrivées et de classés.
- void `finaliserFinCourse` (QString numero)
Termine la course.
- void `terminerCourse` ()
Termine une course.
- void `enregistrerResultat` ()
Enregistre les resultats dans la base de donnees.
- void `enregistrerArriveeCoureur` (QString ordre, QString temps)
Enregistre l'arrivée d'un coureur.
- void `onReadyRead` ()
Décode les trames reçus.
- void `actualiserMessage` (QString message)
- void `numéroterDossard` (int l, int c)
Classe les coureurs.
- void `selectionnerCourse` ()
Sélectionne une course à éditer.

Fonctions membres privées

- void `chargerListeManifestation` ()
Charge la liste des manifestations.
- void `chargerListeCourses` ()
Charge la liste des courses.
- void `initialiserBD` ()
Initialise la base de donnees.
- void `initialiserPort` ()
Initialise le port.
- char `verifierAcquittement` (QString trame)
Vérifie l'acquittement de la trame.
- unsigned short `calculerChecksum` (QString data)
Calcule le checksum.
- void `envoyer` (QString &trame)

- *Envoie une trame.*
- void `effacerTableTemps()`
Efface les temps dans le tableau.
- void `afficherTableTemps()`
Affiche les temps dans le tableau.
- QString `getIdInscrit()` (QString dossard)
Obtiens l'idInscrit.
- bool `estInscrit()` (QString dossard)
Vérifie que le dossard est inscrit à la course.

Attributs privés

- Ui : :ChronoCrossClassement * `ui`
relation vers la classe IHM
- `BaseDeDonnees` * `bd`
association vers la classe BaseDeDonnees
- QTextSerialPort * `port`
connexion au port
- QByteArray `donnees`
les données correspondant au trame
- bool `displayEvent`
- QStringList `manifestation`
la manifestation à la date actuelle
- QVector< QStringList > `courses`
la liste des courses pour une manifestation
- QVector< QStringList > `inscrits`
la liste des inscrits pour la course
- QVector< QString > `arrivees`
recupere le temps et affiche le nombre d'arrivée
- QVector< QString > `dossards`
recupere les dossards
- QString `numeroCourse`
recupere le numero de la course

8.2.1 Documentation des constructeurs et destructeur

8.2.1.1 **ChronoCrossClassement : :ChronoCrossClassement (QWidget * parent = 0)** [explicit]

Paramètres

<i>parent</i>	
---------------	--

Références `chargerListeCourses()`, `chargerListeManifestation()`, `demarrerCourse()`, `enregistrerResultat()`, `initialiserBD()`, `initialiserPort()`, `numeroCourse`, `port`, `selectionnerCourse()`, `terminerCourse()`, `topDepart()`, et `ui`.

```

:
QMainWindow(parent),
ui(new Ui::ChronoCrossClassement)
{
    ui->setupUi(this);
    ui->comboBoxCourses->setEnabled(false);
    ui->pushButtonDemarrageCourse->setEnabled(false);
    ui->pushButtonTopDepart->setEnabled(false);
    ui->pushButtonFinCourse->setEnabled(false);
    ui->pushButtonEnregistrementResultats->setEnabled(false);
    QStringList header; // nom des colonnes

```

```

header << "Classement" << "Dossard" << "Temps";

// On fixe le nombre de colonnes
ui->tableTemps->setColumnCount(header.size());
// On applique les noms des colonnes
ui->tableTemps->setHorizontalHeaderLabels(header);

// on cache les numéros de ligne
ui->tableTemps->verticalHeader()->setHidden(true);

QHeaderView * headerView = ui->tableTemps->horizontalHeader();
// on redimensionne automatiquement la colonne pour occuper l'espace
disponible
#ifdef QT_VERSION >= 0x050000
headerView->setSectionResizeMode(QHeaderView::Stretch);
#else
headerView->setResizeMode(QHeaderView::Stretch);
#endif

connect(ui->comboBoxCourses, SIGNAL(currentIndexChanged(int)), this, SLOT(
    selectionnerCourse()));
connect(ui->pushButtonDemarrageCourse, SIGNAL(clicked()), this, SLOT(
    demarrerCourse()));
connect(ui->pushButtonTopDepart, SIGNAL(clicked()), this, SLOT(topDepart()));
connect(ui->pushButtonEnregistrementResultats, SIGNAL(clicked()), this, SLOT(
    enregistrerResultat()));
connect(ui->pushButtonFinCourse, SIGNAL(clicked()), this, SLOT(terminerCourse(
    )));

// Initialisation
initialiserBD();

port = NULL;
initialiserPort();

chargerListeManifestation();
chargerListeCourses();
numeroCourse = "";
}

```

8.2.1.2 ChronoCrossClassement : ~ChronoCrossClassement ()

Références [BaseDeDonnees](#) : [destruireInstance\(\)](#), et [ui](#).

```

{
    BaseDeDonnees::destruireInstance();
    delete ui;
}

```

8.2.2 Documentation des fonctions membres

8.2.2.1 void ChronoCrossClassement : actualiserMessage (QString message) [private, slot]

Référéncé par [envoyer\(\)](#), et [onReadyRead\(\)](#).

```

{
    qDebug() << message;
}

```

8.2.2.2 ChronoCrossClassement : afficherTableTemps () [private]

Affiche les résultats dans le tableau.

Références [arrivees](#), [dossards](#), [effacerTableTemps\(\)](#), [numeroterDossard\(\)](#), et [ui](#).

Référencé par [enregistrerArriveeCoureur\(\)](#).

```
{
    effacerTableTemps();

    QString temps;
    QString dossard;
    QTime tempsCoureur;
    QString nomprenom;

    disconnect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));

    for(int l = 0; l < arrivees.count(); l++)
    {
        temps = arrivees.at(l);
        temps.chop(2); // enleve les 2 zéros inutilisés
        if(temps.count(":") == 2)
            tempsCoureur = QTime::fromString(temps, "h:m:s.zzz");
        else if(temps.count(":") == 1)
            tempsCoureur = QTime::fromString(temps, "m:s.zzz");
        else
            tempsCoureur = QTime::fromString(temps, "s.zzz");

        temps = tempsCoureur.toString("hh:mm:ss");

        dossard = dossards.at(l);
        //qDebug() << Q_FUNC_INFO << l << dossard << temps;
        QTableWidgetItem *elementClassement = new QTableWidgetItem(
            QString::number(l+1));
        QTableWidgetItem *elementDossard = new QTableWidgetItem(dossard);
        QTableWidgetItem *elementTemps = new QTableWidgetItem(temps);
        QTableWidgetItem *elementNomPrenom = new QTableWidgetItem(nomprenom);
        ui->tableTemps->insertRow(l);
        // propriétés
        elementClassement->setFlags(Qt::ItemIsEnabled);
        elementClassement->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        elementTemps->setFlags(Qt::ItemIsEnabled);
        elementTemps->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        elementDossard->setFlags(Qt::ItemIsEnabled | Qt::ItemIsEditable);
        elementDossard->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        elementNomPrenom->setFlags(Qt::ItemIsEnabled);
        elementNomPrenom->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);

        ui->tableTemps->setItem(l, 0, elementClassement); // l : ligne (row) et
        c : colonne
        ui->tableTemps->setItem(l, 1, elementDossard);
        ui->tableTemps->setItem(l, 2, elementTemps);
        ui->tableTemps->setItem(l, 3, elementNomPrenom);
    }

    connect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));
}
```

8.2.2.3 ChronoCrossClassement : :calculerChecksum (QString data) [private]

Calcule le checksum.

Référencé par [envoyer\(\)](#).

```
{
    unsigned short checksum = 0;
    int i = 0;

    if(data.startsWith("#"))
```

```

        i = 1;
        for (; i < data.length(); i++)
            checksum += data[i].toLatin1();

    return checksum;
}

```

8.2.2.4 ChronoCrossClassement : chargerInscritsCourse () [private, slot]

Charge les inscrits à la course.

Références [bd](#), [COURSE_ID](#), [courses](#), [inscrits](#), [BaseDeDonnees : recuperer\(\)](#), et [ui](#).

Référencé par [chargerListeCourses\(\)](#).

```

{
    if (ui->comboBoxCourses->currentIndex() > 0)
    {
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Course sélectionnée : ")
        << courses.at(ui->comboBoxCourses->currentIndex()-1);

        QString requete;
        QStringList course;
        course = courses.at(ui->comboBoxCourses->currentIndex()-1);
        QString idCourseActuelle = course.at(COURSE_ID);
        bool retour;

        inscrits.clear();

        requete = "SELECT idInscrit, NumeroDossard FROM Inscrit WHERE idCourse"
        = " + idCourseActuelle ;
        retour = bd->recuperer(requete, inscrits);

        if (retour != false)
        {
            //qDebug() << Q_FUNC_INFO << inscrits;
            //qDebug() << Q_FUNC_INFO << inscrits.count();
            ui->labelNbCoureurs->setText("Nb coureurs : " + QString::number(
            inscrits.count()));
        }
        else
        {
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Requête incorrecte")
            << requete;
            ui->labelNbCoureurs->setText("Nb coureurs : --");
        }
    }
    else
    {
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Aucune course"
        sélectionnée");
        ui->labelNbCoureurs->setText("Nb coureurs : --");
    }
}

```

8.2.2.5 ChronoCrossClassement : chargerListeCourses () [private]

Références [bd](#), [chargerInscritsCourse\(\)](#), [COURSE_DISTANCE](#), [COURSE_HEURED-EPART](#), [COURSE_NOM](#), [courses](#), [manifestation](#), [BaseDeDonnees : recuperer\(\)](#), et [ui](#).

Référencé par [ChronoCrossClassement\(\)](#).

```

{
    QString requete;

```

```

QStringList course;
bool retour;

if(manifestation.isEmpty())
    return;

requete = "SELECT * FROM Course WHERE idManifestation = " + manifestation.
    at(0) ;
qDebug() << Q_FUNC_INFO << requete;
retour = bd->recuperer(requete, courses);
if(retour != false)
{
    ui->comboBoxCourses->addItem("");
    for(int i=0; i < courses.size(); i++)
    {
        course = courses.at(i);
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Course : ") << course
        ;
        ui->comboBoxCourses->addItem(course.at(COURSE_NOM) + " - " + course
        .at(COURSE_DISTANCE) + " m - " + course.at(COURSE_HEUREDEPART));
    }
}
else
{
    QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
    QString::fromUtf8("Aucune course !"));
}
connect(ui->comboBoxCourses, SIGNAL(currentIndexChanged(int)), this, SLOT(
    chargerInscritsCourse()));
}

```

8.2.2.6 ChronoCrossClassement : :chargerListeManifestation () [private]

Charge la manifestation du jour.

Références `bd`, `manifestation`, `BaseDeDonnees : :recuperer()`, et `ui`.

Référéncé par `ChronoCrossClassement()`.

```

{
    QString requete;
    bool retour;

    // Récupère la manifestation de la journée dans un QStringList
    requete = "SELECT * FROM Manifestation WHERE Manifestation.Date =
        date(now()) LIMIT 1;";
    // Test
    //requete = "SELECT * FROM Manifestation ";
    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->recuperer(requete, manifestation);
    if(retour != false)
    {
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Manifestation : ") <<
        manifestation;
        ui->labelManifestationInformations->setText(manifestation.at(1));
        ui->comboBoxCourses->setEnabled(true);
    }
    else
    {
        ui->comboBoxCourses->setEnabled(false);
        QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
        QString::fromUtf8("Aucune manifestation !"));
    }
}

```

8.2.2.7 void ChronoCrossClassement : :courseDemarree (QString numero)
[signal]

Référencé par [demarrerCourse\(\)](#), [onReadyRead\(\)](#), et [topDepart\(\)](#).

8.2.2.8 void ChronoCrossClassement : :courseTerminee (QString numero)
[signal]

Référencé par [finaliserDepartCourse\(\)](#), et [onReadyRead\(\)](#).

8.2.2.9 ChronoCrossClassement : :demarrerCourse () [private, slot]

Demarre la course.

Références [courseDemarree\(\)](#), [envoyer\(\)](#), [finaliserDepartCourse\(\)](#), [numeroterDossard\(\)](#), et [ui](#).

Référencé par [ChronoCrossClassement\(\)](#).

```
{
    // 0 -> Mode PTB
    // 5 -> Mode CLOCK
    QString * chaine = new QString("#WP 120 5");
    envoyer(*chaine);
    usleep(100000);

    chaine = new QString("#WC 007 02 00:00 00/00/01");
    envoyer(*chaine);

    chaine = new QString("#RP 003");
    envoyer(*chaine);

    connect(this, SIGNAL(courseDemarree(QString)), this, SLOT(
        finaliserDepartCourse(QString)));
    connect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));
}
```

8.2.2.10 ChronoCrossClassement : :effacerTableTemps () [private]

Efface les résultats du tableau.

Références [numeroterDossard\(\)](#), et [ui](#).

Référencé par [afficherTableTemps\(\)](#), et [finaliserDepartCourse\(\)](#).

```
{
    disconnect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));
    int nb = ui->tableTemps->rowCount();
    // on efface les lignes du tableau une par une
    for(int i = 0; i < nb; i++)
    {
        ui->tableTemps->removeRow(0);
    }
    connect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));
}
```

8.2.2.11 ChronoCrossClassement : :enregistrerArriveeCoureur (QString ordre, QString temps) [private, slot]

Enregistre l'arrivée des coureurs.

Références [afficherTableTemps\(\)](#), [arrivees](#), [dossards](#), [numeroCourse](#), et [ui](#).

Référencé par [finaliserDepartCourse\(\)](#).

```
{
    Q_UNUSED(ordre)

    if(!numeroCourse.isEmpty())
    {
        qDebug() << Q_FUNC_INFO << temps;
        arrivees.push_back(temps);
        dossards.push_back("");

        ui->labelNbArrivees->setText(QString::fromUtf8("Nb Arrivées : ") +
        QString::number(arrivees.count()));
        ui->labelClasses->setText(QString::fromUtf8("Nb Classés : ") +
        QString::number(dossards.count() - dossards.count("")));
        afficherTableTemps();
    }
}
```

8.2.2.12 ChronoCrossClassement : :enregistrerResultat () [private, slot]

Références [arrivees](#), [bd](#), [dossards](#), [BaseDeDonnees : :executer\(\)](#), [getIdInscrit\(\)](#), et [numeroCourse](#).

Référencé par [ChronoCrossClassement\(\)](#).

```
{
    if(!numeroCourse.isEmpty())
    {
        QString temps;
        QString dossard;
        QString idInscrit;

        for(int l = 0; l < arrivees.count(); l++)
        {
            temps = arrivees.at(l);
            dossard = dossards.at(l);
            idInscrit = getIdInscrit(dossard);
            if(!idInscrit.isEmpty() && !dossard.isEmpty())
            {
                QString requete = "INSERT INTO Arrivee(idInscrit, Temps)
VALUES ('" + idInscrit + "','" + temps + "')";
                qDebug() << Q_FUNC_INFO << requete;
                bool retour = bd->executer(requete);
            }
        }
    }
}
```

8.2.2.13 ChronoCrossClassement : :envoyer (QString & trame) [private]

Envoie la trame.

Références [actualiserMessage\(\)](#), [calculerChecksum\(\)](#), et [port](#).

Référencé par [demarrerCourse\(\)](#), [terminerCourse\(\)](#), et [topDepart\(\)](#).

```

{
    unsigned short checksum = 0;

    checksum = calculerChecksum(trame);
    trame += QString("\\t%1").arg(checksum, 4, 16, QChar('0')).toUpper();

    qDebug() << Q_FUNC_INFO << trame;
    actualiserMessage(QString::fromUtf8("Trame envoyée : %1").arg(trame));

    trame += "\\r\\n";

    if (port != NULL && port->isOpen())
    {
        port->write(trame.toLatin1());
    }
}

```

8.2.2.14 ChronoCrossClassement : :estInscrit (QString *dossard*) [private]

Vérifie l'inscription à la course.

Références [inscrits](#).

```

{
    for(int i = 0; i < inscrits.count(); i++)
    {
        if(inscrits.at(i).at(1) == dossard)
            return true;
    }
    return false;
}

```

8.2.2.15 ChronoCrossClassement : :finaliserDepartCourse (QString *numero*) [private, slot]

Références [arrivees](#), [courseTerminee\(\)](#), [dossards](#), [effacerTableTemps\(\)](#), [enregistrerArriveeCoureur\(\)](#), [finaliserFinCourse\(\)](#), [numeroCourse](#), [tempsArrivee\(\)](#), et [ui](#).

Référéncé par [demarrerCourse\(\)](#), et [topDepart\(\)](#).

```

{
    if(numeroCourse.isEmpty())
    {
        qDebug() << Q_FUNC_INFO << "numéro course : " << numero;
        numeroCourse = numero;
        arrivees.clear();
        dossards.clear();

        effacerTableTemps();
        ui->labelNbArrivees->setText(QString::fromUtf8("Nb Arrivées : ") +
        QString::number(arrivees.count()));
        ui->labelClasses->setText(QString::fromUtf8("Nb Classés : ") +
        QString::number(dossards.count() - dossards.count("")));
        ui->comboBoxCourses->setEnabled(false);
        ui->pushButtonDemarrageCourse->setEnabled(false);
        ui->pushButtonTopDepart->setEnabled(false);
        ui->pushButtonFinCourse->setEnabled(true);
        ui->pushButtonEnregistrementResultats->setEnabled(true);
        connect(this, SIGNAL(tempsArrivee(QString,QString)), this, SLOT(
        enregistrerArriveeCoureur(QString,QString)));
        connect(this, SIGNAL(courseTerminee(QString)), this, SLOT(
        finaliserFinCourse(QString)));
    }
}

```

8.2.2.16 ChronoCrossClassement : :finaliserFinCourse (QString numero) [private, slot]

Enregistre les données dans la base de données.

Références [arrivees](#), [dossards](#), [numeroCourse](#), [numeroterDossard\(\)](#), et [ui](#).

Référencé par [finaliserDepartCourse\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO << "numéro course : " << numero;
    // réinitialisation pour prochaine course
    numeroCourse = "";
    arrivees.clear();
    dossards.clear();

    ui->comboBoxCourses->setEnabled(true);
    ui->pushButtonDemarrageCourse->setEnabled(true);
    ui->pushButtonTopDepart->setEnabled(true);
    ui->pushButtonFinCourse->setEnabled(false);
    ui->pushButtonEnregistrementResultats->setEnabled(false);
    disconnect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));
}
```

8.2.2.17 ChronoCrossClassement : :getIdInscrit (QString dossard) [private]

Obtient l'idInscrit.

Références [inscrits](#).

Référencé par [enregistrerResultat\(\)](#).

```
{
    for(int i = 0; i < inscrits.count(); i++)
    {
        if(inscrits.at(i).at(1) == dossard)
            return inscrits.at(i).at(0);
    }
    return QString("");
}
```

8.2.2.18 ChronoCrossClassement : :initialiserBD () [private]

Références [bd](#), [BaseDeDonnees : :connecter\(\)](#), et [BaseDeDonnees : :getInstance\(\)](#).

Référencé par [ChronoCrossClassement\(\)](#).

```
{
    // Connexion à la base de données
    bd = BaseDeDonnees::getInstance();
    bool connexion = bd->connecter();
    if(!connexion)
    {
        QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
            QString::fromUtf8("Impossible de se connecter à la base de données !"));
    }
}
```

8.2.2.19 ChronoCrossClassement : :initialiserPort () [private]

Références [displayEvent](#), [NOM_PORT](#), [onReadyRead\(\)](#), et [port](#).

Référencé par [ChronoCrossClassement\(\)](#).

```
{
    if(port != NULL && port->isOpen())
    {
        port->close();
        disconnect(port, SIGNAL(readyRead()), this, SLOT(onReadyRead()));
    }

    port = new QextSerialPort(NOM_PORT, QextSerialPort::EventDriven);
    port->setBaudRate(BAUD9600);
    port->setFlowControl(FLOW_OFF);
    port->setParity(PAR_NONE);
    port->setDataBits(DATA_8);
    port->setStopBits(STOP_1);
    //port->setTimeout(500);
    displayEvent = false;

    if(port->open(QIODevice::ReadWrite) == true)
    {
        connect(port, SIGNAL(readyRead()), this, SLOT(onReadyRead()));
    }
    if(!port->isOpen())
    {
        QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
            QString::fromUtf8("Impossible d'ouvrir le port !"));
    }
}
```

8.2.2.20 ChronoCrossClassement : :numeroterDossard (int l, int c) [private, slot]

A faire vérifier si le numéro de dossard est valide

Références [dossards](#), et [ui](#).

Référencé par [afficherTableTemps\(\)](#), [demarrerCourse\(\)](#), [effacerTableTemps\(\)](#), [finaliserFinCourse\(\)](#), et [topDepart\(\)](#).

```
{
    QTableWidgetItem *elementDossard;

    // récupère l'élément Dossard dans le TableWidget
    elementDossard = ui->tableTemps->item(l, c);

    // remplace le numéro de dossard saisi dans le QVector
    dossards.replace(l, elementDossard->data(0).toString());
    qDebug() << Q_FUNC_INFO << elementDossard->data(0).toString() <<
dossards;

    ui->labelClasses->setText(QString::fromUtf8("Nb Classés : ") +
        QString::number(dossards.count() - dossards.count("")));
}
```

8.2.2.21 ChronoCrossClassement : :onReadyRead () [private, slot]

Références [actualiserMessage\(\)](#), [courseDemarree\(\)](#), [courseTerminee\(\)](#), [donnees](#), [numeroCourse](#), [port](#), [tempsArrivee\(\)](#), et [verifierAcquittement\(\)](#).

Référencé par [initialiserPort\(\)](#).

```
{
    while(port->bytesAvailable())
```

```

{
    donnees += port->readAll();
    usleep(100000); // cf. timeout
}

if(donnees.contains("\r\n"))
{
    QString trameRecue = QString(donnees.data());
    QStringList trames = trameRecue.split("\r\n");
    QStringList champs;
    char etatAcquittement;

    qDebug() << Q_FUNC_INFO << "trame(s) recue(s) =" << trames;

    /* Liste des trames reçues :

    - "AK X" acquittement avec : X = 'C' accepted, 'F' rejected, 'R' not
    supported
    - "TS 00:00:00 00/00/01" synchro
    - "&P 003 XX 90 00:00:00 0" paramètre 003 (demande du numéro de
    course) XX = le numéro de course
    - "TN 1 2      3.71300 365" temps
    - "CL XX" close course XX = le numéro de course terminée

    */

    //qDebug() << Q_FUNC_INFO << trameRecue.split("\r\n");
    for(int i =0;i<trames.size();i++)
    {
        if(!trames.at(i).isEmpty())
        {
            if(trames.at(i).startsWith("AK")) // trame acquittement ?
            {
                etatAcquittement = verifierAcquittement(trames.at(i));
                if(etatAcquittement == 'C')
                {
                    //actualiserMessage(QString::fromUtf8("Mini-Display HL
975 : <font color=\green\ ">ok</font>"));
                    continue;
                }
                if(etatAcquittement == 'R')
                {
                    actualiserMessage(QString::fromUtf8("Mini-Display HL
975 : erreur !"));
                    continue;
                }
            }
            else if(trames.at(i).startsWith("TS")) // trame synchro ?
            {
            }
            else if(trames.at(i).startsWith("&P")) // trame paramètre ?
            {
                // "&P 003 XX YY 00:00:00 0" paramètre 003 (demande du
                numéro de course) XX = le numéro de course
                /*
                X = Run number (00-99). 00 = no run open -> index 2
                Y = Number of the next Run(00-99) -> index 3
                */

                champs = trames.at(i).split(" ", QString::SkipEmptyParts);
                //qDebug() << Q_FUNC_INFO << "numéro course demarree : " <<
                champs.at(3);
                emit courseDemarree(champs.at(3));
            }
            else if(trames.at(i).startsWith("TN")) // trame temps ?
            {
                /*
                Time (TN) : TN SSSS CC HH:MM:SS.FFFFF DDDDD

                S = Sequential number (0 - 9999) -> ordre arrivée
                C = Channel number (1 - 99) in case of manual entry (M1 -
                M4) -> numéro cellule détection (1 = départ et 2 = arrivée)
                H = Hours (0 - 23)

```

```

M = Minutes (0 - 59)
S = Seconds (0 - 59)
F = decimal part (0 - 99999)
D = Days (0 - 32767) counting from 01.01.2000 -> non

utilisé

*/
//      "TN"          1 1          4.30700  365"
// -> "TN", "1", "1", "4.30700", "365"      05FE"
champs = trames.at(i).split(" ", QString::SkipEmptyParts);
//qDebug() << Q_FUNC_INFO << champs;
// temps à l'arrivée ?
if(champs.at(2) == "2")
{
    //qDebug() << Q_FUNC_INFO << champs.at(3);
    emit tempsArrivee(champs.at(1), champs.at(3));
}
}
else if(trames.at(i).startsWith("CL")) // trame course terminée
?
{
    /*
    Closing of a Run (CL) : CL RR
    R = Run number (1 - 99)

    "CL 06 0115"
    */
    champs = trames.at(i).split("\t");
    champs = champs.at(0).split(" ", QString::SkipEmptyParts);
    //qDebug() << Q_FUNC_INFO << "numéro course terminée : " <<
champs.at(1) << numeroCourse;
    if(!numeroCourse.isEmpty())
    {
        qDebug() << Q_FUNC_INFO << "signaler course terminée :
" << champs.at(1) << numeroCourse;
        emit courseTerminee(champs.at(1));
    }
}
else // autre trame
{
    actualiserMessage(QString::fromUtf8("Mini-Display HL 975 :
%1 (non géré)".arg(trames.at(i))));
}
}
}
donnees.clear();
}
}

```

8.2.2.22 ChronoCrossClassement : :quitter() [private, slot]

Quitte l'application.

```

{
    // on ferme la fenêtre
    close();
}

```

8.2.2.23 ChronoCrossClassement : :selectionnerCourse() [private, slot]

Références [ui](#).

Référencé par [ChronoCrossClassement\(\)](#).

```

{

```

```

QStringList course;

int index = ui->comboBoxCourses->currentIndex();
switch(index)
{
case 0:
    //disconnect(ui->comboBoxCourses, SIGNAL(currentIndexChanged(int)),
    this, SLOT(selectionnerCourse()));
    ui->pushButtonDemarrageCourse->setEnabled(false);
    ui->pushButtonTopDepart->setEnabled(false);
    ui->pushButtonFinCourse->setEnabled(false);
    ui->pushButtonEnregistrementResultats->setEnabled(false);
    //connect(ui->comboBoxCourses, SIGNAL(currentIndexChanged(int)), this,
    SLOT(selectionnerCourse()));
    break;
default:
    if(index > 0)
    {
        ui->pushButtonDemarrageCourse->setEnabled(true);
        ui->pushButtonTopDepart->setEnabled(true);
        ui->pushButtonFinCourse->setEnabled(false);
        ui->pushButtonEnregistrementResultats->setEnabled(false);
        //course = courses.at(index-1);
    }
    break;
}
}

```

8.2.2.24 void ChronoCrossClassement : :tempsArrivee (QString ordre, QString temps) [signal]

Référencé par [finaliserDepartCourse\(\)](#), et [onReadyRead\(\)](#).

8.2.2.25 ChronoCrossClassement : :terminerCourse () [private, slot]

Termine la course.

Références [envoyer\(\)](#).

Référencé par [ChronoCrossClassement\(\)](#).

```

{
    QString * chaine = new QString("#WC 001");
    envoyer(*chaine);
    usleep(100000);

    QString heure = QDateTime::currentDateTime().toString("hh:mm");
    QString date = QDateTime::currentDateTime().toString("dd/MM/yy");

    // En mode CLOCK, on affiche l'heure actuelle
    chaine = new QString("#WC 007 02 "+heure+" "+date);
    envoyer(*chaine);
    // synchro
    chaine = new QString("#WC 008 01");
    envoyer(*chaine);
}

```

8.2.2.26 ChronoCrossClassement : :topDepart () [private, slot]

Demarre la course logiciellement.

Références [courseDemarree\(\)](#), [envoyer\(\)](#), [finaliserDepartCourse\(\)](#), [numeroter-Dossard\(\)](#), et [ui](#).

Référencé par [ChronoCrossClassement\(\)](#).

```

{
    // 0 -> Mode PTB
    // 5 -> Mode CLOCK
    QString * chaine = new QString("#WP 120 5");
    envoyer(*chaine);
    usleep(100000);

    chaine = new QString("#WC 007 02 00:00 00/00/01");
    envoyer(*chaine);

    chaine = new QString("#RP 003");
    envoyer(*chaine);

    chaine = new QString("#WC 008 01");
    envoyer(*chaine);

    connect(this, SIGNAL(courseDemarree(QString)), this, SLOT(
        finaliserDepartCourse(QString)));
    connect(ui->tableTemps, SIGNAL(cellChanged(int,int)), this, SLOT(
        numeroterDossard(int,int)));
}

```

8.2.2.27 ChronoCrossClassement : :verifierAcquittement (QString trame) [private]

Verifie l'acquittement.

Référencé par [onReadyRead\(\)](#).

```

{
    /*
     * Type de trame d'acquittement : cf. http://www.reliableracing.com/
     * downloads/THCOM08.pdf page 15
     * AK X
     * Avec : X = 'C' accepted, 'F' rejected, 'R' not supported
     */
    if(trame.startsWith("AK C"))
        return 'C';
    else if(trame.startsWith("AK F"))
        return 'F';
    else if(trame.startsWith("AK R"))
        return 'R';
    else
        return 'N'; // ce n'est pas une trame d'acquittement
}

```

8.2.3 Documentation des données membres

8.2.3.1 QVector<QString> ChronoCrossClassement : :arrivees [private]

Référencé par [afficherTableTemps\(\)](#), [enregistrerArriveeCoureur\(\)](#), [enregistrerResultat\(\)](#), [finaliserDepartCourse\(\)](#), et [finaliserFinCourse\(\)](#).

8.2.3.2 BaseDeDonnees* ChronoCrossClassement : :bd [private]

Référencé par [chargerInscritsCourse\(\)](#), [chargerListeCourses\(\)](#), [chargerListe-Manifestation\(\)](#), [enregistrerResultat\(\)](#), et [initialiserBD\(\)](#).

8.2.3.3 QVector<QStringList> ChronoCrossClassement : :courses [private]

Référencé par [chargerInscritsCourse\(\)](#), et [chargerListeCourses\(\)](#).

8.2.3.4 `bool ChronoCrossClassement : :displayEvent` [private]

Référencé par [initialiserPort\(\)](#).

8.2.3.5 `QByteArray ChronoCrossClassement : :donnees` [private]

Référencé par [onReadyRead\(\)](#).

8.2.3.6 `QVector<QString> ChronoCrossClassement : :dossards` [private]

Référencé par [afficherTableTemps\(\)](#), [enregistrerArriveeCoureur\(\)](#), [enregistrerResultat\(\)](#), [finaliserDepartCourse\(\)](#), [finaliserFinCourse\(\)](#), et [numeroterDossard\(\)](#).

8.2.3.7 `QVector<QStringList> ChronoCrossClassement : :inscrits` [private]

Référencé par [chargerInscritsCourse\(\)](#), [estInscrit\(\)](#), et [getIdInscrit\(\)](#).

8.2.3.8 `QStringList ChronoCrossClassement : :manifestation` [private]

Référencé par [chargerListeCourses\(\)](#), et [chargerListeManifestation\(\)](#).

8.2.3.9 `QString ChronoCrossClassement : :numeroCourse` [private]

Référencé par [ChronoCrossClassement\(\)](#), [enregistrerArriveeCoureur\(\)](#), [enregistrerResultat\(\)](#), [finaliserDepartCourse\(\)](#), [finaliserFinCourse\(\)](#), et [onReadyRead\(\)](#).

8.2.3.10 `QextSerialPort* ChronoCrossClassement : :port` [private]

Référencé par [ChronoCrossClassement\(\)](#), [envoyer\(\)](#), [initialiserPort\(\)](#), et [onReadyRead\(\)](#).

8.2.3.11 `Ui : :ChronoCrossClassement* ChronoCrossClassement : :ui` [private]

Référencé par [afficherTableTemps\(\)](#), [chargerInscritsCourse\(\)](#), [chargerListeCourses\(\)](#), [chargerListeManifestation\(\)](#), [ChronoCrossClassement\(\)](#), [demarrerCourse\(\)](#), [effacerTableTemps\(\)](#), [enregistrerArriveeCoureur\(\)](#), [finaliserDepartCourse\(\)](#), [finaliserFinCourse\(\)](#), [numeroterDossard\(\)](#), [selectionnerCourse\(\)](#), [topDepart\(\)](#), et [~ChronoCrossClassement\(\)](#).

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

- [chronocrossclassement.h](#)
- [chronocrossclassement.cpp](#)

8.3 Référence de la classe EditionCourse

La fenêtre d'édition de la course.

```
#include <editioncourse.h>
```

Fonctions membres publiques

- [EditionCourse](#) (QObject *parent=0)

- Constructeur de la classe *EditionCourse*.
– `~EditionCourse ()`

Connecteurs privés

- void `creerCourse ()`
Crée la course.
- void `modifierCourse ()`
Modifie la course.
- void `supprimerCourse ()`
Supprime la course.

Fonctions membres privées

- void `chargerListeCourses (QString idManifestation)`
Charge la liste des courses pour une manifestation.

8.3.1 Description détaillée

Auteur

BULIN Julien

Version

0.9

8.3.2 Documentation des constructeurs et destructeur

8.3.2.1 `EditionCourse : :EditionCourse (QObject * parent = 0) [explicit]`

Paramètres

<i>parent</i>	
---------------	--

```

                                :
    QObject (parent)
    {
    }

```

8.3.2.2 `EditionCourse : :~EditionCourse ()`

```

{
}

```

8.3.3 Documentation des fonctions membres

8.3.3.1 `EditionCourse : :chargerListeCourses (QString idManifestation) [private]`

Paramètres

<i>id- Manifestation</i>	QString l'identifiant de la manifestation
------------------------------	---

8.3.3.2 EditionCourse : :creerCourse () [private, slot]

```
{
}
```

8.3.3.3 EditionCourse : :modifierCourse () [private, slot]

```
{
}
```

8.3.3.4 EditionCourse : :supprimerCourse () [private, slot]

```
{
}
```

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

- [editioncourse.h](#)
- [editioncourse.cpp](#)

8.4 Référence de la classe EditionManifestation

La fenêtre d'édition de la manifestation.

```
#include <editionmanifestation.h>
```

Fonctions membres publiques

- [EditionManifestation](#) (QObject *parent=0)
Constructeur de la classe [EditionManifestation](#).
- [~EditionManifestation](#) ()

Connecteurs privés

- void [creerManifestation](#) ()
- void [modifierManifestation](#) ()
Modifie la manifestation.
- void [supprimerManifestation](#) ()
Supprime la manifestation.

Fonctions membres privées

- void [chargerListeManifestations](#) ()
Charge la liste des manifestations.

8.4.1 Description détaillée

Auteur

BULIN Julien

Version

0.9

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 EditionManifestation : :EditionManifestation (QObject * *parent* = 0)
[explicit]

Paramètres

<i>parent</i>	
---------------	--

```

                                :
    QObject (parent)
    {
    }

```

8.4.2.2 EditionManifestation : :~EditionManifestation ()

```

{
}

```

8.4.3 Documentation des fonctions membres

8.4.3.1 EditionManifestation : :chargerListeManifestations () [private]

8.4.3.2 EditionManifestation : :creerManifestation () [private, slot]

```

{
}

```

8.4.3.3 EditionManifestation : :modifierManifestation () [private, slot]

```

{
}

```

8.4.3.4 EditionManifestation : :supprimerManifestation () [private,
slot]

```

{
}

```

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

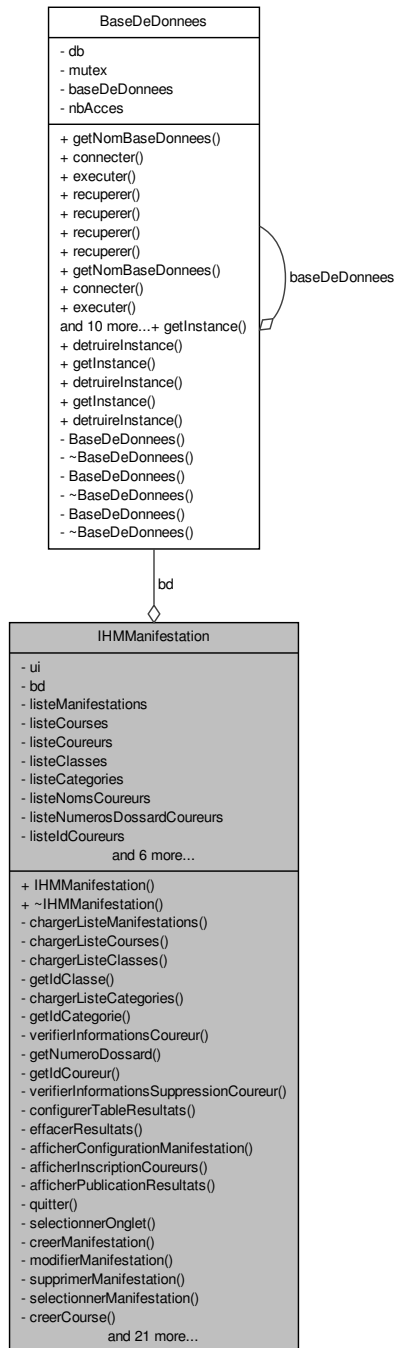
- [editionmanifestation.h](#)
- [editionmanifestation.cpp](#)

8.5 Référence de la classe IHMManifestation

La fenêtre principale de l'application.

```
#include <ihmmanifestation.h>
```

Grphe de collaboration de IHMManifestation :



Fonctions membres publiques

- [IHMManifestation](#) (QWidget *parent=0)
Constructeur de la classe IHMManifestation.
- [~IHMManifestation](#) ()
Destructeur de la classe IHMManifestation.

Connecteurs privés

- void [afficherConfigurationManifestation](#) ()
Affiche l'onglet pour configurer une manifestation.
- void [afficherInscriptionCoureurs](#) ()
Affiche l'onglet pour inscrire les coureurs.
- void [afficherPublicationResultats](#) ()
Affiche l'onglet pour publier les résultats.
- void [quitter](#) ()
Ferme l'application principale.
- void [selectionnerOnglet](#) (int index)
Initialise les données pour l'onglet affiché
- void [creerManifestation](#) ()
Crée la manifestation.
- void [modifierManifestation](#) ()
Modifie la manifestation.
- void [supprimerManifestation](#) ()
Supprime la manifestation.
- void [selectionnerManifestation](#) ()
Sélectionne la manifestation.
- void [creerCourse](#) ()
Crée la course.
- void [modifierCourse](#) ()
Modifie la course.
- void [supprimerCourse](#) ()
Supprime la course.
- void [selectionnerCourse](#) ()
Sélectionne la course.
- void [choisirInscriptionCourse](#) ()
Amène directement à l'onglet Inscription.
- void [creerCoureur](#) ()
Crée un coureur.
- void [modifierCoureur](#) ()
Modifie un coureur.
- void [supprimerCoureur](#) ()
Supprime un coureur.
- void [selectionnerCoureur](#) ()
Sélectionne le coureur.
- void [chargerListeCoureurs](#) ()
Charge la liste des coureurs.
- void [selectionnerListeCoureursCourse](#) ()
Sélectionne la liste des coureurs et des courses.
- void [afficherListeCoureurs](#) ()
Affiche la liste des coureurs.
- void [selectionnerCoureurListe](#) (const QModelIndex &index)
Sélectionne la liste des coureurs.
- void [selectionnerDossardListe](#) (const QModelIndex &index)
Sélectionne la liste des dossards.
- void [demarrerInscriptionCoureur](#) ()
Demarre l'inscription d'un coureur.
- void [inscrireCoureur](#) ()

- *Inscrit le coureur.*
- void [chargerListeInscriptionCourses](#) ()
Charge la liste des coureurs inscrit.
- void [afficherResultats](#) (QString idCourse)
Affiche les résultats de la course.
- void [activerImpression](#) ()
Détecte le déclenchement du bouton imprimer.
- void [ajouterResultat](#) (int ligne, QStringList unResultat)
Ajoute les résultats d'une course dans un tableau.
- void [selectionnerManifestationResultats](#) ()
Sélectionne une manifestation dans l'onglet résultats.
- void [selectionnerCourseResultats](#) ()
Sélectionne une course dans l'onglet résultats.
- void [imprimerResultats](#) ()
Imprime les résultats.

Fonctions membres privées

- void [chargerListeManifestations](#) ()
Charge les manifestations dans le sélectionneur.
- void [chargerListeCourses](#) (QString idManifestation)
Charge les courses dans le sélectionneur.
- void [chargerListeClasses](#) ()
Charge la liste des classes.
- QString [getIdClasse](#) ()
Obtiens l'identifiant de la classe.
- void [chargerListeCategories](#) ()
Charge la liste des categories.
- QString [getIdCategorie](#) ()
Obtiens l'identifiant de la categorie.
- bool [verifierInformationsCoureur](#) ()
Verifie les informations du coureur.
- QString [getNumeroDossard](#) (QString idCoureur, QString idCourse)
Obtiens le numero de dossard.
- QString [getIdCoureur](#) (QString nom)
Obtiens l'identifiant du coureur.
- bool [verifierInformationsSuppressionCoureur](#) ()
Verifie les informations de suppression du coureur.
- void [configurerTableResultats](#) ()
Configure le tableau de résultats dans l'onglet résultats.
- void [effacerResultats](#) ()
Efface les résultats dans l'onglet résultats.

Attributs privés

- Ui : :IHMManifestation * [ui](#)
relation vers la classe IHM
- [BaseDeDonnees](#) * [bd](#)
association vers la classe [BaseDeDonnees](#)
- QVector< QStringList > [listeManifestations](#)
la liste des manifestations créée dans la base de données
- QVector< QStringList > [listeCourses](#)
la liste des courses pour une manifestation
- QVector< QStringList > [listeCoureurs](#)
la liste des coureurs
- QVector< QStringList > [listeClasses](#)
la liste des classes pour les coureurs

- QVector< QStringList > [listeCategories](#)
la liste des catégories pour les coureurs
- QStringList [listeNomsCoureurs](#)
la liste des noms des coureurs pour le QStringListModel
- QStringList [listeNumerosDossardCoureurs](#)
la liste des numéros de dossard des coureurs pour le QStringListModel
- QStringList [listeIdCoureurs](#)
la liste des idCoureur des coureurs pour l'inscription
- QStringListModel * [coureurs](#)
la liste des noms des coureurs pour le QListView
- QStringListModel * [dossards](#)
la liste des numéros de dossard des coureurs pour le QListView
- QVector< QStringList > [listeManifestationsResultats](#)
la liste des manifestations créée dans la base de données dans l'onglet résultats
- QVector< QStringList > [listeCoursesResultats](#)
la liste des courses pour une manifestation dans l'onglet résultats
- QStringList [manifestation](#)
la manifestation pour l'onglet résultats
- bool [editionDossard](#)
état de l'édition d'un dossard.
- QString [idCoureur](#)
Identifiant du coureur.

8.5.1 Description détaillée

Auteur

BULIN Julien, PELLIZZONI Corentin

Version

1.1

8.5.2 Documentation des constructeurs et destructeur

8.5.2.1 IHMManifestation : :IHMManifestation (QWidget * *parent* = 0) [explicit]

Paramètres

<i>parent</i>	
---------------	--

Références [afficherConfigurationManifestation\(\)](#), [afficherInscriptionCoureurs\(\)](#), [afficherPublicationResultats\(\)](#), [bd](#), [chargerListeManifestations\(\)](#), [choisirInscriptionCourse\(\)](#), [configurerTableResultats\(\)](#), [BaseDeDonnees : :connecter\(\)](#), [coureurs](#), [creerCoureur\(\)](#), [demarrerInscriptionCoureur\(\)](#), [dossards](#), [BaseDeDonnees : :getInstance\(\)](#), [BaseDeDonnees : :getNomBaseDonnees\(\)](#), [imprimerResultats\(\)](#), [inscrireCoureur\(\)](#), [modifierCoureur\(\)](#), [modifierCourse\(\)](#), [modifierManifestation\(\)](#), [quitter\(\)](#), [selectionnerCoureur\(\)](#), [selectionnerCoureurListe\(\)](#), [selectionnerDossardListe\(\)](#), [selectionnerListeCoureursCourse\(\)](#), [selectionnerOnglet\(\)](#), [supprimerCoureur\(\)](#), [supprimerCourse\(\)](#), [supprimerManifestation\(\)](#), [TEMPO_STATUS](#), et [ui](#).

QMainWindow(*parent*),

:

```

    ui(new Ui::IHMManifestation)
{
    ui->setupUi(this);

    // Connexion à la base de données
    bd = BaseDeDonnees::getInstance();
    bool connexion = bd->connecter();
    if(connexion)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Connexion réussie à la
        base de données %1").arg(bd->getNomBaseDonnees()), TEMPO_STATUS);
    }
    else
    {
        ui->statusBar->showMessage(QString::fromUtf8("Impossible de se
        connecter à la base de données %1").arg(bd->getNomBaseDonnees()));
    }

    // Menu
    connect(ui->actionConfigurer_une_manifestation, SIGNAL(triggered()), this,
    SLOT(afficherConfigurationManifestation()));
    connect(ui->actionInscrire_les_coureurs, SIGNAL(triggered()), this, SLOT(
    afficherInscriptionCoureurs()));
    connect(ui->actionPublier_les_r_sultats, SIGNAL(triggered()), this, SLOT(
    afficherPublicationResultats()));
    connect(ui->actionQuitter, SIGNAL(triggered()), this, SLOT(quitter()));
    connect(ui->onglets, SIGNAL(currentChanged(int)), this, SLOT(
    selectionnerOnglet(int)));

    // Actions Manifestation : BULIN Julien
    connect(ui->boutonModificationManifestation, SIGNAL(clicked()), this, SLOT(
    modifierManifestation()));
    connect(ui->boutonSuppressionManifestation, SIGNAL(clicked()), this, SLOT(
    supprimerManifestation()));
    connect(ui->boutonInscrireCourse, SIGNAL(clicked()), this, SLOT(
    choisirInscriptionCourse()));
    connect(ui->boutonModificationCourse, SIGNAL(clicked()), this, SLOT(
    modifierCourse()));
    connect(ui->boutonSuppressionCourse, SIGNAL(clicked()), this, SLOT(
    supprimerCourse()));
    connect(ui->boutonImprimerResultats, SIGNAL(clicked()), this, SLOT(
    imprimerResultats()));

    // Actions Inscription : PELLIZZONI Corentin
    connect(ui->listeInscriptionCourses, SIGNAL(currentIndexChanged(int)), this
    , SLOT(selectionnerListeCoureursCourse()));
    //connect(ui->lineEditNumeroDossard, SIGNAL(editingFinished()), this,
    SLOT(inscrireCoureur()));
    connect(ui->lineEditNumeroDossard, SIGNAL(textEdited(QString)), this, SLOT(
    demarrerInscriptionCoureur()));
    connect(ui->lineEditNumeroDossard, SIGNAL(editingFinished()), this, SLOT(
    inscrireCoureur()));
    connect(ui->listeCoureurs, SIGNAL(currentIndexChanged(int)), this, SLOT(
    selectionnerCoureur()));
    connect(ui->listViewCoureurs, SIGNAL(clicked(const QModelIndex &)), this,
    SLOT(selectionnerCoureurListe(const QModelIndex &)));
    connect(ui->listViewCoureurs, SIGNAL(activated(const QModelIndex &)), this,
    SLOT(selectionnerCoureurListe(const QModelIndex &)));
    connect(ui->listViewDossards, SIGNAL(clicked(const QModelIndex &)), this,
    SLOT(selectionnerDossardListe(QModelIndex)));
    connect(ui->listViewDossards, SIGNAL(activated(const QModelIndex &)), this,
    SLOT(selectionnerDossardListe(QModelIndex)));
    connect(ui->boutonModificationCoureur, SIGNAL(clicked()), this, SLOT(
    modifierCoureur()));
    connect(ui->boutonSuppressionCoureur, SIGNAL(clicked()), this, SLOT(
    supprimerCoureur()));
    connect(ui->boutonCreationCoureur, SIGNAL(clicked()), this, SLOT(
    creerCoureur()));

    // Actions Résultats : BULIN Julien
    configurerTableResultats();

    chargerListeManifestations();
    coureurs = new QStringListModel(this);

```

```
dossards = new QStringListModel(this);  
afficherConfigurationManifestation();  
}
```

8.5.2.2 IHMManifestation : :~IHMManifestation ()

Références [bd](#), [BaseDeDonnees : :destruireInstance\(\)](#), et [ui](#).

```
{  
    bd->destruireInstance();  
    delete ui;  
}
```

8.5.3 Documentation des fonctions membres

8.5.3.1 IHMManifestation : :activerImpression () [private, slot]

Références [ui](#).

Référencé par [afficherResultats\(\)](#).

```
{  
    ui->boutonImprimerResultats->setEnabled(true);  
}
```

8.5.3.2 IHMManifestation : :afficherConfigurationManifestation () [private, slot]

Références [ONGLET_MANIFESTATION](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```
{  
    ui->onglets->setCurrentIndex(ONGLET_MANIFESTATION);  
}
```

8.5.3.3 IHMManifestation : :afficherInscriptionCoueurs () [private, slot]

Références [ONGLET_INSCRIPTION](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```
{  
    ui->onglets->setCurrentIndex(ONGLET_INSCRIPTION);  
}
```

8.5.3.4 void IHMManifestation : :afficherListeCoueurs () [private, slot]

Références [coueurs](#), [dossards](#), [listeNomsCoueurs](#), [listeNumerosDossardCoueurs](#), et [ui](#).

Référencé par [selectionnerListeCoueursCourse\(\)](#), et [selectionnerOnglet\(\)](#).

```

{
    if(ui->onglets->currentIndex() != 1)
        return;

    if(!ui->listeInscriptionCourses->currentText().isEmpty())
    {
        coureurs->setStringList(listeNomsCoureurs);
        ui->listViewCoureurs->setModel(coureurs);
        dossards->setStringList(listeNumerosDossardCoureurs);
        ui->listViewDossards->setModel(dossards);
    }
}

```

8.5.3.5 IHMManifestation : :afficherPublicationResultats () [private, slot]

Références [ONGLET_RESULTATS](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    ui->onglets->setCurrentIndex(ONGLET_RESULTATS);
}

```

8.5.3.6 IHMManifestation : :afficherResultats (QString idCourse) [private, slot]

Paramètres

<i>idCourse</i>	QString l'identifiant de la course
-----------------	------------------------------------

Références [activerImpression\(\)](#), [ajouterResultat\(\)](#), [bd](#), [effacerResultats\(\)](#), et [BaseDeDonnees : :recuperer\(\)](#).

Référencé par [selectionnerCourseResultats\(\)](#).

```

{
    // On commence par effacer le contenu précédent
    effacerResultats();

    // Récupérer les résultats d'une course (Requête SQL)
    QVector<QStringList> resultats;
    QString requete = "SELECT Inscrit.NumeroDossard, Coureur.Nom,
        Coureur.Prenom, Classe.Nom AS NomClasse, Classe.Numero, Arrivee.Temps FROM Course, Inscrit,
        Coureur, Classe, Arrivee WHERE Course.IdCourse = Inscrit.idCourse AND
        Inscrit.idCoureur = Coureur.idCoureur AND Coureur.idClasse = Classe.idClasse AND
        Inscrit.idInscrit = Arrivee.idInscrit AND Course.IdCourse = '" + idCourse + "' ORDER BY
        Arrivee.Temps ASC;";
    qDebug() << Q_FUNC_INFO << requete;
    bool retour = bd->recuperer(requete, resultats);
    if(retour != false)
    {
        QStringList unResultat;

        // Ajouter chaque élément dans le TableWidget
        for(int ligne = 0; ligne < resultats.size(); ligne++)
        {
            unResultat = resultats.at(ligne);

            ajouterResultat(ligne, unResultat);
        }

        activerImpression();
    }
}

```

8.5.3.7 IHMManifestation : :ajouterResultat (int ligne, QStringList unResultat) [private, slot]

Références [COLONNE_CLASSE](#), [COLONNE_CLASSEMENT](#), [COLONNE_NOM](#), [COLONNE_NUMERODOSSARD](#), [COLONNE_PRENOM](#), [COLONNE_TEMPS](#), [EMPLACEMENT_NOM](#), [EMPLACEMENT_NOMCLASSE](#), [EMPLACEMENT_NUMEROCLASSE](#), [EMPLACEMENT_NUMERODOSSARD](#), [EMPLACEMENT_PRENOM](#), [EMPLACEMENT_TEMPS](#), et [ui](#).

Référencé par [afficherResultats\(\)](#).

```
{
    QTableWidgetItem *elementClassement, *elementNumeroDossard, *elementNom, *
    elementPrenom, *elementClasse, *elementTemps;

    //Colonne Classement
    ui->tableWidgetResultats->insertRow(ligne);
    elementClassement = new QTableWidgetItem(QString::number(ligne+1));
    elementClassement->setFlags(Qt::ItemIsEnabled);
    if(ligne < 1)
    {
        elementClassement->setBackgroundColor(QColor(239,216,7));
        QFont font("", 16, QFont::Normal);
        font.setBold(true);
        elementClassement->setFont(font);
    }
    else if(ligne < 2)
    {
        elementClassement->setBackgroundColor(QColor(206,206,206));
        QFont font("", 16, QFont::Normal);
        font.setBold(true);
        elementClassement->setFont(font);
    }
    else if(ligne < 3)
    {
        elementClassement->setBackgroundColor(QColor(205,127,50));
        elementClassement->setForegroundColor(Qt::white);
        QFont font("", 16, QFont::Normal);
        font.setBold(true);
        elementClassement->setFont(font);
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_CLASSEMENT,
        elementClassement); // l : ligne (row) et c : colonne

    //Colonne Numéro de Dossard
    elementNumeroDossard = new QTableWidgetItem(unResultat.at(
        EMPLACEMENT_NUMERODOSSARD));
    elementNumeroDossard->setFlags(Qt::ItemIsEnabled);
    if(ligne < 1)
    {
        elementNumeroDossard->setBackgroundColor(QColor(239,216,7));
    }
    else if(ligne < 2)
    {
        elementNumeroDossard->setBackgroundColor(QColor(206,206,206));
    }
    else if(ligne < 3)
    {
        elementNumeroDossard->setBackgroundColor(QColor(205,127,50));
        elementNumeroDossard->setForegroundColor(Qt::white);
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_NUMERODOSSARD,
        elementNumeroDossard); // l : ligne (row) et c : colonne

    //Colonne Nom
    elementNom = new QTableWidgetItem(unResultat.at(EMPLACEMENT_NOM));
    elementNom->setFlags(Qt::ItemIsEnabled);
    if(ligne < 1)
    {
        elementNom->setBackgroundColor(QColor(239,216,7));
    }
}
```

```

    }
    else if(ligne < 2)
    {
        elementNom->setBackgroundColor(QColor(206,206,206));
    }
    else if(ligne < 3)
    {
        elementNom->setBackgroundColor(QColor(205,127,50));
        elementNom->setForeground(Qt::white);
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_NOM, elementNom); // l :
        ligne (row) et c : colonne

    //Colonne Prénom
    elementPrenom = new QTableWidgetItem(unResultat.at(EMPLACEMENT_PRENOM));
    elementPrenom->setFlags(Qt::ItemIsEnabled);
    if(ligne < 1)
    {
        elementPrenom->setBackgroundColor(QColor(239,216,7));
    }
    else if(ligne < 2)
    {
        elementPrenom->setBackgroundColor(QColor(206,206,206));
    }
    else if(ligne < 3)
    {
        elementPrenom->setBackgroundColor(QColor(205,127,50));
        elementPrenom->setForeground(Qt::white);
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_PRENOM, elementPrenom); //
        l : ligne (row) et c : colonne

    //Colonne Classe
    elementClasse = new QTableWidgetItem(unResultat.at(EMPLACEMENT_NOMCLASSE) +
        " " + unResultat.at(EMPLACEMENT_NUMEROCLASSE));
    elementClasse->setFlags(Qt::ItemIsEnabled);
    elementClasse->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
    if(ligne < 1)
    {
        elementClasse->setBackgroundColor(QColor(239,216,7));
    }
    else if(ligne < 2)
    {
        elementClasse->setBackgroundColor(QColor(206,206,206));
    }
    else if(ligne < 3)
    {
        elementClasse->setBackgroundColor(QColor(205,127,50));
        elementClasse->setForeground(Qt::white);
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_CLASSE, elementClasse); //
        l : ligne (row) et c : colonne

    //Colonne Temps
    elementTemps = new QTableWidgetItem(unResultat.at(EMPLACEMENT_TEMPS));
    elementTemps->setFlags(Qt::ItemIsEnabled);
    elementTemps->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
    if(ligne < 1)
    {
        elementTemps->setBackgroundColor(QColor(239,216,7));
    }
    else if(ligne < 2)
    {
        elementTemps->setBackgroundColor(QColor(206,206,206));
    }
    else if(ligne < 3)
    {
        elementTemps->setBackgroundColor(QColor(205,127,50));
        elementTemps->setForeground(Qt::white);
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_TEMPS, elementTemps); // l
        : ligne (row) et c : colonne
}

```

8.5.3.8 IHMManifestation : :chargerListeCategories () [private]

Charge la liste des catégories.

Références [bd](#), [listeCategories](#), [BaseDeDonnees : :recuperer\(\)](#), et [ui](#).

Référencé par [selectionnerOnglet\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    QString requete;
    QStringList categories;
    bool retour;

    // Initialise la liste
    ui->listeInscriptionsCategories->clear();
    listeCategories.clear();

    // Récupère dans la liste des categories dans un QVector de QStringList
    requete = "SELECT * FROM Categorie ORDER BY Nom ASC;";
    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->recuperer(requete, listeCategories);
    if(retour != false)
    {
        for(int i=0; i < listeCategories.size(); i++)
        {
            categories = listeCategories.at(i);
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Catégories : ") <<
            categories;
            ui->listeInscriptionsCategories->addItem(categories.at(1) + " " +
            categories.at(2));
        }
    }
}
```

8.5.3.9 IHMManifestation : :chargerListeClasses () [private]

Références [bd](#), [listeClasses](#), [BaseDeDonnees : :recuperer\(\)](#), et [ui](#).

Référencé par [selectionnerOnglet\(\)](#).

```
{
    QString requete;
    QStringList classe;
    bool retour;

    // Initialise la liste
    ui->listeInscriptionsClasses->clear();
    listeClasses.clear();

    // Récupère dans la liste des classes dans un QVector de QStringList
    requete = "SELECT * FROM Classe ORDER BY Nom ASC;";
    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->recuperer(requete, listeClasses);
    if(retour != false)
    {
        for(int i=0; i < listeClasses.size(); i++)
        {
            classe = listeClasses.at(i);
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Classe : ") << classe
            ;
            ui->listeInscriptionsClasses->addItem(classe.at(1) + " " + classe.
            at(2));
        }
    }
}
```

8.5.3.10 IHMManifestation : :chargerListeCoureurs () [private, slot]

Charge la liste coureur.

Références [bd](#), [listeCoureurs](#), [BaseDeDonnees : :recuperer\(\)](#), [selectionnerCoureur\(\)](#), [selectionnerListeCoureursCourse\(\)](#), et [ui](#).

Référencé par [creerCoureur\(\)](#), [modifierCoureur\(\)](#), [selectionnerOnglet\(\)](#), et [supprimerCoureur\(\)](#).

```
{
    QString requete;
    bool retour;

    qDebug() << Q_FUNC_INFO;

    // Réinitialise la liste
    disconnect(ui->listeCoureurs, SIGNAL(currentIndexChanged(int)), this, SLOT(
        selectionnerCoureur()));
    listeCoureurs.clear();
    ui->listeCoureurs->clear();
    ui->listeCoureurs->addItem("");

    // Récupère dans la liste des coureurs dans un QVector de QStringList
    requete = "SELECT Coureur.*,Classe.Nom as
        NomClasse,Classe.Numero,Categorie.Nom as NomCategorie,Categorie.Sexe as SexeCategorie FROM
        Coureur,Classe,Categorie WHERE Coureur.idCategorie = Categorie.idCategorie AND Coureur.idClasse =
        Classe.idClasse ORDER BY Coureur.Nom ASC;";
    //qDebug() << Q_FUNC_INFO << requete;
    retour = bd->recuperer(requete, listeCoureurs);
    if(retour != false)
    {
        for(int i=0; i < listeCoureurs.size(); i++)
        {
            //qDebug() << Q_FUNC_INFO << QString::fromUtf8("Coureur : ") <<
            listeCoureurs.at(i);
            ui->listeCoureurs->addItem(QString(listeCoureurs.at(i).at(4) +
                QString(" ") + listeCoureurs.at(i).at(5) + QString(" - ") + listeCoureurs.at(i).at(8)
                ) + QString(" ") + listeCoureurs.at(i).at(9) + QString(" - ") + listeCoureurs.at
                (i).at(10)));
        }
        selectionnerCoureur();
        selectionnerListeCoureursCourse();
    }
    connect(ui->listeCoureurs, SIGNAL(currentIndexChanged(int)), this, SLOT(
        selectionnerCoureur()));
}
```

8.5.3.11 IHMManifestation : :chargerListeCourses (QString idManifestation) [private]

Charge la liste des courses pour une manifestation.

Paramètres

<i>id- Manifestation</i>	QString l'identifiant de la manifestation
------------------------------	---

Références [bd](#), [COURSE_DISTANCE](#), [COURSE_HEUREDEPART](#), [COURSE_NOM](#), [listeCourses](#), [ONGLET_MANIFESTATION](#), [BaseDeDonnees : :recuperer\(\)](#), [selectionnerCourse\(\)](#), [selectionnerCourseResultats\(\)](#), et [ui](#).

Référencé par [creerCourse\(\)](#), [modifierCourse\(\)](#), [selectionnerManifestation\(\)](#), [selectionnerManifestationResultats\(\)](#), et [supprimerCourse\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    QString requete;
    QStringList course;
    bool retour;

    // Réinitialise la liste
    disconnect(ui->listeCreationCourses, SIGNAL(currentIndexChanged(int)), this,
        SLOT(selectionnerCourse()));
    disconnect(ui->listeCreationCoursesResultats, SIGNAL(currentIndexChanged(
        int)), this, SLOT(selectionnerCourseResultats()));
    listeCourses.clear();
    ui->listeCreationCourses->clear();
    ui->listeCreationCourses->addItem(""); // index 0
    ui->listeCreationCoursesResultats->clear();
    ui->listeCreationCoursesResultats->addItem(""); // index 0

    // Récupère la liste des courses d'une manifestation dans un QVector de
    // QStringList
    requete = "SELECT * FROM Course WHERE idManifestation = '" +
        idManifestation + "' ORDER BY HeureDepart ASC;";
    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->recuperer(requete, listeCourses);
    if(retour != false)
    {
        for(int i=0; i < listeCourses.size(); i++)
        {
            course = listeCourses.at(i);
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Course : ") << course
            ;
            QTime heureDepart = QTime::fromString(course.at(COURSE_HEUREDEPART)
            , "hh:mm:ss");
            ui->listeCreationCourses->addItem(course.at(COURSE_NOM) + " - " +
            course.at(COURSE_DISTANCE) + "m - " + heureDepart.toString("hh'h'mm"));
            ui->listeCreationCoursesResultats->addItem(course.at(COURSE_NOM) +
            " - " + course.at(COURSE_DISTANCE) + "m - " + heureDepart.toString("hh'h'mm"));
        }
    }

    // Sélectionne par défaut
    if(ui->onglets->currentIndex() == ONGLET_MANIFESTATION)
        selectionnerCourse();
    connect(ui->listeCreationCourses, SIGNAL(currentIndexChanged(int)), this,
        SLOT(selectionnerCourse()));
    connect(ui->listeCreationCoursesResultats, SIGNAL(currentIndexChanged(int))
        , this, SLOT(selectionnerCourseResultats()));
}

```

8.5.3.12 IHMManifestation : :chargerListeInscriptionCourses() [private, slot]

Charge la liste des inscrits a une course.

Références `bd`, `COURSE_NOM`, `listeCourses`, `BaseDeDonnees : :recuperer()`, et `ui`.

Référencé par `selectionnerOnglet()`.

```

{
    qDebug() << Q_FUNC_INFO;
    QString requete;
    QStringList course;
    bool retour;

    // Réinitialise la liste
    //disconnect(ui->listeInscriptionCourses, SIGNAL(currentIndexChanged(int)),
    //    this, SLOT(selectionnerCourse()));
    listeCourses.clear();
    ui->listeInscriptionCourses->clear();
    ui->listeInscriptionCourses->addItem(""); // index 0
}

```

```

// Récupère la liste des courses d'une manifestation dans un QVector de
// QStringList
requete = "SELECT * FROM Course";
qDebug() << Q_FUNC_INFO << requete;
retour = bd->recuperer(requete, listeCourses);
if(retour != false)
{
    for(int i=0; i < listeCourses.size(); i++)
    {
        course = listeCourses.at(i);
        qDebug() << Q_FUNC_INFO << QString::fromUtf8("Course : ") << course
        ;
        ui->listeInscriptionCourses->addItem(course.at(COURSE_NOM));
    }
}
}

```

8.5.3.13 IHMManifestation : :chargerListeManifestations() [private]

Charge la liste des manifestations.

Références [bd](#), [listeManifestations](#), [manifestation](#), [ONGLET_MANIFESTATION](#), [BaseDeDonnees : :recuperer\(\)](#), [selectionnerManifestation\(\)](#), [selectionnerManifestation-Resultats\(\)](#), et [ui](#).

Référencé par [creerManifestation\(\)](#), [IHMManifestation\(\)](#), [modifierManifestation\(\)](#), et [supprimerManifestation\(\)](#).

```

{
    QString requete;
    QStringList manifestation;
    bool retour;

    // Réinitialise la liste
    disconnect(ui->listeCreationManifestations, SIGNAL(currentIndexChanged(int))
    ), this, SLOT(selectionnerManifestation()));
    disconnect(ui->listeCreationManifestationsResultats, SIGNAL(
    currentIndexChanged(int)), this, SLOT(selectionnerManifestationResultats()));
    listeManifestations.clear();
    ui->listeCreationManifestations->clear();
    ui->listeCreationManifestations->addItem(""); // index 0
    ui->listeCreationManifestationsResultats->clear();
    ui->listeCreationManifestationsResultats->addItem(""); // index 0

    // Récupère dans la liste des manifestations dans un QVector de QStringList
    requete = "SELECT * FROM Manifestation ORDER BY Date ASC;";
    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->recuperer(requete, listeManifestations);
    if(retour != false)
    {
        for(int i=0; i < listeManifestations.size(); i++)
        {
            manifestation = listeManifestations.at(i);
            qDebug() << Q_FUNC_INFO << QString::fromUtf8("Manifestation : ") <<
            manifestation;
            ui->listeCreationManifestations->addItem(manifestation.at(1));
            ui->listeCreationManifestationsResultats->addItem(manifestation.at(
            1));
        }
    }

    // Sélectionne par défaut
    if(ui->onglets->currentIndex() == ONGLET_MANIFESTATION)
        selectionnerManifestation();
    connect(ui->listeCreationManifestations, SIGNAL(currentIndexChanged(int)),
    this, SLOT(selectionnerManifestation()));
    connect(ui->listeCreationManifestationsResultats, SIGNAL(
    currentIndexChanged(int)), this, SLOT(selectionnerManifestationResultats()));
}

```

```
}

```

8.5.3.14 IHMManifestation : :choisirInscriptionCourse() [private, slot]

Référencé par [IHMManifestation\(\)](#).

```
{
    QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
        QString::fromUtf8("TODO Sélectionner l'onglet Inscription !"));
    // Le bouton Inscrire amène directement à l'onglet Résultats en attendant
    // la correction des erreurs de la page Inscription
    //ui->onglets->setCurrentIndex(ONGLET_INSCRIPTION);
    //ui->lineEditNomCoureur->setFocus();
}
```

8.5.3.15 IHMManifestation : :configurerTableResultats() [private]

Références [ui](#).

Référencé par [IHMManifestation\(\)](#).

```
{
    QStringList nomColonnes; // nom des colonnes
    nomColonnes << QString::fromUtf8("Classement") << QString::fromUtf8("Numéro
    de dossard") << QString::fromUtf8("Nom") << QString::fromUtf8("Prénom") <<
    QString::fromUtf8("Classe") << QString::fromUtf8("Temps"); // ...

    // On fixe le nombre de colonnes
    ui->tableWidgetResultats->setColumnCount(nomColonnes.size());
    // On applique les noms des colonnes
    ui->tableWidgetResultats->setHorizontalHeaderLabels(nomColonnes);

    // on cache les numéros de ligne
    ui->tableWidgetResultats->verticalHeader()->setHidden(true);

    QHeaderView * headerView = ui->tableWidgetResultats->horizontalHeader();
    // on redimensionne automatiquement la colonne pour occuper l'espace
    // disponible
    #if QT_VERSION >= 0x050000
    headerView->setSectionResizeMode(QHeaderView::Stretch);
    #else
    headerView->setResizeMode(QHeaderView::Stretch);
    #endif
}
```

8.5.3.16 IHMManifestation : :créerCoureur() [private, slot]

Crée un coureur.

Références [bd](#), [chargerListeCoureurs\(\)](#), [BaseDeDonnees : :executer\(\)](#), [getId-Categorie\(\)](#), [getIdClasse\(\)](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Vérifications
    /*if(!verifierInformationsCoureur())
    {
        ui->statusBar->showMessage(QString::fromUtf8("Les informations sur le

```

```

        coureur sont invalides ou manquantes"), 2000);
        return;
    }*/

    QString requete = "INSERT INTO
        Coureur(idCategorie,idClasse,INE,Nom,Prenom,DateNaissance,Sexe) VALUES('" + getIdCategorie() + "','" + get
        + ui->lineEditNumeroINEleve->text() + "','" + ui->lineEditNomCoureur->text() +
        "','" + ui->lineEditPrenomCoureur->text() + "','" + ui->
        dateEditDateNaissanceCoureur->date().toString("yyyy-MM-dd") + "','" + ui->listeInscriptionSexe->
        currentText() + "')";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Création du coureur %1
réussie").arg(ui->lineEditNomCoureur->text()), 2000);
        chargerListeCoureurs();
    }
    else
    {
        ui->statusBar->showMessage(QString::fromUtf8("Impossible de créer le
coureur %1 !").arg(ui->lineEditNomCoureur->text()), 2000);
    }
}

```

8.5.3.17 IHMManifestation : :creerCourse () [private, slot]

A faire vérifier si la validité de l'heure

Références [bd](#), [chargerListeCourses\(\)](#), [BaseDeDonnees : :executer\(\)](#), [liste-Manifestations](#), [TEMPO_STATUS](#), et [ui](#).

```

{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Vérifications
    if(ui->lineEditNomCourse->text().isEmpty())
    {
        QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
            QString::fromUtf8("Il faut indiquer un nom et la longueur de la course !"));
        ui->statusBar->showMessage(QString::fromUtf8("Il faut indiquer un nom
et la longueur de la course"), TEMPO_STATUS);
        return;
    }

    QString idManifestation = listeManifestations.at(ui->
        listeCreationManifestations->currentIndex()-1).at(0);
    QString requete = "INSERT INTO
        Course(idManifestation,Nom,HeureDepart,Distance) VALUES('" + idManifestation + "','" + ui->lineEditNomCourse->
        + ui->timeEditHeureDebutCourse->time().toString("hh:mm:ss") + "','" + ui->
        lineEditLongueurCourse->text() + "')";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Création de la course %1
réussie").arg(ui->lineEditNomCourse->text()), TEMPO_STATUS);
        chargerListeCourses(idManifestation);
    }
}

```

8.5.3.18 IHMManifestation : :creerManifestation () [private, slot]

A faire vérifier si la date n'est pas passée

Références [bd](#), [chargerListeManifestations\(\)](#), [BaseDeDonnees : :executer\(\)](#), et [ui](#).

```
{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Vérifications
    if(ui->lineEditNomManifestation->text().isEmpty())
    {
        QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
            QString::fromUtf8("Il faut indiquer un nom de manifestation !"));
        ui->statusBar->showMessage(QString::fromUtf8("Il faut indiquer un nom
            de manifestation"), 2000);
        return;
    }

    QString requete = "INSERT INTO Manifestation(Nom,Date) VALUES(' " + ui->
        lineEditNomManifestation->text() + "', ' " + ui->dateEditEditionManifestation->date().
        toString("yyyy-MM-dd") + "')";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Création de la
            manifestation %1 réussie").arg(ui->lineEditNomManifestation->text()), 2000);
        chargerListeManifestations();
    }
}
```

8.5.3.19 IHMManifestation : :demarrerInscriptionCoureur () [private, slot]

Démarre l'inscription d'un coureur.

Références [editionDossard](#).

Référencé par [IHMManifestation\(\)](#).

```
{
    // début de l'édition
    editionDossard = true;
}
```

8.5.3.20 IHMManifestation : :effacerResultats () [private]

Références [ui](#).

Référencé par [afficherResultats\(\)](#), et [selectionnerCourseResultats\(\)](#).

```
{
    int nb = ui->tableWidgetResultats->rowCount();
    // on efface les lignes du tableau une par une
    for(int i = 0; i < nb; i++)
    {
        ui->tableWidgetResultats->removeRow(0);
    }
    ui->boutonImprimerResultats->setEnabled(false);
}
```

8.5.3.21 QString IHMManifestation : getIdCategorie () [private]

Références [listeCategories](#), et [ui](#).

Référencé par [creerCoureur\(\)](#), et [modifierCoureur\(\)](#).

```
{
    qDebug() << Q_FUNC_INFO;
    return listeCategories.at(ui->listeInscriptionsCategories->currentIndex()).
        at(0);
}
```

8.5.3.22 IHMManifestation : getIdClasse () [private]

Obtiens l'idClasses.

Références [listeClasses](#), et [ui](#).

Référencé par [creerCoureur\(\)](#), et [modifierCoureur\(\)](#).

```
{
    return listeClasses.at(ui->listeInscriptionsClasses->currentIndex()).at(0);
}
```

8.5.3.23 IHMManifestation : getIdCoureur (QString nom) [private]

Obtiens l'idCoureur.

Références [listeIdCoureurs](#), et [listeNomsCoureurs](#).

Référencé par [inscrireCoureur\(\)](#).

```
{
    for(int i = 0; i < listeNomsCoureurs.size(); i++)
    {
        if(listeNomsCoureurs.at(i) == nom)
            return listeIdCoureurs.at(i);
    }
    return QString("");
}
```

8.5.3.24 IHMManifestation : getNumeroDossard (QString idCoureur, QString idCourse) [private]

Obtiens les numéros de dossard.

Paramètres

<i>idCoureur</i>	QString l'identifiant du coureur
<i>idCourse</i>	QString l'identifiant de la course

Références [bd](#), et [BaseDeDonnees : recuperer\(\)](#).

Référencé par [selectionnerListeCoureursCourse\(\)](#).

```
{
    QString numeroDossard;
```

```

QString requete;
bool retour;

if(idCoureur.isEmpty())
    return QString("");
if(idCourse.isEmpty())
    return QString("");

// Récupère le numéro de dossard dans un QString
requete = "SELECT NumeroDossard FROM Inscrit WHERE idCoureur = ' " +
    idCoureur + "' AND idCourse = ' " + idCourse + "'";
//qDebug() << Q_FUNC_INFO << requete;
retour = bd->recuperer(requete, numeroDossard);
if(retour != false)
{
    return numeroDossard;
}
return QString("");
}

```

8.5.3.25 void IHMManifestation ::imprimerResultats () [private, slot]

Références [manifestation](#), [MANIFESTATION_DATE](#), [MANIFESTATION_NOM](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    QPrinter printer;
    printer.setOrientation(QPrinter::Landscape);
    QPrintDialog *dialog = new QPrintDialog(&printer, this);
    dialog->setWindowTitle(tr("Imprimer"));
    if (dialog->exec() == QDialog::Accepted)
    {
        QPainter painter;
        painter.begin(&printer);

        painter.save();
        QString datestring = manifestation.at(MANIFESTATION_DATE);
        QDate date = QDate::fromString(datestring, "yyyy-MM-dd");

        // Les informations sur la manifestation et la course à imprimer
        painter.drawText(40, 80, QString::fromUtf8("Manifestation : ") +
            manifestation.at(MANIFESTATION_NOM).toUtf8() + " - " + QString::fromUtf8("Date
            : ") + date.toString("dd-MM-yyyy"));
        painter.drawText(40, 120, QString::fromUtf8("Course : ") + ui->
            label_ChargerNomCourseResultats->text().toUtf8() + " - " + QString::fromUtf8("Heure de
            départ : ") + ui->label_ChargerHeureDepartCourseResultats->text().toUtf8() + " -
            " + QString::fromUtf8("Longueur de la course : ") + ui->
            label_ChargerLongueurCourseResultats->text().toUtf8() + QString::fromUtf8(" m"));
        painter.restore();

        // Les résultats
        painter.translate(40, 160);
        ui->tableWidgetResultats->render(&painter);
    }
}

```

8.5.3.26 IHMManifestation ::inscrireCoureur () [private, slot]

Inscrit un coureur.

Références [coureurs](#), [editionDossard](#), [getIdCoureur\(\)](#), [idCoureur](#), [listeNomsCoureurs](#), [selectionnerCoureurListe\(\)](#), [selectionnerListeCoureursCourse\(\)](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{

```

```

QString requete;
// édition en cours ?
if(editionDossard == true)
{
    //qDebug() << Q_FUNC_INFO << ui->lineEditNomInscrit->text() <<
    ui->lineEditNumeroDossard->text();
    //qDebug() << Q_FUNC_INFO << "DOSSARD nouveau : " <<
    ui->lineEditNumeroDossard->text();
    //qDebug() << Q_FUNC_INFO << "DOSSARD ancien : ";
    //qDebug() << Q_FUNC_INFO << "Id Course : " <<
    listeCourses.at(ui->listeInscriptionCourses->currentIndex()-1).at(0));
    QString idCoureur = getIdCoureur(ui->lineEditNomInscrit->text());
    QString idCourse;
    qDebug() << Q_FUNC_INFO << "Id Coureur : " << idCoureur;
    qDebug() << Q_FUNC_INFO << "Id Course : " << idCourse;

    //requete = "INSERT Inscrit.NumeroDossard FROM Coureur WHERE idCoureur
    = '" + idCoureur + "' AND idCourse = '" + idCourse + "'";

    selectionnerListeCoureursCourse();
}
// fin de l'édition
editionDossard = false;

// on passe au suivant (Amélioration Thierry Vaira)
for(int i = 0; i < listeNomsCoureurs.size(); i++)
{
    if(listeNomsCoureurs.at(i) == ui->lineEditNomInscrit->text())
    {
        QModelIndex index = coureurs->index(++i%coureurs->rowCount(), 0);
        ui->listViewDossards->setCurrentIndex(index);
        ui->listViewCoureurs->setCurrentIndex(index);
        selectionnerCoureurListe(index);
        break;
    }
}
}

```

8.5.3.27 IHMManifestation : :modifierCoureur () [private, slot]

Références `bd`, `chargerListeCoureurs()`, `COUREUR_ID`, `BaseDeDonnees : :executer()`, `getIdCategorie()`, `getIdClasse()`, `listeCoureurs`, `TEMPO_STATUS`, et `ui`.

Référencé par `IHMManifestation()`.

```

{
    if(ui->listeCoureurs->currentIndex() == 0)
        return;
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    QString requete = "UPDATE Coureur SET INE = '" + ui->lineEditNumeroINEleve
    ->text() + "', Nom = '" + ui->lineEditNomCoureur->text() + "', Prenom = '" + ui
    ->lineEditPrenomCoureur->text() + "', DateNaissance = '" + ui->
    dateEditDateNaissanceCoureur->date().toString("yyyy-MM-dd") + "', Sexe = '" + ui->
    listeInscriptionSexe->currentText() + "', idCategorie = '" + getIdCategorie() + "', idClasse = '"
    + getIdClasse() + "' WHERE idCoureur = '" + listeCoureurs.at(ui->listeCoureurs
    ->currentIndex()-1).at(COUREUR_ID) + "'";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);
    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Modification du coureur
        %1 réussie").arg(ui->lineEditNomCoureur->text()), TEMPO_STATUS);
        chargerListeCoureurs();
    }
    else
    {
        ui->statusBar->showMessage(QString::fromUtf8("Impossible de modifier le

```

```

        coureur %1 !").arg(ui->lineEditNomCoureur->text()), TEMPO_STATUS);
    }
}

```

8.5.3.28 IHMManifestation : :modifierCourse() [private, slot]

A faire vérifier si la validité de l'heure

Références [bd](#), [chargerListeCourses\(\)](#), [BaseDeDonnees : :executer\(\)](#), [listeCourses](#), [listeManifestations](#), [TEMPO_STATUS](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Vérifications
    if(ui->lineEditNomCourse->text().isEmpty())
    {
        QMessageBox::critical(0, QString::fromUtf8("Chrono-Cross"),
            QString::fromUtf8("Il faut indiquer un nom de course !"));
        ui->statusBar->showMessage(QString::fromUtf8("Il faut indiquer un nom
de course"), TEMPO_STATUS);
        return;
    }

    // Récupère l'identifiant de la course
    QString idCourse = listeCourses.at(ui->listeCreationCourses->currentIndex()
-1).at(0);

    QString requete = "UPDATE Course SET Nom = '" + ui->lineEditNomCourse->text
() + "', HeureDepart = '" + ui->timeEditHeureDebutCourse->time().toString("
hh:mm:ss") + "', Distance = '" + ui->lineEditLongueurCourse->text() + "' WHERE
idCourse = '" + idCourse + "'";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Modification de la course
%1 réussie").arg(ui->lineEditNomCourse->text()), TEMPO_STATUS);
        chargerListeCourses(listeManifestations.at(ui->
listeCreationManifestations->currentIndex()-1).at(0));
    }
}

```

8.5.3.29 IHMManifestation : :modifierManifestation() [private, slot]

A faire vérifier si la date n'est pas passée

Références [bd](#), [chargerListeManifestations\(\)](#), [BaseDeDonnees : :executer\(\)](#), [liste-Manifestations](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Vérifications
    if(ui->lineEditNomManifestation->text().isEmpty())

```

```

{
    ui->statusBar->showMessage(QString::fromUtf8("Il faut indiquer un nom
de manifestation"), 2000);
    return;
}

// Récupère l'identifiant de la manifestation
QString idManifestation = listeManifestations.at(ui->
listeCreationManifestations->currentIndex()-1).at(0);

QString requete = "UPDATE Manifestation SET Nom = '" + ui->
lineEditNomManifestation->text() + "', Date = '" + ui->dateEditEditionManifestation->date().
toString("yyyy-MM-dd") + "' WHERE idManifestation = '" + idManifestation + "'";

qDebug() << Q_FUNC_INFO << requete;
retour = bd->executer(requete);

if(retour)
{
    ui->statusBar->showMessage(QString::fromUtf8("Modification de la
manifestation %1 réussie").arg(ui->lineEditNomManifestation->text()), 2000);
    chargerListeManifestations();
}
}

```

8.5.3.30 IHMManifestation : :quitter () [private, slot]

Quitte l'application.

Référencé par [IHMManifestation\(\)](#).

```

{
    // on ferme la fenêtre
    close();
}

```

8.5.3.31 IHMManifestation : :selectionnerCoureur () [private, slot]

Sélectionne un coureur.

Références [COUREUR_DATENAissance](#), [COUREUR_NOM](#), [COUREUR_NOM_C-ATEGORIE](#), [COUREUR_NOM_CLASSE](#), [COUREUR_NUMERO_CLASSE](#), [COUREUR_NUMERO_INE](#), [COUREUR_PRENOM](#), [COUREUR_SEXE](#), [COUREUR_SEXE_CATEGORIE](#), [listeCoureurs](#), et [ui](#).

Référencé par [chargerListeCoureurs\(\)](#), et [IHMManifestation\(\)](#).

```

{
    QDate DateNaissance;
    QStringList coureur;
    int position;

    int index = ui->listeCoureurs->currentIndex();

    qDebug() << Q_FUNC_INFO << index;
    switch(index)
    {
    case 0:
        ui->lineEditNomCoureur->setText("");
        ui->lineEditPrenomCoureur->setText("");
        DateNaissance = QDate::fromString("2000-01-01", "yyyy-MM-dd");
        ui->dateEditDateNaissanceCoureur->setDate(DateNaissance);
        ui->listeInscriptionSexe->setCurrentIndex(0);
        ui->lineEditNumeroINEleve->setText("");
        ui->listeInscriptionsClasses->setCurrentIndex(0);
    }
}

```

```

        ui->listeInscriptionsCategories->setCurrentIndex(0);
        ui->boutonCreationCoureur->setEnabled(true);
        ui->boutonModificationCoureur->setEnabled(false);
        ui->boutonSuppressionCoureur->setEnabled(false);
        break;
    default:
        if(index > 0)
        {
            coureur = listeCoureurs.at(index-1);
            qDebug() << Q_FUNC_INFO << coureur;
            ui->lineEditNomCoureur->setText(coureur.at(COUREUR_NOM));
            ui->lineEditPrenomCoureur->setText(coureur.at(COUREUR_PRENOM));
            DateNaissance = QDate::fromString(coureur.at(
COUREUR_DATENAISSANCE), "yyyy-MM-dd");
            ui->dateEditDateNaissanceCoureur->setDate(DateNaissance);
            position = ui->listeInscriptionSexe->findText(coureur.at(
COUREUR_SEXE));
            if(position != -1)
                ui->listeInscriptionSexe->setCurrentIndex(position);
            ui->lineEditNumeroINEleve->setText(coureur.at(
COUREUR_NUMERO_INE));
            position = ui->listeInscriptionsClasses->findText(coureur.at(
COUREUR_NOM_CLASSE) + " " + coureur.at(COUREUR_NUMERO_CLASSE));
            if(position != -1)
                ui->listeInscriptionsClasses->setCurrentIndex(position);
            position = ui->listeInscriptionsCategories->findText(coureur.at(
COUREUR_NOM_CATEGORIE) + " " + coureur.at(COUREUR_SEXE_CATEGORIE) + " ");
            if(position != -1)
                ui->listeInscriptionsCategories->setCurrentIndex(position);
            ui->boutonCreationCoureur->setEnabled(false);
            ui->boutonModificationCoureur->setEnabled(true);
            ui->boutonSuppressionCoureur->setEnabled(true);
        }
        break;
    }
}

```

8.5.3.32 IHMManifestation : :selectionnerCoureurListe (const QModelIndex & index) [private, slot]

Références [dossards](#), [listeCoureurs](#), [listeNumerosDossardCoureurs](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#), et [inscrireCoureur\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << index.data() << index.row();
    if(listeCoureurs.count() > 0)
    {
        QModelIndex indexDossard = dossards->index(index.row(), 0);
        ui->listViewDossards->scrollTo(indexDossard);
        ui->lineEditNomInscrit->setText(index.data().toString());
        ui->lineEditNumeroDossard->setText(listeNumerosDossardCoureurs.at(index
        .row()));
        ui->lineEditNumeroDossard->setFocus();
    }
}

```

8.5.3.33 IHMManifestation : :selectionnerCourse () [private, slot]

Sélectionne une course à éditer.

Références [COURSE_DISTANCE](#), [COURSE_HEUREDEPART](#), [COURSE_NOM](#), [listeCourses](#), et [ui](#).

Référencé par [chargerListeCourses\(\)](#), et [selectionnerManifestation\(\)](#).

```

{

```

```

QTime heureDebutCourse(8, 0);
QStringList course;

int index = ui->listeCreationCourses->currentIndex();

switch(index)
{
case 0:
    ui->timeEditHeureDebutCourse->setTime(heureDebutCourse);
    ui->lineEditNomCourse->setText("");
    ui->lineEditLongueurCourse->setText("");
    if(ui->listeCreationManifestations->currentIndex() > 0)
    {
        ui->boutonCreationCourse->setEnabled(true);
    }
    else
    {
        disconnect(ui->listeCreationCourses, SIGNAL(currentIndexChanged(int)), this, SLOT(selectionnerCourse()));
        listeCourses.clear();
        ui->listeCreationCourses->clear();
        ui->listeCreationCourses->addItem(""); // index 0
        ui->listeInscriptionCourses->clear();
        ui->boutonCreationCourse->setEnabled(false);
        connect(ui->listeCreationCourses, SIGNAL(currentIndexChanged(int)), this, SLOT(selectionnerCourse()));
    }
    ui->boutonModificationCourse->setEnabled(false);
    ui->boutonSuppressionCourse->setEnabled(false);
    ui->boutonInscrireCourse->setEnabled(false);
    break;
default:
    if(index > 0)
    {
        course = listeCourses.at(index-1);
        QTime heureCourse = QTime::fromString(course.at(COURSE_HEUREDEPART), "hh:mm:ss");
        ui->timeEditHeureDebutCourse->setTime(heureCourse);
        ui->lineEditNomCourse->setText(course.at(COURSE_NOM));
        ui->lineEditLongueurCourse->setText(course.at(COURSE_DISTANCE));

        ui->boutonCreationCourse->setEnabled(false);
        ui->boutonModificationCourse->setEnabled(true);
        ui->boutonSuppressionCourse->setEnabled(true);
        ui->listeInscriptionCourses->setCurrentIndex(index-1);
        ui->boutonInscrireCourse->setEnabled(true);
    }
    break;
}
}

```

8.5.3.34 IHMManifestation : :selectionnerCourseResultats () [private, slot]

Références [afficherResultats\(\)](#), [COURSE_DISTANCE](#), [COURSE_HEUREDEPART](#), [COURSE_NOM](#), [effacerResultats\(\)](#), [listeCourses](#), et [ui](#).

Référencé par [chargerListeCourses\(\)](#), et [selectionnerManifestationResultats\(\)](#).

```

{
    QStringList course;

    int index = ui->listeCreationCoursesResultats->currentIndex();
    ui->label_ChargerNomCourseResultats->clear();
    ui->label_ChargerHeureDepartCourseResultats->clear();
    ui->label_ChargerLongueurCourseResultats->clear();

    switch(index)
    {
case 0:

```

```

        effacerResultats();
        break;
    default:
        if(index > 0)
        {
            course = listeCourses.at(index-1);
            QTime heureCourse = QTime::fromString(course.at(COURSE_HEUREDEPART)
, "hh:mm:ss");
            ui->label_ChargerNomCourseResultats->setText(course.at(COURSE_NOM))
;
            ui->label_ChargerHeureDepartCourseResultats->setText(heureCourse.
toString("hh'h'mm"));
            ui->label_ChargerLongueurCourseResultats->setText(course.at(
COURSE_DISTANCE));
            afficherResultats(course.at(0));
        }
        break;
    }
}

```

8.5.3.35 IHMManifestation : :selectionnerDossardListe (const QModelIndex & index) [private, slot]

Références [coureurs](#), [listeCoureurs](#), [listeNumerosDossardCoureurs](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO << index.data() << index.row();
    if(listeCoureurs.count() > 0)
    {
        qDebug() << Q_FUNC_INFO;
        QModelIndex indexCoureur = coureurs->index(index.row(), 0);
        ui->listViewCoureurs->scrollTo(indexCoureur);
        ui->lineEditNomInscrit->setText(indexCoureur.data().toString());
        ui->lineEditNumeroDossard->setText(listeNumerosDossardCoureurs.at(index
.row()));
        ui->lineEditNumeroDossard->setFocus();
    }
}

```

8.5.3.36 IHMManifestation : :selectionnerListeCoureursCourse () [private, slot]

Sélectionne la liste des coureurs dans une course.

Références [afficherListeCoureurs\(\)](#), [coureurs](#), [dossards](#), [getNumeroDossard\(\)](#), [listeCoureurs](#), [listeCourses](#), [listeldCoureurs](#), [listeNomsCoureurs](#), [listeNumerosDossardCoureurs](#), et [ui](#).

Référencé par [chargerListeCoureurs\(\)](#), [IHMManifestation\(\)](#), et [inscrireCoureur\(\)](#).

```

{
    // aucune course sélectionnée ?
    if(ui->listeInscriptionCourses->currentIndex() == 0)
    {
        // Efface les QListView
        listeNomsCoureurs.clear();
        listeNumerosDossardCoureurs.clear();
        coureurs->setStringList(listeNomsCoureurs);
        ui->listViewCoureurs->setModel(coureurs);
        dossards->setStringList(listeNumerosDossardCoureurs);
        ui->listViewDossards->setModel(dossards);
        return;
    }
}

```

```

    }

    // aucune course ?
    if(listeCourses.size() == 0)
        return;

    // Récupère la course sélectionnée
    QStringList course = listeCourses.at(ui->listeInscriptionCourses->
        currentIndex()-1);
    qDebug() << Q_FUNC_INFO << course;

    // Réinitialise la liste
    disconnect(ui->listeInscriptionCourses, SIGNAL(currentIndexChanged(int)),
        this, SLOT(selectionnerListeCoureursCourse()));
    listeNomsCoureurs.clear();
    listeNumerosDossardCoureurs.clear();
    listeIdCoureurs.clear();

    for(int i=0; i < listeCoureurs.size(); i++)
    {
        listeNomsCoureurs.push_back(QString(listeCoureurs.at(i).at(4) + QString
            (" ") + listeCoureurs.at(i).at(5)));
        listeIdCoureurs.push_back(listeCoureurs.at(i).at(0));
        if(listeCoureurs.count() > 0 && listeCourses.count() > 0)
        {
            if(ui->listeInscriptionCourses->currentIndex() > 0)
            {
                QString numeroDossard = getNumeroDossard(listeCoureurs.at(i).at
                    (0), listeCourses.at(ui->listeInscriptionCourses->currentIndex()-1).at(0));
                //qDebug() << Q_FUNC_INFO << QString::fromUtf8("Coureur : ") <<
                listeCoureurs.at(i) << numeroDossard;
                listeNumerosDossardCoureurs.push_back(numeroDossard);
            }
        }
    }
    connect(ui->listeInscriptionCourses, SIGNAL(currentIndexChanged(int)), this
        , SLOT(selectionnerListeCoureursCourse()));
    afficherListeCoureurs();
}

```

8.5.3.37 IHMManifestation : :selectionnerManifestation () [private, slot]

Sélectionne une manifestation à éditer.

Références [chargerListeCourses\(\)](#), [listeManifestations](#), [manifestation](#), [ONGLET_MANIFESTATION](#), [selectionnerCourse\(\)](#), et [ui](#).

Référéncé par [chargerListeManifestations\(\)](#).

```

{
    QDate maintenant = QDate::currentDate();
    QStringList manifestation;

    int index = ui->listeCreationManifestations->currentIndex();
    switch(index)
    {
    case 0:
        ui->dateEditEditionManifestation->setDate(maintenant);
        ui->lineEditNomManifestation->setText("");
        ui->boutonCreationManifestation->setEnabled(true);
        ui->boutonModificationManifestation->setEnabled(false);
        ui->boutonSuppressionManifestation->setEnabled(false);
        ui->boutonCreationCourse->setEnabled(false);
        ui->listeCreationCourses->setCurrentIndex(0);
        if(ui->onglets->currentIndex() == ONGLET_MANIFESTATION)
            selectionnerCourse();
        break;
    default:
        if(index > 0)
        {

```

```

        manifestation = listeManifestations.at(index-1);
        QDate dateManifestation = QDate::fromString(manifestation.at(2), "
yyyy-MM-dd");
        ui->dateEditEditionManifestation->setDate(dateManifestation);
        ui->lineEditNomManifestation->setText(manifestation.at(1));
        ui->boutonCreationManifestation->setEnabled(false);
        ui->boutonModificationManifestation->setEnabled(true);
        ui->boutonSuppressionManifestation->setEnabled(true);
        ui->boutonCreationCourse->setEnabled(true);
        chargerListeCourses(manifestation.at(0));
    }
    break;
}
}

```

8.5.3.38 IHMManifestation : :selectionnerManifestationResultats () [private, slot]

Références [chargerListeCourses\(\)](#), [listeManifestations](#), [manifestation](#), [selectionnerCourseResultats\(\)](#), et [ui](#).

Référéncé par [chargerListeManifestations\(\)](#).

```

{
    int index = ui->listeCreationManifestationsResultats->currentIndex();

    switch(index)
    {
    case 0:
        ui->listeCreationCoursesResultats->setCurrentIndex(0);
        selectionnerCourseResultats();
        break;
    default:
        if(index > 0)
        {
            manifestation = listeManifestations.at(index-1);
            chargerListeCourses(manifestation.at(0));
        }
        break;
    }
}

```

8.5.3.39 IHMManifestation : :selectionnerOnglet (int index) [private, slot]

Paramètres

<i>index</i>	int l'onglet sélectionné
--------------	--------------------------

Références [afficherListeCoureurs\(\)](#), [chargerListeCategories\(\)](#), [chargerListeClasses\(\)](#), [chargerListeCoureurs\(\)](#), [chargerListeInscriptionCourses\(\)](#), et [editionDossard](#).

Référéncé par [IHMManifestation\(\)](#).

```

{
    switch(index)
    {
    case 0 : // Manifestation
        //chargerListeManifestations();
        break;
    case 1 : // Inscription
        editionDossard = false;
        chargerListeClasses();
        chargerListeCategories();
    }
}

```

```

        chargerListeCoureurs();
        chargerListeInscriptionCourses();
        afficherListeCoureurs();
        break;
    case 2 : // Résultats
        break;
    }
}

```

8.5.3.40 IHMManifestation : :supprimerCoureur() [private, slot]

Références [bd](#), [chargerListeCoureurs\(\)](#), [BaseDeDonnees : :executer\(\)](#), [ui](#), et [verifierInformationsSuppressionCoureur\(\)](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    if(ui->listeCoureurs->currentIndex() == 0)
        return;

    qDebug() << Q_FUNC_INFO ;
    bool retour = false;

    // Vérifications
    if(!verifierInformationsSuppressionCoureur())
    {
        ui->statusBar->showMessage(QString::fromUtf8("Les informations sur le
        coureur sont invalides ou manquantes"), 2000);
        return;
    }

    // Récupère le nom et le prenom du Coureur
    QString Nom = ui->lineEditNomCoureur->text();
    QString Prenom = ui->lineEditPrenomCoureur->text();
    QString requete = "DELETE FROM Coureur WHERE Nom = '" + Nom + "' AND Prenom
        = '" + Prenom + "'";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Suppression du coureur %1
        réussie").arg(ui->lineEditNomCoureur->text()), 2000);
        chargerListeCoureurs();
    }
}

```

8.5.3.41 IHMManifestation : :supprimerCourse() [private, slot]

Références [bd](#), [chargerListeCourses\(\)](#), [BaseDeDonnees : :executer\(\)](#), [listeCourses](#), [listeManifestations](#), [TEMPO_STATUS](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Récupère l'identifiant de la course
    QString idCourse = listeCourses.at(ui->listeCreationCourses->currentIndex()
        -1).at(0);

    QString requete = "DELETE FROM Course WHERE idCourse = '" + idCourse + "'";

    qDebug() << Q_FUNC_INFO << requete;
}

```

```

    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Suppression de la course
        %1 réussie").arg(ui->lineEditNomCourse->text()), TEMPO_STATUS);
        chargerListeCourses(listeManifestations.at(ui->
        listeCreationManifestations->currentIndex()-1).at(0));
    }
}

```

8.5.3.42 IHMManifestation : :supprimerManifestation() [private, slot]

Références [bd](#), [chargerListeManifestations\(\)](#), [BaseDeDonnees : :executer\(\)](#), [listeManifestations](#), et [ui](#).

Référencé par [IHMManifestation\(\)](#).

```

{
    qDebug() << Q_FUNC_INFO;
    bool retour = false;

    // Récupère l'identifiant de la manifestation
    QString idManifestation = listeManifestations.at(ui->
    listeCreationManifestations->currentIndex()-1).at(0);

    QString requete = "DELETE FROM Manifestation WHERE idManifestation = '" +
    idManifestation + "'";

    qDebug() << Q_FUNC_INFO << requete;
    retour = bd->executer(requete);

    if(retour)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Suppression de la
        manifestation %1 réussie").arg(ui->lineEditNomManifestation->text()), 2000);
        chargerListeManifestations();
    }
}

```

8.5.3.43 IHMManifestation : :verifierInformationsCoureur() [private]

Vérifie les informations du coureur.

A faire vérifier si la date de naissance n'est pas dans le futur

Références [ui](#).

```

{
    if(ui->lineEditNomCoureur->text().isEmpty())
    {
        return false;
    }

    if(ui->lineEditPrenomCoureur->text().isEmpty())
    {
        return false;
    }

    //Vérifier numéro INE
    /*if(ui->lineEditNumeroINEEleve->text().isEmpty() ||
    ui->lineEditNumeroINEEleve->text().length() != 11)*/

    if(ui->lineEditNumeroINEEleve->text().contains(QRegExp("
    [0-9]{9}[A-Z]{2}")))
    {

```

```

        return true;
    }

    return false;
}

```

8.5.3.44 IHMManifestation : :verifierInformationsSuppressionCoureur () [private]

Vérifie que les informations du coureur sont supprimées.

Références [ui](#).

Référencé par [supprimerCoureur\(\)](#).

```

{
    if(ui->lineEditNomCoureur->text().isEmpty())
    {
        return false;
    }

    if(ui->lineEditPrenomCoureur->text().isEmpty())
    {
        return false;
    }
    return true;
}

```

8.5.4 Documentation des données membres

8.5.4.1 BaseDeDonnees* IHMManifestation : :bd [private]

Référencé par [afficherResultats\(\)](#), [chargerListeCategories\(\)](#), [chargerListeClasses\(\)](#), [chargerListeCoureurs\(\)](#), [chargerListeCourses\(\)](#), [chargerListeInscriptionCourses\(\)](#), [chargerListeManifestations\(\)](#), [creerCoureur\(\)](#), [creerCourse\(\)](#), [creerManifestation\(\)](#), [getNumeroDossard\(\)](#), [IHMManifestation\(\)](#), [modifierCoureur\(\)](#), [modifierCourse\(\)](#), [modifierManifestation\(\)](#), [supprimerCoureur\(\)](#), [supprimerCourse\(\)](#), [supprimerManifestation\(\)](#), et [~IHMManifestation\(\)](#).

8.5.4.2 QStringListModel* IHMManifestation : :coureurs [private]

Référencé par [afficherListeCoureurs\(\)](#), [IHMManifestation\(\)](#), [inscrireCoureur\(\)](#), [selectionnerDossardListe\(\)](#), et [selectionnerListeCoureursCourse\(\)](#).

8.5.4.3 QStringListModel* IHMManifestation : :dossards [private]

Référencé par [afficherListeCoureurs\(\)](#), [IHMManifestation\(\)](#), [selectionnerCoureurListe\(\)](#), et [selectionnerListeCoureursCourse\(\)](#).

8.5.4.4 bool IHMManifestation : :editionDossard [private]

Référencé par [demarrerInscriptionCoureur\(\)](#), [inscrireCoureur\(\)](#), et [selectionnerOnglet\(\)](#).

8.5.4.5 IHMManifestation : :idCoureur [private]

Référencé par [inscrireCoureur\(\)](#).

8.5.4.6 QVector<QStringList> IHMManifestation : :listeCategories [private]

Référencé par [chargerListeCategories\(\)](#), et [getIdCategorie\(\)](#).

8.5.4.7 QVector<QStringList> IHMManifestation : :listeClasses [private]

Référencé par [chargerListeClasses\(\)](#), et [getIdClasse\(\)](#).

8.5.4.8 QVector<QStringList> IHMManifestation : :listeCoureurs [private]

Référencé par [chargerListeCoureurs\(\)](#), [modifierCoureur\(\)](#), [selectionnerCoureur\(\)](#), [selectionnerCoureurListe\(\)](#), [selectionnerDossardListe\(\)](#), et [selectionnerListeCoureursCourse\(\)](#).

8.5.4.9 QVector<QStringList> IHMManifestation : :listeCourses [private]

Référencé par [chargerListeCourses\(\)](#), [chargerListeInscriptionCourses\(\)](#), [modifierCourse\(\)](#), [selectionnerCourse\(\)](#), [selectionnerCourseResultats\(\)](#), [selectionnerListeCoureursCourse\(\)](#), et [supprimerCourse\(\)](#).

8.5.4.10 QVector<QStringList> IHMManifestation : :listeCoursesResultats [private]**8.5.4.11 QStringList IHMManifestation : :listIdCoureurs [private]**

Référencé par [getIdCoureur\(\)](#), et [selectionnerListeCoureursCourse\(\)](#).

8.5.4.12 QVector<QStringList> IHMManifestation : :listeManifestations [private]

Référencé par [chargerListeManifestations\(\)](#), [creerCourse\(\)](#), [modifierCourse\(\)](#), [modifierManifestation\(\)](#), [selectionnerManifestation\(\)](#), [selectionnerManifestationResultats\(\)](#), [supprimerCourse\(\)](#), et [supprimerManifestation\(\)](#).

8.5.4.13 QVector<QStringList> IHMManifestation : :listeManifestationsResultats [private]**8.5.4.14 QStringList IHMManifestation : :listeNomsCoureurs [private]**

Référencé par [afficherListeCoureurs\(\)](#), [getIdCoureur\(\)](#), [inscrireCoureur\(\)](#), et [selectionnerListeCoureursCourse\(\)](#).

8.5.4.15 QStringList IHMManifestation : :listeNumerosDossardCoureurs [private]

Référencé par [afficherListeCoureurs\(\)](#), [selectionnerCoureurListe\(\)](#), [selectionnerDossardListe\(\)](#), et [selectionnerListeCoureursCourse\(\)](#).

8.5.4.16 QStringList IHMManifestation : :manifestation [private]

Référencé par [chargerListeManifestations\(\)](#), [imprimerResultats\(\)](#), [selectionnerManifestation\(\)](#), et [selectionnerManifestationResultats\(\)](#).

8.5.4.17 `Ui : :IHManifestation* IHManifestation : :ui` `[private]`

Référencé par `activerImpression()`, `afficherConfigurationManifestation()`, `afficherInscriptionCoureurs()`, `afficherListeCoureurs()`, `afficherPublicationResultats()`, `ajouterResultat()`, `chargerListeCategories()`, `chargerListeClasses()`, `chargerListeCoureurs()`, `chargerListeCourses()`, `chargerListeInscriptionCourses()`, `chargerListeManifestations()`, `configurerTableResultats()`, `creerCoureur()`, `creerCourse()`, `creerManifestation()`, `effacerResultats()`, `getIdCategorie()`, `getIdClasse()`, `IHManifestation()`, `imprimerResultats()`, `inscrireCoureur()`, `modifierCoureur()`, `modifierCourse()`, `modifierManifestation()`, `selectionnerCoureur()`, `selectionnerCoureurListe()`, `selectionnerCourse()`, `selectionnerCourseResultats()`, `selectionnerDossardListe()`, `selectionnerListeCoureursCourse()`, `selectionnerManifestation()`, `selectionnerManifestationResultats()`, `supprimerCoureur()`, `supprimerCourse()`, `supprimerManifestation()`, `verifierInformationsCoureur()`, `verifierInformationsSuppressionCoureur()`, et `~IHManifestation()`.

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

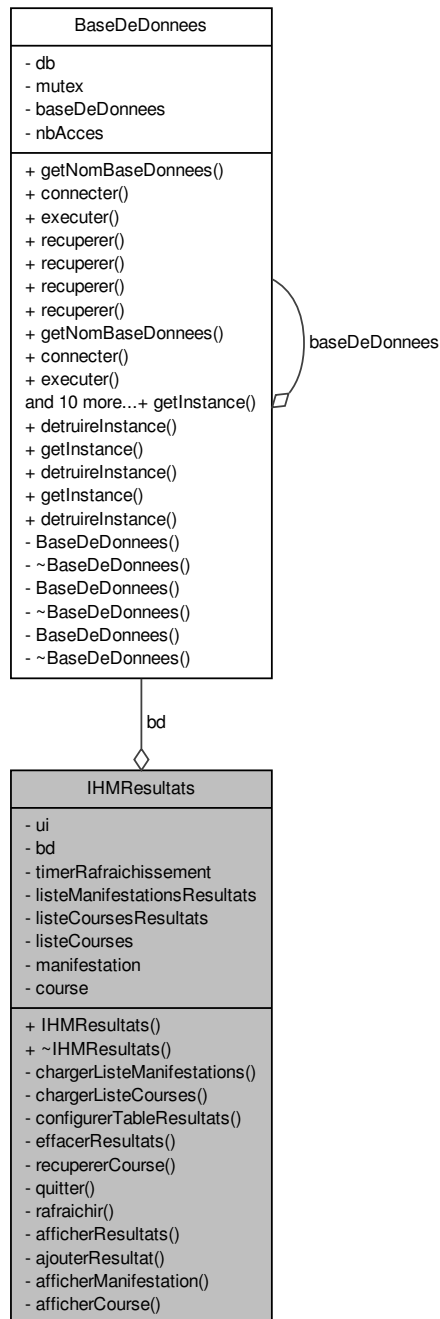
- `ihmanifestation.h`
- `chronocrossclassement.cpp`
- `editionmanifestation.cpp`
- `ihmanifestation.cpp`

8.6 Référence de la classe IHMResultats

La fenêtre principale de l'application.

```
#include <ihmresultats.h>
```

Grphe de collaboration de IHMResultats :



Fonctions membres publiques

- [IHMResultats](#) (QWidget *parent=0)
Constructeur de la classe [IHMResultats](#).
- [~IHMResultats](#) ()
Destructeur de la classe [IHMResultats](#).

Connecteurs privés

- void [quitter](#) ()
Ferme l'application principale.
- void [rafraichir](#) ()
Rafraichit l'IHM.
- void [afficherResultats](#) (QString idCourse)
Affiche les résultats de la course.
- void [ajouterResultat](#) (int ligne, QStringList unResultat)
Ajoute les résultats d'une course dans un tableau.
- void [afficherManifestation](#) ()
Affiche les paramètres de la manifestation.
- void [afficherCourse](#) ()
Affiche les paramètres de la course.

Fonctions membres privées

- void [chargerListeManifestations](#) ()
Charge la liste des manifestations.
- void [chargerListeCourses](#) (QString idManifestation)
Charge la liste des courses pour une manifestation.
- void [configurerTableResultats](#) ()
Configure le tableau de résultats.
- void [effacerResultats](#) ()
Supprime les résultats d'une course dans un tableau.
- void [recupererCourse](#) ()
Recupère les données de la course.

Attributs privés

- Ui : :IHMResultats * [ui](#)
relation vers la classe IHM
- [BaseDeDonnees](#) * [bd](#)
association vers la classe [BaseDeDonnees](#)
- QTimer * [timerRafraichissement](#)
pour le rafraichissement de l'IHM
- QVector< QStringList > [listeManifestationsResultats](#)
la liste des manifestations créée dans la base de données dans l'onglet résultats
- QVector< QStringList > [listeCoursesResultats](#)
la liste des courses pour une manifestation dans l'onglet résultats
- QVector< QStringList > [listeCourses](#)
la liste des courses pour une manifestation
- QStringList [manifestation](#)
la manifestation pour l'affichage des résultats
- QStringList [course](#)
la course pour l'affichage des résultats

8.6.1 Description détaillée

Auteur

BULIN Julien

Version

1.1

8.6.2 Documentation des constructeurs et destructeur

8.6.2.1 IHMResultats : :IHMResultats (QWidget * parent = 0) [explicit]

Paramètres

<i>parent</i>	
---------------	--

Références [afficherManifestation\(\)](#), [bd](#), [configurerTableResultats\(\)](#), [BaseDeDonnees : :connecter\(\)](#), [BaseDeDonnees : :getInstance\(\)](#), [BaseDeDonnees : :getNomBaseDonnees\(\)](#), [rafraichir\(\)](#), [timerRafraichissement](#), et [ui](#).

```

                                :
    QMainWindow(parent),
    ui(new Ui::IHMResultats)
{
    ui->setupUi(this);

    // Connexion à la base de données
    bd = BaseDeDonnees::getInstance();
    bool connexion = bd->connecter();
    if(!connexion)
    {
        ui->statusBar->showMessage(QString::fromUtf8("Impossible de se
        connecter à la base de données %1").arg(bd->getNomBaseDonnees()));
    }

    timerRafraichissement = new QTimer(this);
    connect(timerRafraichissement, SIGNAL(timeout()), this, SLOT(rafraichir()))
        ;

    configurerTableResultats();
    afficherManifestation();

    showFullScreen();

    timerRafraichissement->start(1000);
}

```

8.6.2.2 IHMResultats : :~IHMResultats ()

Références [bd](#), [BaseDeDonnees : :destruireInstance\(\)](#), et [ui](#).

```

{
    bd->destruireInstance();
    delete ui;
}

```

8.6.3 Documentation des fonctions membres

8.6.3.1 IHMResultats : :afficherCourse () [private, slot]

A faire afficher les paramètres de la course

```
{
}
```

8.6.3.2 IHMResultats : :afficherManifestation () [private, slot]

Références [bd](#), [manifestation](#), [MANIFESTATION_DATE](#), [MANIFESTATION_NOM](#), - [BaseDeDonnees : :recuperer\(\)](#), [recupererCourse\(\)](#), et [ui](#).

Référencé par [IHMResultats\(\)](#), et [rafraichir\(\)](#).

```
{
    QString requete = "SELECT Manifestation.idManifestation, Manifestation.Nom,
        Manifestation.Date FROM Manifestation WHERE Manifestation.Date = date(now());";
    manifestation.clear();
    bool retour = bd->recuperer(requete, manifestation);
    //qDebug() << Q_FUNC_INFO << manifestation << retour;
    if(retour != false)
    {
        ui->labelDateHeure->setText(manifestation.at(MANIFESTATION_NOM) + "
            " + manifestation.at(MANIFESTATION_DATE));
        recupererCourse();
    }
    else
        ui->labelDateHeure->setText("Aucune manifestation !");
}
```

8.6.3.3 IHMResultats : :afficherResultats (QString idCourse) [private, slot]

Paramètres

<i>idCourse</i>	QString l'identifiant de la course
-----------------	------------------------------------

Références [ajouterResultat\(\)](#), [bd](#), [effacerResultats\(\)](#), et [BaseDeDonnees : :recuperer\(\)](#).

Référencé par [rafraichir\(\)](#).

```
{
    // On commence par effacer le contenu précédent
    effacerResultats();

    // Récupérer les résultats d'une course (Requête SQL)
    QVector<QStringList> resultats;
    QString requete = "SELECT Inscrit.NumeroDossard, Coureur.Nom,
        Coureur.Prenom, Classe.Nom AS NomClasse, Classe.Numero, Arrivee.Temps FROM Course, Inscrit,
        Coureur, Classe, Arrivee WHERE Course.IdCourse = Inscrit.idCourse AND
        Inscrit.idCoureur = Coureur.idCoureur AND Coureur.idClasse = Classe.idClasse AND
        Inscrit.idInscrit = Arrivee.idInscrit AND Course.IdCourse = ' " + idCourse + "' ORDER BY
        Arrivee.Temps ASC;";
    qDebug() << Q_FUNC_INFO << requete;
    bool retour = bd->recuperer(requete, resultats);
    if(retour != false)
    {
        QStringList unResultat;

        for(int ligne = 0; ligne < resultats.size(); ligne++)
        {
            unResultat = resultats.at(ligne);
        }
    }
}
```

```

        ajouterResultat(ligne, unResultat);
    }
}

```

8.6.3.4 IHMResultats :ajouterResultat (int ligne, QStringList unResultat) [private, slot]

Références [COLONNE_CLASSE](#), [COLONNE_CLASSEMENT](#), [COLONNE_NOM](#), [COLONNE_NUMERODOSSARD](#), [COLONNE_PRENOM](#), [COLONNE_TEMPS](#), [EMPLACEMENT_NOM](#), [EMPLACEMENT_NOMCLASSE](#), [EMPLACEMENT_NUMEROCLASSE](#), [EMPLACEMENT_NUMERODOSSARD](#), [EMPLACEMENT_PRENOM](#), [EMPLACEMENT_TEMPS](#), et [ui](#).

Référencé par [afficherResultats\(\)](#).

```

{
    QTableWidgetItem *elementClassement, *elementNumeroDossard, *elementNom, *
    elementPrenom, *elementClasse, *elementTemps;

    // Taille de la police en gras
    QFont font("", 20, QFont::Normal);
    font.setBold(true);

    //Colonne Classement
    ui->tableWidgetResultats->insertRow(ligne);
    elementClassement = new QTableWidgetItem(QString::number(ligne+1));
    elementClassement->setBackgroundColor(QColor(255,255,255));
    elementClassement->setForeground(Qt::black);
    elementClassement->setFont(font);
    elementClassement->setFlags(Qt::ItemIsEnabled);
    if(ligne < 1)
    {
        elementClassement->setBackgroundColor(QColor(255,236,139));
    }
    else if(ligne < 2)
    {
        elementClassement->setBackgroundColor(QColor(220,220,220));
    }
    else if(ligne < 3)
    {
        elementClassement->setBackgroundColor(QColor(222,184,135));
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_CLASSEMENT,
        elementClassement); // l : ligne (row) et c : colonne

    //Colonne Numéro de Dossard
    elementNumeroDossard = new QTableWidgetItem(unResultat.at(
        EMPLACEMENT_NUMERODOSSARD));
    elementNumeroDossard->setBackgroundColor(QColor(255,255,255));
    elementNumeroDossard->setForeground(Qt::black);
    elementNumeroDossard->setFont(font);
    elementNumeroDossard->setFlags(Qt::ItemIsEnabled);
    if(ligne < 1)
    {
        elementNumeroDossard->setBackgroundColor(QColor(255,236,139));
    }
    else if(ligne < 2)
    {
        elementNumeroDossard->setBackgroundColor(QColor(220,220,220));
    }
    else if(ligne < 3)
    {
        elementNumeroDossard->setBackgroundColor(QColor(222,184,135));
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_NUMERODOSSARD,
        elementNumeroDossard); // l : ligne (row) et c : colonne
}

```

```

//Colonne Nom
elementNom = new QTableWidgetItem(unResultat.at(EMPLACEMENT_NOM));
elementNom->setBackgroundColor(QColor(255,255,255));
elementNom->setForeground(Qt::black);
elementNom->setFont(font);
elementNom->setFlags(Qt::ItemIsEnabled);
if(ligne < 1)
{
    elementNom->setBackgroundColor(QColor(255,236,139));
}
else if(ligne < 2)
{
    elementNom->setBackgroundColor(QColor(220,220,220));
}
else if(ligne < 3)
{
    elementNom->setBackgroundColor(QColor(222,184,135));
}
ui->tableWidgetResultats->setItem(ligne, COLONNE_NOM, elementNom); // l :
    ligne (row) et c : colonne

//Colonne Prénom
elementPrenom = new QTableWidgetItem(unResultat.at(EMPLACEMENT_PRENOM));
elementPrenom->setBackgroundColor(QColor(255,255,255));
elementPrenom->setForeground(Qt::black);
elementPrenom->setFont(font);
elementPrenom->setFlags(Qt::ItemIsEnabled);
if(ligne < 1)
{
    elementPrenom->setBackgroundColor(QColor(255,236,139));
}
else if(ligne < 2)
{
    elementPrenom->setBackgroundColor(QColor(220,220,220));
}
else if(ligne < 3)
{
    elementPrenom->setBackgroundColor(QColor(222,184,135));
}
ui->tableWidgetResultats->setItem(ligne, COLONNE_PRENOM, elementPrenom); //
    l : ligne (row) et c : colonne

//Colonne Classe
elementClasse = new QTableWidgetItem(unResultat.at(EMPLACEMENT_NOMCLASSE) +
    " " + unResultat.at(EMPLACEMENT_NUMEROCLASSE));
elementClasse->setBackgroundColor(QColor(255,255,255));
elementClasse->setForeground(Qt::black);
elementClasse->setFont(font);
elementClasse->setFlags(Qt::ItemIsEnabled);
elementClasse->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
if(ligne < 1)
{
    elementClasse->setBackgroundColor(QColor(255,236,139));
}
else if(ligne < 2)
{
    elementClasse->setBackgroundColor(QColor(220,220,220));
}
else if(ligne < 3)
{
    elementClasse->setBackgroundColor(QColor(222,184,135));
}
ui->tableWidgetResultats->setItem(ligne, COLONNE_CLASSE, elementClasse); //
    l : ligne (row) et c : colonne

//Colonne Temps
elementTemps = new QTableWidgetItem(unResultat.at(EMPLACEMENT_TEMPS));
elementTemps->setBackgroundColor(QColor(255,255,255));
elementTemps->setForeground(Qt::black);
elementTemps->setFont(font);
elementTemps->setFlags(Qt::ItemIsEnabled);
elementTemps->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
if(ligne < 1)
{

```

```

        elementTemps->setBackgroundColor(QColor(255,236,139));
    }
    else if(ligne < 2)
    {
        elementTemps->setBackgroundColor(QColor(220,220,220));
    }
    else if(ligne < 3)
    {
        elementTemps->setBackgroundColor(QColor(222,184,135));
    }
    ui->tableWidgetResultats->setItem(ligne, COLONNE_TEMPS, elementTemps); // 1
        : ligne (row) et c : colonne
}

```

8.6.3.5 IHMResultats : :chargerListeCourses (QString *idManifestation*) [private]

Paramètres

<i>id- Manifestation</i>	QString l'identifiant de la manifestation
------------------------------	---

8.6.3.6 IHMResultats : :chargerListeManifestations () [private]

8.6.3.7 IHMResultats : :configurerTableResultats () [private]

Références [ui](#).

Référencé par [IHMResultats\(\)](#).

```

{
    QStringList nomColonnes; // nom des colonnes
    nomColonnes << QString::fromUtf8("Classement") << QString::fromUtf8("Numéro  
de dossard") << QString::fromUtf8("Nom") << QString::fromUtf8("Prénom") <<  
    QString::fromUtf8("Classe") << QString::fromUtf8("Temps"); // ...

    // On fixe le nombre de colonnes
    ui->tableWidgetResultats->setColumnCount(nomColonnes.size());

    // On applique les noms des colonnes
    ui->tableWidgetResultats->setHorizontalHeaderLabels(nomColonnes);

    // on cache les numéros de ligne
    ui->tableWidgetResultats->verticalHeader()->setHidden(true);

    QHeaderView * headerView = ui->tableWidgetResultats->horizontalHeader();

    // on redimensionne automatiquement la colonne pour occuper l'espace  
    disponible
    headerView->setResizeMode(QHeaderView::Stretch);
}

```

8.6.3.8 IHMResultats : :effacerResultats () [private]

Références [ui](#).

Référencé par [afficherResultats\(\)](#).

```

{
    int nb = ui->tableWidgetResultats->rowCount();
    // on efface les lignes du tableau une par une
    for(int i = 0; i < nb; i++)

```

```

    {
        ui->tableWidgetResultats->removeRow(0);
    }
}

```

8.6.3.9 IHMResultats : :quitter() [private, slot]

Quitte l'application.

```

{
    // on ferme la fenêtre
    close();
}

```

8.6.3.10 IHMResultats : :rafraichir() [private, slot]

Références [afficherManifestation\(\)](#), [afficherResultats\(\)](#), [course](#), et [COURSE_ID_COURSE](#).

Référencé par [IHMResultats\(\)](#).

```

{
    //qDebug() << Q_FUNC_INFO;
    afficherManifestation();

    // Récupérer et affiche les résultats d'une course (Requête SQL)
    if(!course.isEmpty())
    {
        QString idCourse = course.at(COURSE_ID_COURSE);
        afficherResultats(idCourse);
    }
}

```

8.6.3.11 IHMResultats : :recupererCourse() [private]

Références [bd](#), [course](#), [COURSE_DISTANCE](#), [COURSE_HEUREDEPART](#), [COURSE_NOM](#), [manifestation](#), [MANIFESTATION_ID](#), [BaseDeDonnees : :recuperer\(\)](#), et [ui](#).

Référencé par [afficherManifestation\(\)](#).

```

{
    // Récupérer la course la plus proche
    QString requete = "SELECT Course.* FROM Manifestation, Course WHERE
        Manifestation.idManifestation = " + manifestation.at(MANIFESTATION_ID) + " AND
        Manifestation.idManifestation = Course.idManifestation AND Manifestation.Date =
        date(now()) ORDER BY ABS(time(now())-Course.HeureDepart) ASC LIMIT 1;";
    course.clear();
    bool retour = bd->recuperer(requete, course);
    qDebug() << Q_FUNC_INFO << course << retour;
    if(retour != false)
    {
        QTime heureDepart = QTime::fromString(course.at(COURSE_HEUREDEPART), "
        hh:mm:ss");
        ui->label_NomCourseResultats->setText(QString::fromUtf8("Course : ") +
        course.at(COURSE_NOM));
        ui->label_HeureDepartCourseResultats->setText(QString::fromUtf8("Heure
        de départ : ") + heureDepart.toString("hh'h'mm"));
        ui->label_LongueurCourseResultats->setText(QString::fromUtf8("Longueur
        : ") + course.at(COURSE_DISTANCE) + " m");
    }
    else
    {

```

```
    ui->label_NomCourseResultats->setText("Course : ");  
    ui->label_HeureDepartCourseResultats->setText(QString::fromUtf8("Heure  
de départ : "));  
    ui->label_LongueurCourseResultats->setText("Longueur : ");  
}  
}
```

8.6.4 Documentation des données membres

8.6.4.1 BaseDeDonnees* IHMResultats : :bd [private]

Référencé par [afficherManifestation\(\)](#), [afficherResultats\(\)](#), [IHMResultats\(\)](#), [recupererCourse\(\)](#), et [~IHMResultats\(\)](#).

8.6.4.2 QStringList IHMResultats : :course [private]

Référencé par [rafraichir\(\)](#), et [recupererCourse\(\)](#).

8.6.4.3 QVector<QStringList> IHMResultats : :listeCourses [private]

8.6.4.4 QVector<QStringList> IHMResultats : :listeCoursesResultats [private]

8.6.4.5 QVector<QStringList> IHMResultats : :listeManifestationsResultats [private]

8.6.4.6 QStringList IHMResultats : :manifestation [private]

Référencé par [afficherManifestation\(\)](#), et [recupererCourse\(\)](#).

8.6.4.7 QTimer* IHMResultats : :timerRafraichissement [private]

Référencé par [IHMResultats\(\)](#).

8.6.4.8 Ui : :IHMResultats* IHMResultats : :ui [private]

Référencé par [afficherManifestation\(\)](#), [ajouterResultat\(\)](#), [configurerTableResultats\(\)](#), [effacerResultats\(\)](#), [IHMResultats\(\)](#), [recupererCourse\(\)](#), et [~IHMResultats\(\)](#).

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

- [ihmresultats.h](#)
- [ihmresultats.cpp](#)

9 Documentation des fichiers

9.1 Référence du fichier basededonnees.cpp

```
#include "basededonnees.h" #include <QDebug> #include <Q-  
MessageBox>
```

9.2 Référence du fichier basededonnees.cpp

```
#include "basededonnees.h" #include <QDebug> #include <Q-  
MessageBox>
```

9.3 Référence du fichier basededonnees.cpp

```
#include "basededonnees.h" #include <QDebug> #include <Q-  
MessageBox>
```

9.4 Référence du fichier basededonnees.h

```
#include <QObject> #include <QtSql/QtSql> #include <QSql-  
Database> #include <QMutex>
```

Classes

- class [BaseDeDonnees](#)
Déclaration de la classe utilisant la base de données.

Macros

- #define [HOSTNAME](#) "192.168.52.119"
- #define [USERNAME](#) "root"
- #define [PASSWORD](#) "password"
- #define [DATABASENAME](#) "Chrono-cross"

9.4.1 Documentation des macros

9.4.1.1 #define [DATABASENAME](#) "Chrono-cross"

9.4.1.2 #define [HOSTNAME](#) "192.168.52.119"

9.4.1.3 #define [PASSWORD](#) "password"

Référencé par [BaseDeDonnees](#) : `:connecter()`.

9.4.1.4 #define [USERNAME](#) "root"

Référencé par [BaseDeDonnees](#) : `:connecter()`.

9.5 Référence du fichier basededonnees.h

```
#include <QObject> #include <QtSql/QtSql> #include <QSql-  
Database> #include <QMutex>
```

Classes

- class [BaseDeDonnees](#)
Déclaration de la classe utilisant la base de données.

Macros

- #define [HOSTNAME](#) "192.168.52.119"
- #define [USERNAME](#) "root"
- #define [PASSWORD](#) "password"
- #define [DATABASENAME](#) "Chrono-cross"

9.5.1 Documentation des macros

9.5.1.1 #define DATABASENAME "Chrono-cross"

9.5.1.2 #define HOSTNAME "192.168.52.119"

9.5.1.3 #define PASSWORD "password"

9.5.1.4 #define USERNAME "root"

9.6 Référence du fichier basededonnees.h

```
#include <QObject> #include <QtSql/QtSql> #include <QtSql-  
Database> #include <QMutex>
```

Classes

- class [BaseDeDonnees](#)
Déclaration de la classe utilisant la base de données.

Macros

- #define [HOSTNAME](#) "192.168.52.119"
- #define [USERNAME](#) "root"
- #define [PASSWORD](#) "password"
- #define [DATABASENAME](#) "Chrono-cross"

9.6.1 Documentation des macros

9.6.1.1 #define DATABASENAME "Chrono-cross"

9.6.1.2 #define HOSTNAME "192.168.52.119"

9.6.1.3 #define PASSWORD "password"

9.6.1.4 #define USERNAME "root"

9.7 Référence du fichier Changelog.dox

9.8 Référence du fichier chronocrossclassement.cpp

Définition de la classe [ChronoCrossClassement](#).

```
#include "chronocrossclassement.h" #include "ui_chronocrossclassement.-  
h" #include "basededonnees.h" #include "unistd.h" #include  
<QMessageBox> #include <QWidget> #include <QDebug>
```

9.8.1 Description détaillée

Auteur

PELLIZZONI Corentin

Version

1.0

9.9 Référence du fichier chronocrossclassement.h

Déclaration de la classe [ChronoCrossClassement](#).

```
#include <QMainWindow> #include "qextserialenumerator.h" ×  
#include "qextserialport.h"
```

Classes

– class [ChronoCrossClassement](#)

Macros

```
– #define NOM_PORT QString("/dev/ttyUSB0")  
– #define TIMEOUT 1500  
– #define PERIODE 1000  
– #define COURSE_ID 0  
– #define COURSE_NOM 2  
– #define COURSE_DISTANCE 3  
– #define COURSE_HEUREDEPART 4  
– #define AK "AK"  
– #define ID "ID"  
– #define SN "SN"  
– #define OP "OP"  
– #define CL "CL"  
– #define DS "DS"  
– #define DE "DE"  
– #define VE "VE"  
– #define RR "RR"  
– #define SYNCHRO "IT"  
– #define PARAMETER "&P"  
– #define SYSTEM_EVENT "&S"  
– #define DISPLAY_EVENT "&D"
```

9.9.1 Description détaillée

Auteur

PELLIZZONI Corentin

Version

1.0

9.9.2 Documentation des macros

9.9.2.1 `#define AK "AK"`

9.9.2.2 `#define CL "CL"`

9.9.2.3 `#define COURSE_DISTANCE 3`

Référencé par `ChronoCrossClassement` : `:chargerListeCourses()`, `IHMManifestation` : `:chargerListeCourses()`, `IHMResultats` : `:recupererCourse()`, `IHMManifestation` : `:selectionnerCourse()`, et `IHMManifestation` : `:selectionnerCourseResultats()`.

9.9.2.4 `#define COURSE_HEUREDEPART 4`

Référencé par `ChronoCrossClassement` : `:chargerListeCourses()`, `IHMManifestation` : `:chargerListeCourses()`, `IHMResultats` : `:recupererCourse()`, `IHMManifestation` : `:selectionnerCourse()`, et `IHMManifestation` : `:selectionnerCourseResultats()`.

9.9.2.5 `#define COURSE_ID 0`

Référencé par `ChronoCrossClassement` : `:chargerInscritsCourse()`.

9.9.2.6 `#define COURSE_NOM 2`

Référencé par `ChronoCrossClassement` : `:chargerListeCourses()`, `IHMManifestation` : `:chargerListeCourses()`, `IHMManifestation` : `:chargerListeInscriptionCourses()`, `IHMResultats` : `:recupererCourse()`, `IHMManifestation` : `:selectionnerCourse()`, et `IHMManifestation` : `:selectionnerCourseResultats()`.

9.9.2.7 `#define DE "DE"`

9.9.2.8 `#define DISPLAY_EVENT "&D"`

9.9.2.9 `#define DS "DS"`

9.9.2.10 `#define ID "ID"`

9.9.2.11 `#define NOM_PORT QString("/dev/ttyUSB0")`

Référencé par `ChronoCrossClassement` : `:initialiserPort()`.

9.9.2.12 `#define OP "OP"`

9.9.2.13 `#define PARAMETER "&P"`

9.9.2.14 `#define PERIODE 1000`

9.9.2.15 `#define RR "RR"`

9.9.2.16 `#define SN "SN"`

9.9.2.17 `#define SYNCHRO " !T"`

9.9.2.18 `#define SYSTEM_EVENT "&S"`

9.9.2.19 `#define TIMEOUT 1500`

9.9.2.20 `#define VE "VE"`

9.10 Référence du fichier editioncourse.cpp

Définition de la classe [EditionCourse](#).

```
#include "editioncourse.h" #include "ui_ihmmanifestation.-  
h" #include "basededonnees.h" #include "ihmmanifestation.-  
h"
```

9.10.1 Description détaillée

Auteur

BULIN Julien

Version

0.9

9.11 Référence du fichier editioncourse.h

```
#include <QMainWindow> #include <QStringListModel> ×  
#include <QDebug> #include <QObject>
```

Classes

- class [EditionCourse](#)
La fenêtre d'édition de la course.

9.11.1 Description détaillée

9.12 Référence du fichier editionmanifestation.cpp

Définition de la classe [EditionManifestation](#).

```
#include "editionmanifestation.h" #include "ui_ihmmanifestation.-  
h" #include "basedonnees.h" #include "ihmmanifestation.-  
h"
```

9.12.1 Description détaillée

Auteur

BULIN Julien

Version

0.9

9.13 Référence du fichier editionmanifestation.h

```
#include <QMainWindow> #include <QStringListModel> ×  
#include <QDebug> #include <QObject>
```

Classes

- class [EditionManifestation](#)
La fenêtre d'édition de la manifestation.

9.13.1 Description détaillée

9.14 Référence du fichier ihmmanifestation.cpp

Définition de la classe [IHManifestation](#).

```
#include "ihmanifestation.h" #include "ui_ihmmanifestation.-  
h" #include "basedonnees.h" #include <QMessageBox> ×  
#include <QColor> #include <QPrinter> #include <QPrint-  
Dialog> #include <QPainter>
```

9.14.1 Description détaillée

Auteur

BULIN Julien, PELLIZZONI Corentin

Version

1.0

9.15 Référence du fichier ihmmanifestation.h

```
#include <QMainWindow> #include <QStringListModel> ×  
#include <QDebug> #include <QTableView>
```

Classes

- class [IHManifestation](#)
La fenêtre principale de l'application.

Macros

- #define [MANIFESTATION_NOM](#) 1
- #define [MANIFESTATION_DATE](#) 2
- #define [COURSE_ID_COURSE](#) 0
- #define [COURSE_IDMANIFESTATION](#) 1
- #define [COURSE_NOM](#) 2
- #define [COURSE_DISTANCE](#) 3
- #define [COURSE_HEUREDEPART](#) 4
- #define [COUREUR_ID](#) 0
- #define [COUREUR_IDCATEGORIE](#) 1
- #define [COUREUR_IDCLASSE](#) 2
- #define [COUREUR_NUMERO_INE](#) 3
- #define [COUREUR_NOM](#) 4
- #define [COUREUR_PRENOM](#) 5
- #define [COUREUR_DATENAISSANCE](#) 6
- #define [COUREUR_SEXE](#) 7
- #define [COUREUR_NOM_CLASSE](#) 8
- #define [COUREUR_NUMERO_CLASSE](#) 9
- #define [COUREUR_NOM_CATEGORIE](#) 10
- #define [COUREUR_SEXE_CATEGORIE](#) 11
- #define [TEMPO_STATUS](#) 5000
- #define [ONGLET_MANIFESTATION](#) 0
- #define [ONGLET_INSCRIPTION](#) 1
- #define [ONGLET_RESULTATS](#) 2
- #define [COLONNE_CLASSEMENT](#) 0
- #define [COLONNE_NUMERODOSSARD](#) 1
- #define [COLONNE_NOM](#) 2
- #define [COLONNE_PRENOM](#) 3
- #define [COLONNE_CLASSE](#) 4
- #define [COLONNE_TEMPS](#) 5
- #define [EMPLACEMENT_NUMERODOSSARD](#) 0
- #define [EMPLACEMENT_NOM](#) 1
- #define [EMPLACEMENT_PRENOM](#) 2
- #define [EMPLACEMENT_NOMCLASSE](#) 3
- #define [EMPLACEMENT_NUMEROCLASSE](#) 4
- #define [EMPLACEMENT_TEMPS](#) 5

9.15.1 Description détaillée

9.15.2 Documentation des macros

9.15.2.1 #define COLONNE_CLASSE 4

Référéncé par [IHMRésultats](#) : [:ajouterResultat\(\)](#), et [IHManifestation](#) : [:ajouter-Resultat\(\)](#).

9.15.2.2 #define COLONNE_CLASSEMENT 0

Référéncé par [IHMRésultats](#) : [:ajouterResultat\(\)](#), et [IHManifestation](#) : [:ajouter-Resultat\(\)](#).

9.15.2.3 #define COLONNE_NOM 2

Référencé par [IHMResultats : :ajouterResultat\(\)](#), et [IHManifestation : :ajouter-Resultat\(\)](#).

9.15.2.4 #define COLONNE_NUMERODOSSARD 1

Référencé par [IHMResultats : :ajouterResultat\(\)](#), et [IHManifestation : :ajouter-Resultat\(\)](#).

9.15.2.5 #define COLONNE_PRENOM 3

Référencé par [IHMResultats : :ajouterResultat\(\)](#), et [IHManifestation : :ajouter-Resultat\(\)](#).

9.15.2.6 #define COLONNE_TEMPS 5

Référencé par [IHMResultats : :ajouterResultat\(\)](#), et [IHManifestation : :ajouter-Resultat\(\)](#).

9.15.2.7 #define COUREUR_DATENAissance 6

Référencé par [IHManifestation : :selectionnerCoureur\(\)](#).

9.15.2.8 #define COUREUR_ID 0

Référencé par [IHManifestation : :modifierCoureur\(\)](#).

9.15.2.9 #define COUREUR_IDCATEGORIE 1**9.15.2.10 #define COUREUR_IDCLASSE 2****9.15.2.11 #define COUREUR_NOM 4**

Référencé par [IHManifestation : :selectionnerCoureur\(\)](#).

9.15.2.12 #define COUREUR_NOM_CATEGORIE 10

Référencé par [IHManifestation : :selectionnerCoureur\(\)](#).

9.15.2.13 #define COUREUR_NOM_CLASSE 8

Référencé par [IHManifestation : :selectionnerCoureur\(\)](#).

9.15.2.14 #define COUREUR_NUMERO_CLASSE 9

Référencé par [IHManifestation : :selectionnerCoureur\(\)](#).

9.15.2.15 #define COUREUR_NUMERO_INE 3

Référencé par [IHManifestation : :selectionnerCoureur\(\)](#).

9.15.2.16 `#define COUREUR_PRENOM 5`

Référencé par `IHMManifestation : :selectionnerCoureur()`.

9.15.2.17 `#define COUREUR_SEXE 7`

Référencé par `IHMManifestation : :selectionnerCoureur()`.

9.15.2.18 `#define COUREUR_SEXE_CATEGORIE 11`

Référencé par `IHMManifestation : :selectionnerCoureur()`.

9.15.2.19 `#define COURSE_DISTANCE 3`

9.15.2.20 `#define COURSE_HEUREDEPART 4`

9.15.2.21 `#define COURSE_ID_COURSE 0`

Référencé par `IHMResultats : :rafraichir()`.

9.15.2.22 `#define COURSE_IDMANIFESTATION 1`

9.15.2.23 `#define COURSE_NOM 2`

9.15.2.24 `#define EMPLACEMENT_NOM 1`

Référencé par `IHMResultats : :ajouterResultat()`, et `IHMManifestation : :ajouter-Resultat()`.

9.15.2.25 `#define EMPLACEMENT_NOMCLASSE 3`

Référencé par `IHMResultats : :ajouterResultat()`, et `IHMManifestation : :ajouter-Resultat()`.

9.15.2.26 `#define EMPLACEMENT_NUMEROCLASSE 4`

Référencé par `IHMResultats : :ajouterResultat()`, et `IHMManifestation : :ajouter-Resultat()`.

9.15.2.27 `#define EMPLACEMENT_NUMERODOSSARD 0`

Référencé par `IHMResultats : :ajouterResultat()`, et `IHMManifestation : :ajouter-Resultat()`.

9.15.2.28 `#define EMPLACEMENT_PRENOM 2`

Référencé par `IHMResultats : :ajouterResultat()`, et `IHMManifestation : :ajouter-Resultat()`.

9.15.2.29 `#define EMPLACEMENT_TEMPS 5`

Référencé par `IHMResultats : :ajouterResultat()`, et `IHMManifestation : :ajouter-Resultat()`.

9.15.2.30 #define MANIFESTATION_DATE 2

Référencé par `IHMResultats : :afficherManifestation()`, et `IHMManifestation : :imprimerResultats()`.

9.15.2.31 #define MANIFESTATION_NOM 1

Référencé par `IHMResultats : :afficherManifestation()`, et `IHMManifestation : :imprimerResultats()`.

9.15.2.32 #define ONGLET_INSCRIPTION 1

Référencé par `IHMManifestation : :afficherInscriptionCoureurs()`.

9.15.2.33 #define ONGLET_MANIFESTATION 0

Référencé par `IHMManifestation : :afficherConfigurationManifestation()`, `IHMManifestation : :chargerListeCourses()`, `IHMManifestation : :chargerListeManifestations()`, et `IHMManifestation : :selectionnerManifestation()`.

9.15.2.34 #define ONGLET_RESULTATS 2

Référencé par `IHMManifestation : :afficherPublicationResultats()`.

9.15.2.35 #define TEMPO_STATUS 5000

Référencé par `IHMManifestation : :creerCourse()`, `IHMManifestation : :IHMManifestation()`, `IHMManifestation : :modifierCoureur()`, `IHMManifestation : :modifierCourse()`, et `IHMManifestation : :supprimerCourse()`.

9.16 Référence du fichier ihmresultats.cpp

Définition de la classe `IHMResultats`.

```
#include "ihmresultats.h"    #include "ui_ihmresultats.h" ×
#include "basededonnees.h" #include <QMessageBox> #include
<QColor>
```

9.16.1 Description détaillée**Auteur**

BULIN Julien

Version

1.1

9.17 Référence du fichier ihmresultats.h

```
#include <QMainWindow> #include <QDebug>
```

Classes

- class `IHMResultats`
La fenêtre principale de l'application.

Macros

- #define `COURSE_ID_COURSE` 0
- #define `COURSE_IDMANIFESTATION` 1
- #define `COURSE_NOM` 2
- #define `COURSE_DISTANCE` 3
- #define `COURSE_HEUREDEPART` 4
- #define `MANIFESTATION_ID` 0
- #define `MANIFESTATION_NOM` 1
- #define `MANIFESTATION_DATE` 2
- #define `TEMPO_STATUS` 5000
- #define `COLONNE_CLASSEMENT` 0
- #define `COLONNE_NUMERODOSSARD` 1
- #define `COLONNE_NOM` 2
- #define `COLONNE_PRENOM` 3
- #define `COLONNE_CLASSE` 4
- #define `COLONNE_TEMPS` 5
- #define `EMPLACEMENT_NUMERODOSSARD` 0
- #define `EMPLACEMENT_NOM` 1
- #define `EMPLACEMENT_PRENOM` 2
- #define `EMPLACEMENT_NOMCLASSE` 3
- #define `EMPLACEMENT_NUMEROCLASSE` 4
- #define `EMPLACEMENT_TEMPS` 5

9.17.1 Description détaillée

9.17.2 Documentation des macros

9.17.2.1 #define `COLONNE_CLASSE` 4

9.17.2.2 #define `COLONNE_CLASSEMENT` 0

9.17.2.3 #define `COLONNE_NOM` 2

9.17.2.4 #define `COLONNE_NUMERODOSSARD` 1

9.17.2.5 #define `COLONNE_PRENOM` 3

9.17.2.6 #define `COLONNE_TEMPS` 5

9.17.2.7 #define `COURSE_DISTANCE` 3

9.17.2.8 #define `COURSE_HEUREDEPART` 4

9.17.2.9 #define `COURSE_ID_COURSE` 0

9.17.2.10 #define `COURSE_IDMANIFESTATION` 1

9.17.2.11 #define `COURSE_NOM` 2

9.17.2.12 `#define EMPLACEMENT_NOM 1`

9.17.2.13 `#define EMPLACEMENT_NOMCLASSE 3`

9.17.2.14 `#define EMPLACEMENT_NUMEROCLASSE 4`

9.17.2.15 `#define EMPLACEMENT_NUMERODOSSARD 0`

9.17.2.16 `#define EMPLACEMENT_PRENOM 2`

9.17.2.17 `#define EMPLACEMENT_TEMPS 5`

9.17.2.18 `#define MANIFESTATION_DATE 2`

9.17.2.19 `#define MANIFESTATION_ID 0`

Référencé par `IHMResultats : :recupererCourse()`.

9.17.2.20 `#define MANIFESTATION_NOM 1`

9.17.2.21 `#define TEMPO_STATUS 5000`

9.18 Référence du fichier main.cpp

Programme principal Chrono-cross-classement.

```
#include <QApplication> #include "chronocrossclassement.h"
```

Fonctions

– `int main (int argc, char *argv[])`

Programme principal : Crée et affiche la fenêtre principale de l'application.

9.18.1 Description détaillée

Crée et affiche la fenêtre principale de l'application

9.18.2 Documentation des fonctions

9.18.2.1 `main (int argc, char * argv[])`

Paramètres

<code>argc</code>	
<code>argv[]</code>	

Renvoie

int

```
{
    QApplication a(argc, argv);
    ChronoCrossClassement w;
    w.show();

    return a.exec();
}
```

9.19 Référence du fichier main.cpp

Programme principal.

```
#include <QApplication> #include <QLocale> #include <Q-
Translator> #include <QLibraryInfo> #include "ihmmanifestation.-
h"
```

Fonctions

– int `main` (int argc, char *argv[])

9.19.1 Description détaillée

Crée et affiche la fenêtre principale de l'application

9.19.2 Documentation des fonctions**9.19.2.1 int main (int argc, char * argv[])**

```
{
    QApplication a(argc, argv);
    QString locale = QLocale::system().name().section('_', 0, 0);
    QTranslator translator;
    translator.load(QString("qt_") + locale, QLibraryInfo::location(
        QLibraryInfo::TranslationsPath));
    a.installTranslator(&translator);

    IHMManifestation w;
    w.show();

    return a.exec();
}
```

9.20 Référence du fichier main.cpp

Programme principal.

```
#include <QtGui/QApplication> #include "ihmresultats.h"
```

Fonctions

– int `main` (int argc, char *argv[])

9.20.1 Description détaillée

Crée et affiche la fenêtre principale de l'application

9.20.2 Documentation des fonctions

9.20.2.1 int main (int argc, char * argv[])

```
{  
    QApplication a(argc, argv);  
    IHMResultats w;  
    w.show();  
  
    return a.exec();  
}
```

9.21 Référence du fichier README.dox

Index

- ~BaseDeDonnees
 - BaseDeDonnees, [12](#)
- ~ChronoCrossClassement
 - ChronoCrossClassement, [23](#)
- ~EditionCourse
 - EditionCourse, [37](#)
- ~EditionManifestation
 - EditionManifestation, [39](#)
- ~IHManifestation
 - IHManifestation, [46](#)
- ~IHMResultats
 - IHMResultats, [74](#)
- AK
 - chronocrossclassement.h, [84](#)
- BaseDeDonnees, [9](#)
 - ~BaseDeDonnees, [12](#)
 - BaseDeDonnees, [12](#)
 - baseDeDonnees, [19](#)
 - BaseDeDonnees, [12](#)
 - connecter, [12](#), [13](#)
 - db, [19](#)
 - detruireInstance, [13](#)
 - executer, [13](#), [14](#)
 - getInstance, [14](#)
 - getNomBaseDonnees, [14](#), [15](#)
 - mutex, [19](#)
 - nbAcces, [19](#)
 - recuperer, [15–19](#)
- CL
 - chronocrossclassement.h, [84](#)
- COLONNE_CLASSE
 - ihmanifestation.h, [87](#)
 - ihmresultats.h, [91](#)
- COLONNE_CLASSEMENT
 - ihmanifestation.h, [87](#)
 - ihmresultats.h, [91](#)
- COLONNE_NOM
 - ihmanifestation.h, [87](#)
 - ihmresultats.h, [91](#)
- COLONNE_PRENOM
 - ihmanifestation.h, [88](#)
 - ihmresultats.h, [91](#)
- COLONNE_TEMPS
 - ihmanifestation.h, [88](#)
 - ihmresultats.h, [91](#)
- COUREUR_ID
 - ihmanifestation.h, [88](#)
- COUREUR_IDCATEGORIE
 - ihmanifestation.h, [88](#)
- COUREUR_IDCLASSE
 - ihmanifestation.h, [88](#)
- COUREUR_NOM
 - ihmanifestation.h, [88](#)
- COUREUR_NOM_CLASSE
 - ihmanifestation.h, [88](#)
- COUREUR_NUMERO_INE
 - ihmanifestation.h, [88](#)
- COUREUR_PRENOM
 - ihmanifestation.h, [88](#)
- COUREUR_SEXE
 - ihmanifestation.h, [89](#)
- COURSE_DISTANCE
 - chronocrossclassement.h, [84](#)
 - ihmanifestation.h, [89](#)
 - ihmresultats.h, [91](#)
- COURSE_HEUREDEPART
 - chronocrossclassement.h, [84](#)
 - ihmanifestation.h, [89](#)
 - ihmresultats.h, [91](#)
- COURSE_ID
 - chronocrossclassement.h, [84](#)
- COURSE_ID_COURSE
 - ihmanifestation.h, [89](#)
 - ihmresultats.h, [91](#)
- COURSE_NOM
 - chronocrossclassement.h, [84](#)
 - ihmanifestation.h, [89](#)
 - ihmresultats.h, [91](#)
- Changelog.dox, [83](#)
- Chrono-Cross/basededonnees.h
 - DATABASENAME, [81](#)
 - HOSTNAME, [81](#)
 - PASSWORD, [81](#)
 - USERNAME, [81](#)
- Chrono-Cross/main.cpp
 - main, [92](#)
- ChronoCrossClassement, [19](#)
 - ~ChronoCrossClassement, [23](#)
 - ChronoCrossClassement, [22](#)
 - actualiserMessage, [23](#)
 - afficherTableTemps, [23](#)
 - arrivees, [35](#)

- bd, 35
- calculerChecksum, 24
- chargerInscritsCourse, 25
- chargerListeCourses, 25
- chargerListeManifestation, 26
- ChronoCrossClassement, 22
- courseDemarree, 26
- courseTerminee, 27
- courses, 35
- demarrerCourse, 27
- displayEvent, 35
- donnees, 36
- dossards, 36
- effacerTableTemps, 27
- enregistrerArriveeCoureur, 27
- enregistrerResultat, 28
- envoyer, 28
- estInscrit, 29
- finaliserDepartCourse, 29
- finaliserFinCourse, 29
- getIdInscrit, 30
- initialiserBD, 30
- initialiserPort, 30
- inscrits, 36
- manifestation, 36
- numeroCourse, 36
- numeroDossier, 31
- onReadyRead, 31
- port, 36
- quitter, 33
- selectionnerCourse, 33
- tempsArrivee, 34
- terminerCourse, 34
- topDepart, 34
- ui, 36
- verifierAcquittement, 35
- DATABASENAME
 - Chrono-Cross/basededonnees.h, 81
 - Gestion-Cross/basededonnees.h, 82
 - Resultats-Cross/basededonnees.h, 82
- DE
 - chronocrossclassement.h, 84
- DISPLAY_EVENT
 - chronocrossclassement.h, 84
- DS
 - chronocrossclassement.h, 84
- EMPLACEMENT_NOM
 - ihmmanifestation.h, 89
 - ihmresultats.h, 91
- EMPLACEMENT_PRENOM
 - ihmmanifestation.h, 89
 - ihmresultats.h, 92
- EMPLACEMENT_TEMPS
 - ihmmanifestation.h, 89
 - ihmresultats.h, 92
- EditionCourse, 36
 - ~EditionCourse, 37
 - EditionCourse, 37
 - chargerListeCourses, 37
 - creerCourse, 38
 - EditionCourse, 37
 - modifierCourse, 38
 - supprimerCourse, 38
- EditionManifestation, 38
 - ~EditionManifestation, 39
 - EditionManifestation, 39
 - chargerListeManifestations, 39
 - creerManifestation, 39
 - EditionManifestation, 39
 - modifierManifestation, 39
 - supprimerManifestation, 39
- Gestion-Cross/basededonnees.h
 - DATABASENAME, 82
 - HOSTNAME, 82
 - PASSWORD, 82
 - USERNAME, 82
- Gestion-Cross/main.cpp
 - main, 93
- HOSTNAME
 - Chrono-Cross/basededonnees.h, 81
 - Gestion-Cross/basededonnees.h, 82
 - Resultats-Cross/basededonnees.h, 82
- ID
 - chronocrossclassement.h, 84
- IHMManifestation, 40
 - ~IHMManifestation, 46
 - IHMManifestation, 44
 - activerImpression, 46
 - afficherConfigurationManifestation, 46
 - afficherInscriptionCoureurs, 46
 - afficherListeCoureurs, 46
 - afficherPublicationResultats, 47
 - afficherResultats, 47
 - ajouterResultat, 48
 - bd, 69
 - chargerListeCategories, 49
 - chargerListeClasses, 50

- chargerListeCoureurs, 50
- chargerListeCourses, 51
- chargerListeInscriptionCourses, 52
- chargerListeManifestations, 53
- choisirInscriptionCourse, 54
- configurerTableResultats, 54
- coureurs, 69
- creerCoureur, 54
- creerCourse, 55
- creerManifestation, 55
- demarrerInscriptionCoureur, 56
- dossards, 69
- editionDossard, 69
- effacerResultats, 56
- getIdCategorie, 56
- getIdClasse, 57
- getIdCoureur, 57
- getNumeroDossard, 57
- idCoureur, 69
- IHMManifestation, 44
- imprimerResultats, 58
- inscrireCoureur, 58
- listeCategories, 69
- listeClasses, 70
- listeCoureurs, 70
- listeCourses, 70
- listeCoursesResultats, 70
- listIdCoureurs, 70
- listeManifestations, 70
- listeManifestationsResultats, 70
- listeNomsCoureurs, 70
- listeNumerosDossardCoureurs, 70
- manifestation, 70
- modifierCoureur, 59
- modifierCourse, 60
- modifierManifestation, 60
- quitter, 61
- selectionnerCoureur, 61
- selectionnerCoureurListe, 62
- selectionnerCourse, 62
- selectionnerCourseResultats, 63
- selectionnerDossardListe, 64
- selectionnerListeCoureursCourse, 64
- selectionnerManifestation, 65
- selectionnerManifestationResultats, 66
- selectionnerOnglet, 66
- supprimerCoureur, 67
- supprimerCourse, 67
- supprimerManifestation, 68
- ui, 70
- verifierInformationsCoureur, 68
- verifierInformationsSuppression-Coureur, 69
- IHMResultats, 71
 - ~IHMResultats, 74
 - IHMResultats, 74
 - afficherCourse, 74
 - afficherManifestation, 75
 - afficherResultats, 75
 - ajouterResultat, 76
 - bd, 80
 - chargerListeCourses, 78
 - chargerListeManifestations, 78
 - configurerTableResultats, 78
 - course, 80
 - effacerResultats, 78
 - IHMResultats, 74
 - listeCourses, 80
 - listeCoursesResultats, 80
 - listeManifestationsResultats, 80
 - manifestation, 80
 - quitter, 79
 - rafraichir, 79
 - recupererCourse, 79
 - timerRafraichissement, 80
 - ui, 80
- MANIFESTATION_DATE
 - ihmmanifestation.h, 89
 - ihmresultats.h, 92
- MANIFESTATION_ID
 - ihmresultats.h, 92
- MANIFESTATION_NOM
 - ihmmanifestation.h, 90
 - ihmresultats.h, 92
- NOM_PORT
 - chronocrossclassement.h, 84
- ONGLET_INSCRIPTION
 - ihmmanifestation.h, 90
- ONGLET_RESULTATS
 - ihmmanifestation.h, 90
- OP
 - chronocrossclassement.h, 84
- PARAMETER
 - chronocrossclassement.h, 84
- PASSWORD
 - Chrono-Cross/basededonnees.h, 81
 - Gestion-Cross/basededonnees.h, 82

- Resultats-Cross/basededonnees.h,
82
- PERIODE
 - chronocrossclassement.h, 85
- README.dox, 94
- RR
 - chronocrossclassement.h, 85
- Resultats-Cross/basededonnees.h
 - DATABASENAME, 82
 - HOSTNAME, 82
 - PASSWORD, 82
 - USERNAME, 82
- Resultats-Cross/main.cpp
 - main, 94
- SN
 - chronocrossclassement.h, 85
- SYNCHRO
 - chronocrossclassement.h, 85
- SYSTEM_EVENT
 - chronocrossclassement.h, 85
- TEMPO_STATUS
 - ihmmanifestation.h, 90
 - ihmresultats.h, 92
- TIMEOUT
 - chronocrossclassement.h, 85
- USERNAME
 - Chrono-Cross/basededonnees.h, 81
 - Gestion-Cross/basededonnees.h, 82
 - Resultats-Cross/basededonnees.h,
82
- VE
 - chronocrossclassement.h, 85
- activerImpression
 - IHMManifestation, 46
- actualiserMessage
 - ChronoCrossClassement, 23
- afficherConfigurationManifestation
 - IHMManifestation, 46
- afficherCourse
 - IHMResultats, 74
- afficherInscriptionCoureurs
 - IHMManifestation, 46
- afficherListeCoureurs
 - IHMManifestation, 46
- afficherManifestation
 - IHMResultats, 75
- afficherPublicationResultats
 - IHMManifestation, 47
- afficherResultats
 - IHMManifestation, 47
 - IHMResultats, 75
- afficherTableTemps
 - ChronoCrossClassement, 23
- ajouterResultat
 - IHMManifestation, 48
 - IHMResultats, 76
- arrivees
 - ChronoCrossClassement, 35
- baseDeDonnees
 - BaseDeDonnees, 19
- basededonnees.cpp, 80, 81
- basededonnees.h, 81, 82
- bd
 - ChronoCrossClassement, 35
 - IHMManifestation, 69
 - IHMResultats, 80
- calculerChecksum
 - ChronoCrossClassement, 24
- chargerInscritsCourse
 - ChronoCrossClassement, 25
- chargerListeCategories
 - IHMManifestation, 49
- chargerListeClasses
 - IHMManifestation, 50
- chargerListeCoureurs
 - IHMManifestation, 50
- chargerListeCourses
 - ChronoCrossClassement, 25
 - EditionCourse, 37
 - IHMManifestation, 51
 - IHMResultats, 78
- chargerListeInscriptionCourses
 - IHMManifestation, 52
- chargerListeManifestation
 - ChronoCrossClassement, 26
- chargerListeManifestations
 - EditionManifestation, 39
 - IHMManifestation, 53
 - IHMResultats, 78
- choisirInscriptionCourse
 - IHMManifestation, 54
- chronocrossclassement.cpp, 83
- chronocrossclassement.h, 83
 - AK, 84
 - CL, 84
 - COURSE_DISTANCE, 84
 - COURSE_HEUREDEPART, 84

- COURSE_ID, [84](#)
- COURSE_NOM, [84](#)
- DE, [84](#)
- DISPLAY_EVENT, [84](#)
- DS, [84](#)
- ID, [84](#)
- NOM_PORT, [84](#)
- OP, [84](#)
- PARAMETER, [84](#)
- PERIODE, [85](#)
- RR, [85](#)
- SN, [85](#)
- SYNCHRO, [85](#)
- SYSTEM_EVENT, [85](#)
- TIMEOUT, [85](#)
- VE, [85](#)
- configurerTableResultats
 - IHMManifestation, [54](#)
 - IHMResultats, [78](#)
- connecter
 - BaseDeDonnees, [12](#), [13](#)
- coureurs
 - IHMManifestation, [69](#)
- course
 - IHMResultats, [80](#)
- courseDemarree
 - ChronoCrossClassement, [26](#)
- courseTerminee
 - ChronoCrossClassement, [27](#)
- courses
 - ChronoCrossClassement, [35](#)
- creerCoureur
 - IHMManifestation, [54](#)
- creerCourse
 - EditionCourse, [38](#)
 - IHMManifestation, [55](#)
- creerManifestation
 - EditionManifestation, [39](#)
 - IHMManifestation, [55](#)
- db
 - BaseDeDonnees, [19](#)
- demarrerCourse
 - ChronoCrossClassement, [27](#)
- demarrerInscriptionCoureur
 - IHMManifestation, [56](#)
- detruireInstance
 - BaseDeDonnees, [13](#)
- displayEvent
 - ChronoCrossClassement, [35](#)
- donnees
 - ChronoCrossClassement, [36](#)
- dossards
 - ChronoCrossClassement, [36](#)
 - IHMManifestation, [69](#)
- editionDossard
 - IHMManifestation, [69](#)
- editioncourse.cpp, [85](#)
- editioncourse.h, [85](#)
- editionmanifestation.cpp, [85](#)
- editionmanifestation.h, [86](#)
- effacerResultats
 - IHMManifestation, [56](#)
 - IHMResultats, [78](#)
- effacerTableTemps
 - ChronoCrossClassement, [27](#)
- enregistrerArriveeCoureur
 - ChronoCrossClassement, [27](#)
- enregistrerResultat
 - ChronoCrossClassement, [28](#)
- envoyer
 - ChronoCrossClassement, [28](#)
- estInscrit
 - ChronoCrossClassement, [29](#)
- executer
 - BaseDeDonnees, [13](#), [14](#)
- finaliserDepartCourse
 - ChronoCrossClassement, [29](#)
- finaliserFinCourse
 - ChronoCrossClassement, [29](#)
- getIdCategorie
 - IHMManifestation, [56](#)
- getIdClasse
 - IHMManifestation, [57](#)
- getIdCoureur
 - IHMManifestation, [57](#)
- getIdInscrit
 - ChronoCrossClassement, [30](#)
- getInstance
 - BaseDeDonnees, [14](#)
- getNomBaseDonnees
 - BaseDeDonnees, [14](#), [15](#)
- getNumeroDossard
 - IHMManifestation, [57](#)
- idCoureur
 - IHMManifestation, [69](#)

- ihmmanifestation.cpp, [86](#)
- ihmmanifestation.h, [86](#)
 - COLONNE_CLASSE, [87](#)
 - COLONNE_CLASSEMENT, [87](#)
 - COLONNE_NOM, [87](#)
 - COLONNE_PRENOM, [88](#)
 - COLONNE_TEMPS, [88](#)
 - COUREUR_ID, [88](#)
 - COUREUR_IDCATEGORIE, [88](#)
 - COUREUR_IDCLASSE, [88](#)
 - COUREUR_NOM, [88](#)
 - COUREUR_NOM_CLASSE, [88](#)
 - COUREUR_NUMERO_INE, [88](#)
 - COUREUR_PRENOM, [88](#)
 - COUREUR_SEXE, [89](#)
 - COURSE_DISTANCE, [89](#)
 - COURSE_HEUREDEPART, [89](#)
 - COURSE_ID_COURSE, [89](#)
 - COURSE_NOM, [89](#)
 - EMPLACEMENT_NOM, [89](#)
 - EMPLACEMENT_PRENOM, [89](#)
 - EMPLACEMENT_TEMPS, [89](#)
 - MANIFESTATION_DATE, [89](#)
 - MANIFESTATION_NOM, [90](#)
 - ONGLET_INSCRIPTION, [90](#)
 - ONGLET_RESULTATS, [90](#)
 - TEMPO_STATUS, [90](#)
- ihmresultats.cpp, [90](#)
- ihmresultats.h, [90](#)
 - COLONNE_CLASSE, [91](#)
 - COLONNE_CLASSEMENT, [91](#)
 - COLONNE_NOM, [91](#)
 - COLONNE_PRENOM, [91](#)
 - COLONNE_TEMPS, [91](#)
 - COURSE_DISTANCE, [91](#)
 - COURSE_HEUREDEPART, [91](#)
 - COURSE_ID_COURSE, [91](#)
 - COURSE_NOM, [91](#)
 - EMPLACEMENT_NOM, [91](#)
 - EMPLACEMENT_PRENOM, [92](#)
 - EMPLACEMENT_TEMPS, [92](#)
 - MANIFESTATION_DATE, [92](#)
 - MANIFESTATION_ID, [92](#)
 - MANIFESTATION_NOM, [92](#)
 - TEMPO_STATUS, [92](#)
- imprimerResultats
 - IHMManifestation, [58](#)
- initialiserBD
 - ChronoCrossClassement, [30](#)
- initialiserPort
 - ChronoCrossClassement, [30](#)
- inscrireCoureur
 - IHMManifestation, [58](#)
- inscrits
 - ChronoCrossClassement, [36](#)
- listeCategories
 - IHMManifestation, [69](#)
- listeClasses
 - IHMManifestation, [70](#)
- listeCoureurs
 - IHMManifestation, [70](#)
- listeCourses
 - IHMManifestation, [70](#)
 - IHMResultats, [80](#)
- listeCoursesResultats
 - IHMManifestation, [70](#)
 - IHMResultats, [80](#)
- listeldCoureurs
 - IHMManifestation, [70](#)
- listeManifestations
 - IHMManifestation, [70](#)
- listeManifestationsResultats
 - IHMManifestation, [70](#)
 - IHMResultats, [80](#)
- listeNomsCoureurs
 - IHMManifestation, [70](#)
- listeNumerosDossardCoureurs
 - IHMManifestation, [70](#)
- main
 - Chrono-Cross/main.cpp, [92](#)
 - Gestion-Cross/main.cpp, [93](#)
 - Resultats-Cross/main.cpp, [94](#)
- main.cpp, [92](#), [93](#)
- manifestation
 - ChronoCrossClassement, [36](#)
 - IHMManifestation, [70](#)
 - IHMResultats, [80](#)
- modifierCoureur
 - IHMManifestation, [59](#)
- modifierCourse
 - EditionCourse, [38](#)
 - IHMManifestation, [60](#)
- modifierManifestation
 - EditionManifestation, [39](#)
 - IHMManifestation, [60](#)
- mutex
 - BaseDeDonnees, [19](#)
- nbAcces

- BaseDeDonnees, [19](#)
- numeroCourse
 - ChronoCrossClassement, [36](#)
- numeroterDossard
 - ChronoCrossClassement, [31](#)
- onReadyRead
 - ChronoCrossClassement, [31](#)
- port
 - ChronoCrossClassement, [36](#)
- quitter
 - ChronoCrossClassement, [33](#)
 - IHMManifestation, [61](#)
 - IHMResultats, [79](#)
- rafraichir
 - IHMResultats, [79](#)
- recuperer
 - BaseDeDonnees, [15–19](#)
- recupererCourse
 - IHMResultats, [79](#)
- selectionnerCoureur
 - IHMManifestation, [61](#)
- selectionnerCoureurListe
 - IHMManifestation, [62](#)
- selectionnerCourse
 - ChronoCrossClassement, [33](#)
 - IHMManifestation, [62](#)
- selectionnerCourseResultats
 - IHMManifestation, [63](#)
- selectionnerDossardListe
 - IHMManifestation, [64](#)
- selectionnerListeCoureursCourse
 - IHMManifestation, [64](#)
- selectionnerManifestation
 - IHMManifestation, [65](#)
- selectionnerManifestationResultats
 - IHMManifestation, [66](#)
- selectionnerOnglet
 - IHMManifestation, [66](#)
- supprimerCoureur
 - IHMManifestation, [67](#)
- supprimerCourse
 - EditionCourse, [38](#)
 - IHMManifestation, [67](#)
- supprimerManifestation
 - EditionManifestation, [39](#)
 - IHMManifestation, [68](#)
- tempsArrivee
 - ChronoCrossClassement, [34](#)
- terminerCourse
 - ChronoCrossClassement, [34](#)
- timerRafraichissement
 - IHMResultats, [80](#)
- topDepart
 - ChronoCrossClassement, [34](#)
- ui
 - ChronoCrossClassement, [36](#)
 - IHMManifestation, [70](#)
 - IHMResultats, [80](#)
- verifierAcquittement
 - ChronoCrossClassement, [35](#)
- verifierInformationsCoureur
 - IHMManifestation, [68](#)
- verifierInformationsSuppressionCoureur
 - IHMManifestation, [69](#)