

Dossier projet Chrono-cross Revue finale

version 1.0







Sommaire

Sommaire	1
Introduction	2
Présentation générale	2
Présentation individuelle	2
Logiciels et matériels utilisés	2
Planification des tâches	3
Analyse du système	4
Diagramme de cas d'utilisation	4
Diagramme de déploiement	4
Diagramme de classes de l'application Gestion-Cross	6
Diagramme de classes de l'application Resultats-Cross	7
Base de données	8
Schéma relationnel	8
Exemples de requêtes SQL	9
Cas d'utilisation : configurer la manifestation	12
Scénarios de manifestation	12
Scénario de création de manifestation	12
Scénario de modification de manifestation	14
Scénario de suppression de manifestation	16
Scénarios de course	18
Scénario de création de course	18
Scénario de modification de course	21
Scénario de suppression de course	24
Cas d'utilisation : publier les résultats	27
Scénario de l'affichage des résultats	27
Cas d'utilisation : imprimer les résultats	31
Scénario d'impression	31
L'application "Resultats-Cross"	32
Cas d'utilisation : afficher les résultats	32
Prise en main de commandes utiles pour l'utilisation de la carte Raspberry Pi	33
Recette	34
Bilan :	34

Introduction

Dans le cadre de ma deuxième année de BTS SN IR j'ai travaillé sur le projet chrono-cross: chronométrage de course à pied. Il m'a été confié le module de gestion de manifestation que je vais présenter ci-après.

Présentation générale

Le projet Chrono-Cross consiste à mettre en place une automatisation presque complète d'une course à pied ou d'un cross. Il doit permettre la configuration d'une manifestation et des courses et d'autre part l'inscription des coureurs. Il doit aussi réaliser l'affichage des temps sur un panneau lumineux, lors du passage des coureurs devant la ligne d'arrivée. La détection est faite grâce à un capteur infrarouge. L'affichage des résultats (classement des coureurs) est accessible par l'organisateur et les coureurs, sur grand écran, avec la possibilité d'imprimer les résultats.

Présentation individuelle

Mon objectif est de faire une application qui permettra à l'organisateur du cross de :

- Créer, modifier ou supprimer une manifestation (nom et date de l'évènement);

Une fois la manifestation créée, il pourra :

- Créer, modifier ou supprimer des courses (nom, heure de départ, et la longueur de la course)

Durant la course :

- Afficher les résultats sur grand écran via une carte Raspberry Pi

Une fois la course terminée l'application permettra de :

- Afficher les résultats, et/ou
- Imprimer les résultats.

Logiciels et matériels utilisés

Le diagramme de Gantt a été réalisé avec Planner.

L'élaboration des diagrammes a été effectuée avec BOUML 7.4.

Les IHM ont été fait avec Qt Designer, et le codage avec Qt Creator en langage de programmation C++ avec l'API QT 4.8.

La rédaction du dossier : Google Doc.

Pour la réalisation du diaporama j'ai utilisé Google Slide.

Une carte Raspberry Pi pour l'affichage des résultats sur la télévision.









Planification des tâches

Les tâches à réaliser sont effectuées dans l'ordre chronologique suivant. Pour analyser le système, je commence par étudier le cahier des charges pour repérer les différentes tâches à réaliser. La conception débute par la réalisation des diagrammes : de cas d'utilisation, de séquence, de classe et de déploiement. La réalisation du code source permet de configurer des manifestations et l'affichage des résultats. Un exécutable est implémenté sur le PC course de l'organisateur. Une carte Raspberry Pi est reliée à la télévision par un câble HDMI pour afficher les résultats. Les tests unitaires permettent de vérifier la fonctionnalité des applications.





Analyse du système

Diagramme de cas d'utilisation



Le diagramme de cas d'utilisation permet de pointer les fonctionnalités majeures du projet qui sont :

- La configuration d'une manifestation,
- La possibilité de publier les résultats ainsi que l'option pour l'organisateur de les imprimer,
- La possibilité de visualiser les résultats sur grand écran par les coureurs.

Diagramme de déploiement





Les deux applications "Gestion-Cross" et "Resultats-Cross" ont été développées en C++ avec l'API QT 4.8. La base de données MySQL "Chrono-Cross" est installée sur le PC Course et accessible aussi à partir de la Raspberry Pi.

Le système de chronométrage est relié via un adaptateur USB/RS232.



Diagramme de classes de l'application Gestion-Cross

Ce diagramme de classes est une retranscription des attributs et méthodes des classes utilisés dans le projet Gestion-Cross.

IHMManifestation		
 listeManifestations : QStringList 		
- listeCourses : QStringList		BaseDeDonnees
 listeCoureurs : QStringList 		- nbAcces : int
- listeClasses : QStringList	\rightarrow	- db : QSqlDatabase
- listeCategories : OStringList	-baseDeDonnees	- mutex : QMutex
 listeNomsCoureurs : QStringList 		+ getInstance() : BaseDeDonnees
- listeNumerosDossardCoursurs : OStringlist		+ detruireInstance() : void
 listel/Courseurs : OString list 		+ getNomBaseDonnees() : OString
enursurs : OStringListMadel	ы	+ connecter/serveur : OString, nomBase : OString) : hool
 oureurs : OstringListModel 	-00-	+ avanter(requete : OString) : bool
dossards : ComingListModel Extended to Chine List	/	+ executer(requete : OString, doppoor : OString) : bool
InstemantestationsResultats : UStringList InstemantestationsResultats : UStringList		+ recuperentrequete : OString, donnees : OString) : bool
- listeCoursesResultats : OStringList		+ recuperen(requete : Ostring, donnees : OstringList) : bool
- mannestation : Q5tringList		+ recuperentrequete : Ostring, donnees : Ovector <ostrings) :="" bool<="" th=""></ostrings)>
- editionDossard : bool		+ recuperen(requete : Ustring, donnees : Qvector <ustringlist>) : bool</ustringlist>
- idCoureur : QString		- BaseDeDonnees()
+ IHMManifestation(parent : QWidget)		 ~BaseDeDonnees()
+ ~IHMManifestation()		
 chargerListeManifestations() : void 		
 chargerListeCourses(idManifestation : QString) : void 		
 chargerListeClasses() : void 		
- get/dClasse() : QString		
 chargerListeCategories() : void 		
 getIdCategorie() : QString 		
 verifierInformationsCoureur() : bool 		
- getNumeroDossard(idCoureur : QString, idCourse : QString) : QString	a	
- getIdCoureur(nom : QString) : QString		
- verifierInformationsSuppressionCoureur() : bool		
- configurerTableBesultats() : void		
 effacerBesultats() : void 		
- afficherConfigurationManifestation() : void		
 afficherInscriptionCourseurs() : void 		
 afficharPublicationBesultats() : void 		
- guitter() : wid		
 selectionnerOnglet/index : int) : void 		
 crearManifectation() : void 		
- modifierManifestation() : void		
 mountermanifestation(): void 		
 selectionnerManifestation() : wid 		
 selectionnermannestation() : voio serectionnermannestation() : voio 		
modifierCourse() : void		
 modifierGourse() : void supprimerCourse() : void 		
- selectionnerCourse() : wid		
 descriptionCourse() - void 		
eroseCeurourit unid		
mediliarCourse() : usid		
 modifierCoureur(): void 		
 supprimercoureur() - void 		
 chargerListeCoureurs(): void 		
 selectionnerListeCoureursCourse() : void 		
- afficherListeGoureurs() : void		
 selectionnerCoureurListe(index : QModelIndex) : void 		
 selectionnerDossardListe(index : QModelIndex) : void 		
 demarrerInscriptionCoureur() : void 		
 inscrireCoureur() : void 		
 selectionnerCoureur() : void 		
 chargerListeInscriptionCourses() : void 		
 afficherResultats(idCourse : QString) : void 		
 activerImpression() : void 		
 ajouterResultat(ligne : int, unResultat : QStringList) : void 		
 selectionnerManifestationResultats() : void 		
 selectionnerCourseResultats() : void 		
 imprimerResultats() : void 		



Diagramme de classes de l'application Resultats-Cross

Ce diagramme de classe est une retranscription des attributs et méthodes des classes utilisés dans le projet Resultats-Cross.







Base de données

La base de données Chrono-cross contient les informations relatives aux :

- manifestations,
- courses,
- inscriptions à la course,
- coureurs,
- temps d'arrivée des coureurs,
- catégories des coureurs,
- classe du coureur.

Schéma relationnel

Le schéma ci-dessous précise les liaisons entre les tables et les multiplicités.



Pour gérer la base de données Chrono-Cross les requêtes SQL utilisées sont :

- **INSERT** est une commande SQL qui ajoute un ou plusieurs tuple(s) dans la table d'une base de données relationnelles.
- **UPDATE** est une commande SQL qui modifie un ou plusieurs tuple(s) dans la table d'une base de données relationnelles.
- **DELETE** est une commande SQL qui supprime un ou plusieurs tuple(s) dans la table d'une base de données relationnelles.



- **SELECT** est une commande SQL qui permet d'extraire des données des tables d'une base de données relationnelles.

Exemples de requêtes SQL

- Requête INSERT :

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

Cette requête insère dans la table "Manifestation" dans le champ "Nom" et "Date" les valeurs contenues dans les variables "lineEditNomManifestation" et "dateEditEditionManifestation".

- Requête UPDATE :

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

Cette requête modifie dans la table "Manifestation" les données du champ "Nom" par les données contenues dans la variable "lineEditNomManifestation", dans le champ "Date" les données contenue dans la variable "dateEditEditionManifestation".

- Requête **DELETE** :

DELETE FROM Manifestation WHERE idManifestation = "" + idManifestation + "

Cette requête supprime dans la table "Manifestation" le "Nom" et la "Date" de la manifestation identifiée sous le numéro "idManifestation".

- Requête SELECT :

SELECT * FROM Manifestation ORDER BY Date ASC;

Cette requête récupère les données de tous les champs de la table "Manifestation" classées par date.



L'application d'édition de manifestation

🛞 🖨 🗊 Gestion de Chross			
Manifestation Inscription Result	ats		
15 16			
10			
			1
Manifestation :			: 4
Date de la manifestati	on:	19/04/2018	2
Nom de la manifestati	on :	Nom de la manifestation	
Crée	m Mo	difier	C 17
4		5 6	✓ 5
			7
		A second	
Nom de la course :		Nom de la course	8
Heure de début de la d	course :	08:00:00	9
Longueur de la course		En mètre	⊐10</th
Créer	Modifier	Supprimer	
11	12	13 🔓 14	
a.			

Partie manifestation :

- 1 : Liste déroulante des manifestations 2 : Ligne de saisie de la date
- 3 : Ligne de saisie du nom
- 5 : Bouton de modification

Partie course :

- 7 : Liste déroulante des courses
- 9 : Ligne de saisie de l'heure de départ 10 : Ligne de saisie de la distance
- 11 : Bouton de création
- 13 : Bouton de suppression
- 15 : Onglet "Manifestation"

- 4 : Bouton de création
- 6 : Bouton de suppression
- 8 : Ligne de saisie du nom
- 12 : Bouton de modification
- 14 : Bouton vers l'onglet "Inscription"
- 16 : Onglet "Resultats"



L'affichage des résultats :

Manifestation Inscription Resultats	
÷	<u> </u>
Course : 4 Longueur de la	ourse : 5
Classement Numéro de dossard Nom Prénom Classe	Temps
6	
0	
Imprimer 7	

- 1 : Liste déroulante des manifestations
- 3 : Afficheur du nom de la course

5 : Afficheur de la longueur de la course course

7 : Bouton d'impression

- 2 : Liste déroulante des courses
- 4 : Afficheur de l'heure de départ
- 6 : Tableau des résultats de la



Cas d'utilisation : configurer la manifestation

Scénarios de manifestation

Scénario de création de manifestation



Pour la création d'une manifestation la méthode appelée est :



La déclaration d'une variable "retour" de type booléen initialisée en false.

La condition "if" pour vérifier la présence d'une saisie dans la ligne d'édition.

La création de la variable requete qui contient le code SQL pour une insertion.

L'appel de la méthode executer() pour réaliser l'exécution de la requête. Elle retourne true si la requête a été exécutée avec succès sinon false.

Si la requête s'est bien exécutée alors on recharge la liste des manifestations pour mettre à jour la liste déroulante.



Pour créer une manifestation il suffit de modifier la date en saisissant la date souhaitée au clavier ou en utilisant le bouton de scroll. La date affichée par défaut est celle du jour. Puis il faut entrer le nom de la manifestation, et appuyer sur le bouton "Créer". Une fois que les manipulations sont effectuées, les paramètres de la manifestation sont enregistrés dans la base de données.

😣 🗐 🗐 Gestia	on de Chross	
Manifestation	Inscription Resultats	
1	Manifestation :	\$
1	Date de la manifestation :	19/04/2018
i	Nom de la manifestation :	Nom de la manifestation 2 Modifier Supprimer
I: Ligne de	saisie de la date	2 : Ligne de saisie du nom

- 1 : Ligne de saisie de la date
- 3 : Bouton de création

Si lors de la création de la manifestation l'organisateur oublie de saisir le nom de la manifestation, un message d'alerte prévient l'organisateur :





Scénario de modification de manifestation



Pour la modification d'une manifestation la méthode appelée est :

void IHMManifestation::modifierManifestation()
{
 bool retour = false;
 // Vérifications
 if(ui->lineEditNomManifestation->text().isEmpty())
 {
 return;
 }
 // Récupére 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 + "";
 retour = bd->executer(requete);
 if(retour)
 {
 chargerListeManifestations();
 }
}

La déclaration d'une variable "retour" de type booléen initialisée en false.

La condition "if" pour vérifier la présence d'une saisie dans la ligne d'édition.

La création d'une variable idManifestation qui contient la valeur de l'identifiant SQL.

La création de la variable requete qui contient le code SQL pour une modificaion.

L'appel de la méthode executer() pour réaliser l'exécution de la requête. Elle retourne true si la requête a été exécutée avec succès sinon false.

Si la requête s'est bien exécutée alors on recharge la liste des manifestations pour mettre à jour la liste déroulante.



Pour modifier une manifestation, l'organisateur sélectionne, dans la liste déroulante des manifestations, celle qu'il souhaite modifier :

Manifestation :		
Date de la manifestation :	Cross Campu Cross Colleg	us 2018
Nom de la manifestation :	Nom de la m	anifestation
Créer	Modifier	Supprimer

Après modification de la manifestation, l'organisateur peut modifier la date si nécessaire, en respectant la même procédure que celle utilisée lors de la création d'une manifestation (cf. page 12). Lorsque la date est actualisée, il suffit d'entrer le nouveau nom choisi pour la manifestation puis appuyer sur le bouton "Modifier". Une fois que les manipulations sont effectuées les modifications apportées aux paramètres de la manifestation sont enregistrées dans la base de données.

lanifestation	Inscription Resultats		
	Manifestation -	Cross Campus 2018	. 1
	Jace de la manuestación :	05/04/2018	- //
1			

- 1 : Liste déroulante des manifestations
- 3 : Ligne de saisie du nom

- 2 : Ligne de saisie de la date
- 4 : Bouton de modification



Scénario de suppression de manifestation

犬 :Organisateur	:IHMManifestation	:BaseDeDonnees
supprimerManifestation() e chargerListeMani	DELETE FROM Manifestation WHERE idManifestation = "" +
Pour la suppression d'une mani	ifestation la méthod	e appelée est :
void IHMManifestation::supprimerManifes { bool retour = false;	station()	
// Récupére l'identifiant de la m QString idManifestation = listel QString requete = "DELETE EL	anifestation Manifestations.at(ui->liste0	CreationManifestations->currentIndex()-1).at(0);

retour = bd->executer(requete);

}

f(retour)	
ť í	chargerListeManifestations():
	onalgerzietennameetatiene(),

La déclaration d'une variable "retour" de type booléen initialisée en false.

La création d'une variable idManifestation qui contient la valeur de l'identifiant SQL.

La création de la variable requete qui contient le code SQL pour une suppression.

L'appel de la méthode executer() pour réaliser l'exécution de la requête. Elle retourne true si la requête a été exécutée avec succès sinon false.

Si la requête s'est bien exécutée alors on recharge la liste des manifestations pour mettre à jour la liste déroulante.

Pour supprimer une manifestation de la base de données, il est nécessaire que l'organisateur sélectionne dans la liste déroulante des manifestations, celle qu'il souhaite supprimer.

	Cross Camp	us 2018 🔓
Date de la manifestation :	Cross Colleg	je 2018
Nom de la manifestation :	Nom de la m	anifestation
Créer	Modifier	Supprimer



Lorsque la manifestation est identifiée, il suffit d'appuyer sur le bouton "Supprimer".

anifestation	Inscription Resultats	
	Manifestation :	Cross Campus 2018
1	Manifestation : Date de la manifestation :	Cross Campus 2018 2018

- 1 : Liste déroulante des manifestations
- 2 : Bouton de suppression



Scénarios de course

Scénario de création de course



retour = b	d->executer(requete);
f(retour) { }	chargerListeCourses(idManifestation);

}

La déclaration d'une variable "retour" de type booléen initialisée en false.

La condition "if" pour vérifier la présence d'une saisie dans la ligne d'édition.

La création d'une variable idManifestation qui contient la valeur de l'identifiant SQL.

La création de la variable requete qui contient le code SQL pour une insertion.

L'appel de la méthode executer() pour réaliser l'exécution de la requête. Elle retourne true si la requête a été exécutée avec succès sinon false.

Si la requête s'est bien exécutée alors on recharge la liste des manifestations pour mettre à jour la liste déroulante.



Pour créer une course, il est nécessaire que l'organisateur sélectionne dans la liste déroulante des manifestations, la manifestation dans laquelle il veut créer la course.

Manifestation :		
Date de la manifestation :	Cross Campus 2018	
Nom de la manifestation :	Nom de la m	anifestation
Créer	Modifier	Supprimer

Une fois sélectionnée il suffit de saisir le nom de la course, modifier l'horaire en saisissant l'heure souhaitée au clavier ou en utilisant le bouton de scroll. L'heure affichée par défaut est 8 heures du matin . Lorsque l'heure est paramétrée, il suffit d'entrer la longueur de la course et d'appuyer sur le bouton "Créer". Les paramètres de la course sont alors enregistrés dans la base de données.



😣 🖨 🗊 🛛 Gestio	on de Chross
Manifestation	Inscription Resultats
N E N	Manifestation : Cross Campus 2018 2018 2018 2018 2018 2018 2018 2018
N F L	Nom de la course : Heure de début de la course : Longueur de la course : En mètre Supprimer Inscrire

- 1 : Liste déroulante des manifestations
- 3 : Ligne de saisie de l'heure de départ
- 5 : Bouton de création

- 2 : Ligne de saisie du nom
- 4 : Ligne de saisie de la distance

Si lors de la création de la course l'organisateur oublie de saisir le nom de la manifestation ou la longueur de la course, il y a un message d'alerte qui le prévient :





Scénario de modification de course



Pour la modification d'une course la méthode appelée est :

bool retour = false; // Vérifications
// Vérifications
ff(ui->lineEditNomCourse->text().isEmpty()) { return;
// Récupére 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 + "";
retour = bd->executer(requete);
f(retour) { chargerListeCourses(listeManifestations.at(ui->listeCreationManifestations->currentIndex()-1).at(0)); }

La déclaration d'une variable "retour" de type booléen initialisée en false.

La condition "if" pour vérifier la présence d'une saisie dans la ligne d'édition.

La création d'une variable idCourse qui contient la valeur de l'identifiant SQL.

La création de la variable requete qui contient le code SQL pour une modification.

L'appel de la méthode executer() pour réaliser l'exécution de la requête. Elle retourne true si la requête a été exécutée avec succès sinon false.

Si la requête s'est bien exécutée alors on recharge la liste des manifestations pour mettre à jour la liste déroulante.



Pour modifier une course, il est nécessaire que l'organisateur sélectionne dans la liste déroulante des manifestations, celle qu'il veut modifier.

Manifestation :		N
Date de la manifestation :	Cross Campus 2018 Cross College 2018 Nom de la manifestation	
Nom de la manifestation :		
Créer	Modifier	Supprimer

Une fois la manifestation sélectionnée il peut choisir la course à modifier.

Cross M15 E - 3500m - 13b00	
Cross M15 M - 5000m - 14h00	
Heure de début de la course :	08:00:00
Longueur de la course :	En mètre
Créer Modifier	Supprimer Inscrire



L'organisateur peut dès lors modifier le nom, l'heure et la longueur de la course si nécessaire, puis appuyer sur le bouton "Modifier". Une fois que les manipulations sont effectuées, les modifications apportées aux paramètres de la course sont enregistrées dans la base de données.

😣 🗐 🗊 Gestion de Chross		
Manifestation Inscription Resultats		
Manifestation : Date de la manifestation Nom de la manifestation Créer	Cross Campus 2018 Cross Campus 2018 Cross Campus 2018 Modifier Supprimer	
Cross M15 F - 3500m - 13	shoo ==)2	
Nom de la course :	Cross M15 F 3	
Heure de début de la cou	rse : 13:00:00 ‡ 4	
Longueur de la course :	3500 5	
Créer	6 Mod Supprimer Inscrire	

- 1 : Liste déroulante des manifestations
- 3 : Ligne de saisie du nom
- 5 : Ligne de saisie de la distance
- 2 : Liste déroulante des courses
- 4 : Ligne de saisie de l'heure
- 6 : Bouton de modification



Scénario de suppression de course



Pour la suppression d'une course la méthode appelée est :

);

La déclaration d'une variable "retour" de type booléen initialisée en false.

La création d'une variable idCourse qui contient la valeur de l'identifiant SQL.

La création de la variable requete qui contient le code SQL pour une suppression.

L'appel de la méthode executer() pour réaliser l'exécution de la requête. Elle retourne true si la requête a été exécutée avec succès sinon false.

Si la requête s'est bien exécutée alors on recharge la liste des manifestations pour mettre à jour la liste déroulante.



Pour supprimer une course, il est nécessaire que l'organisateur sélectionne dans la liste déroulante des manifestations, celle dans laquelle il veut supprimer la course.

Manifestation :		N
Date de la manifestation :	Cross Campus 2018 Cross College 2018	
Nom de la manifestation :	Nom de la manifestation	
Créer	Modifier	Supprimer

Une fois que la manifestation est sélectionnée, il peut choisir la course à supprimer.

Cross M15 F - 3500m - 13h00	
Cross M15 M - 5000m - 14h00	
Teure de début de la course :	08:00:00
ongueur de la course :	En mètre



Puis il appuie sur le bouton "Supprimer". Lorsque les manipulations sont effectuées, la manifestation est supprimée de la base de données.

😣 🖨 🗊 Gest	ion de Chross	
Manifestation	Inscription Resultats	
	Manifestation :	Cross Campus 2018 2
	Date de la manifestation :	05/04/2018
	Nom de la manifestation :	Cross Campus 2018
	Créer	difier Supprimer
	Cross M15 F - 3500m - 13h00	\$)2
	Nom de la course :	Cross M15 F
	Heure de début de la course :	13:00:00
	Longueur de la course :	3500
	Créer Modifier	Suppringer 3 Inscrire

- 1 : Liste déroulante des manifestations
- 2 : Liste déroulante des courses

3 : Bouton de suppression

Grâce à ces fonctionnalitées il est aisément possible pour l'organisateur de gérer les manifestations et les courses. Lorsque les manifestations et courses sont terminées, il a la possibilité de rejoindre l'onglet "Inscription" à l'aide du bouton Inscrire.





Cas d'utilisation : publier les résultats

Scénario de l'affichage des résultats



Pour la publication des résultats sur l'application Gestion-Cross les méthodes appelées sont .

- La méthode effacerResultats() est appelé afin de vider le tableau des résultats potentiellement déjà présent.



La déclaration d'une variable "nb" de type int.

La boucle "for" pour ajouter le nombre de lignes nécessaire dans le tableau. Rend indisponible le bouton "Imprimer".



- La méthode afficherResultats() est appelé afin d'insérer les résultats dans le tableau.

void IHMManifestation::afficherResultats(QString idCourse)
// <u>On commence par</u> effacer le contenu précédent effacerResultats();
// Récupérer les résultats d'une course (Requête SQL) QVector <qstringlist> resultats; Débies envire :: SEE EDIT facet Numero Dessert, Courses Nem, Courses Nem AS Nem Classo Reserve</qstringlist>
Usting requete = SELEC 1 Inscnt.Numerobossaro, Coureur.Nom, Coureur.Prenom, Classe.Nom AS NomClasse, Classe.Numero, Arrives Temos FROM Course. Inscrit.Coureur. Classe.Arrivee WHERE Course. IdCourse = Inscrit.idCourse.ND Inscrit.idCourse.idCourse.idCourse.
AND Courseur idClasse = Classe idClasse AND Inscrit.idInscrit = Arrivee.idInscrit AND Course.idCourse = " + idCourse + " ORDER BY Arrivee.Temps
ASC;*;
bool retour = bd->recuperer(requete, resultats);
(in(retour 1= taise))
1 DStringList unResultat
// Ajouter chaque élément dans le TableWidget
for(int ligne = 0; ligne < resultats.size(); ligne++)
usBecultet = secultets at(lisea);
univesukar – resukarstat(ingrie),
ajouterResultat(ligne, unResultat);
activerimpression();
, (
-

L'appel de la méthode "effacerResultats" avant d'insérer les résultats dans le tableau. Déclare une liste de type *vector* appelé "resultats".

La création de la variable requete qui contient le code SQL pour une sélection.

L'appel de la méthode recuperer() pour récupérer des information de l'exécution de la requête et les informations des résultats. Elle retourne true si la requête a été exécutée avec succès sinon false.

La condition "if" pour vérifier la présence de résultats pour les insérer dans le tableau. Déclare la liste "unResultat".

Boucle "for" qui ajoute les résultats dans le tableau.

Exécute la méthode "activerImpression()".

- La méthode activerImpression() est appelé afin de rendre le bouton "Imprimer" accessible.

void IHMN	Manifestation::activerImpression()
`	ui->boutonImprimerResultats->setEnabled(true);

}

28/34



Pour l'affichage des résultats l'organisateur doit sélectionner l'onglet "Resultats". Cet onglet permet à l'organisateur de visionner les résultats d'une course qui s'est terminée, et de pouvoir, s'il le souhaite, les imprimer sous format papier ou format PDF. Pour y parvenir il doit sélectionner, dans la liste déroulante des manifestations, la manifestation, puis dans la liste déroulante des courses, la course qu'il souhaite visionner.

(🕽 🗐 🗊 Gestion de	Chross				
	Manifestation Ins	cription Resultats				
1	Cross Campus 201 Cross College 201	8		Cross M15 F - 3500n Cross M15 M - 5000	7 n - 13h00 m - 14h00	
	Classement	Numéro de dossard	Nom	Prénom	Classe	Temps

- 1 : Liste déroulante des manifestations
- 2 : Liste déroulante des courses



Une fois la course sélectionnée, le nom de la course saisi, l'heure de départ et la longueur de la course s'affichent au dessus du tableau de résultats. Le tableau classe les coureurs par ordre d'arrivée. Les 3 premières personnes arrivées sont affichées en couleur (Or, Argent, Bronze), les suivantes dans un affichage classique. Le tableau des résultats indique : le classement, le numéro de dossard, le nom, le prénom, la classe et le temps des élèves.

Cross Campus 20	18		Cross M15 F - 3	500m - 13h00	
ourse :	Cross M15 F	Heure de départ :	13h00	Longueur de la course : 3500	
Classement	Numéro de dossard	Nom	Prénom	Classe Te	mps
1	101	PERRICHON	Julia	4E 1 00:	12:56
2	102	MOUTARD	Camille	4E 1 00:	13:44
3	103	MOLIST	Lucille	4E 1 00:	14:11
4	104	RIES	Clementine	4E 1 00:	14:52
5	105	LAMOUREUX	Felicia	4E 1 00:	15:17
5	106	STEY	Pauline	4E 1 00:	16:30
7	102	MOUTARD	Camille	4E 1 00:	17:50
в	108	BODIN	Alexia	4E 1 00:	18:54
9	109	TERREC	Maeile	4E 1 00:	21:06
10	110	FORNES	Marie	4E 1 00:	21:46
11	111	WINTREBERT	Pauline	4E 1 00:	22:08
12	112	GOURLET	Romane	4E 1 00:	22:25
13	113	VINCENT	Ines	4E 1 00:	23:01
14	114	DUTOT	Camille	4E 1 00:	25:06
			i i i i i i i i i i i i i i i i i i i	Service Control	

- 1 : Liste déroulante des manifestations
- 3 : Affichage des paramètres de la course
- 5 : Bouton d'impression

- 2 : Liste déroulante des courses
- 4 : Tableau de résultats



Cas d'utilisation : imprimer les résultats

Scénario d'impression



La méthode imprimerResultats() est appelé afin d'imprimer le tableau de résultats.

Quand les résultats de la course sont affichés il est possible de les imprimer sous différents formats (papier, PDF) grâce à une fenêtre de dialogue.

nprimante	
Nom :	Hewlett-Packard-HP-LaserJet-M1536dnf-MFP 🛟 Propriétés
Emplacement :	
Emplacement : Type :	HP - HP LaserJet M1536dnf MFP

Exemple d'impression en format PDF :

Manifestation : Cross Campus 2018 - Date : 24-05-2018

Course : Cross M15 F - Heure de départ : 11h00 - Longueur de la course : 3500 m

Classement	Numéro de dossard	Nom	Prénom	Classe	Temps
1	101	PERRICHON	Julia	4E 1	00:12:56
2	102	MOUTARD	Camille	4E 1	00:13:44
3	103	MOLIST	Lucille	4E 1	00:14:11
4	104	RIES	Clementine	4E 1	00:14:52
5	105	LAMOUREUX	Felicia	4E 1	00:15:17
6	106	STEY	Pauline	4E 1	00:16:30
7	107	BIRE-HESLOUIS	Maele	4E 1	00:17:50
8	108	BODIN	Alexia	4E 1	00:18:54
9	109	TERREC	Maelle	4E 1	00:21:06
10	110	FORNES	Marie	4E 1	00:21:46
11	111	WINTREBERT	Pauline	4E 1	00:22:08
12	112	GOURLET	Romane	4E 1	00:22:25
13	113	VINCENT	Ines	4E 1	00:23:01
14	114	DUTOT	Camille	4E 1	00:25:06
15	115	PREVOST	Emmie	4E 1	00:29:12



L'application "Resultats-Cross"

Cas d'utilisation : afficher les résultats



Le code qui affiche les résultats sur cet application est similaire au code page 27 sans les parties relatives à l'impression (effacerResultats(), ajouterResultats()).

La méthode supplémentaire est la méthode rafraichir()



Appel de la méthode afficherManifestation().

Condition "if" pour vérifier la présence d'une course, dans ce cas elle affiche les résultats.

Cette IHM est en plein écran. Elle affiche les paramètres de la manifestation et la course en cours sur un grand écran TV.

Le tableau comporte les mêmes colonnes que l'onglet "Resultats" de l'IHM de Gestion-Manifestation (Classement, Numéro de dossard, Nom, Prénom, Classe, Temps).

		Cross Fin /	Annee BTS 2018-05-25		
se : Course BTS SN		Heure de départ : 16h0	00	Longueur : 3800 m	
Classement	Numéro de doss	ard Nom	Prénom	Classe	Temps
	112	GOURLET	Romane	4E 1	00:00:07
	111	WINTREBERT	Pauline	4E 1	00:00:12
	113	VINCENT	Ines	4E 1	00:00:19
			2		
			14		

L'application s'actualise toute les secondes et affiche les informations de la manifestation et la course la plus récente sur l'écran télé. Une fois la course terminé les classements s'affichent. En voici un exemple ci-dessus.

Prise en main de commandes utiles pour l'utilisation de la carte Raspberry Pi

Mon professeur a mis à ma disposition une carte Raspberry Pi, un écran d'ordinateur et un clavier me permettant de commencer à utiliser la carte. Nous avons eu besoin d'installer Raspbian qui est un OS de type Debian pour la RaspberryPi. Puis nous avons paramétré la carte, téléchargé Qt et la bibliothèque MySQL nécessaire au programme et le client subversion pour accéder au dépôt du projet.

```
$ sudo raspi-config
```

```
$ apt-get qt4-dev-tools
$ apt-get qt-default
$ apt-get libqt4-sql-mysql
```

```
$ apt-get subversion
```

La Raspberry Pi doit être configurée en mode Kiosk.



<u>Recette</u>

Tests	Oui	Non
La création d'une manifestation est possible	X	
La création des courses pour une manifestation est possible	X	
L'affichage des informations pendant une course est fonctionnel	X	
L'affichage du classement d'une course est fonctionnel	X	
L'impression des résultats est possible	x	

Bilan :

Ce projet m'a permis d'utiliser l'environnement de développement Qt, dans le langage de programmation C++. Cela m'a permis aussi d'approcher des attentes d'un milieu professionnel par les limites de temps, le travail en équipe et les contraintes attendu par le client.

Pellizzoni Corentin

Projet Chrono-cross



PROJET CHRONO-CROSS

REVUE FINALE

version 1.0

Étudiant n°3 : Pellizzoni Corentin (IR)



1



0		•	
	mm	aır	ρ
$\mathbf{D}\mathbf{U}$		an	L

Présentation générale	3
Présentation individuelle	3
Planification	4
Ressources logicielles	5
Travail personnel	6
Application Gestion-Cross	6
Diagramme de déploiement	6
Diagramme de cas d'utilisation Gestion-Cross	6
Diagramme de classe	7
IHM	8
Base de données	9
Cas d'utilisation : créer un coureur	10
Scénario Création d'un coureur	11
Scénario Modification d'un coureur	13
Scénario Supprimer un coureur	14
Cas d'utilisation : inscrire un coureur à une course	15
Application ChronoCrossClassement	16
Diagramme de déploiement	16
Diagramme de cas d'utilisation ChronoCrossClassement	16
Diagramme de classe	17
IHM	18
Base de données	19
Module Chronomètre	20
Format	20
Acquittement	21
Messages	21
Liste des messages	23
Échanges	23
Démarrage d'une course	23
Temps à l'arrivée	24
Terminer une course	24
Cas d'utilisation : Démarrer une course	25
Scénario Démarrer une course manuellement	25
Scénario Démarrer une course logiciellement	26
Scénario Terminer une course	27
Cas d'utilisation : Chronométrer et classer les arrivées	28
Scénario Arrivée des coureurs	29
Recette	31
Bilan	31
Annexes	32
Ressources	32
Mise en oeuvre du chronomètre TAG heuer hl975	32

2



Présentation générale

Le projet Chrono-Cross consiste à mettre en place une automatisation presque complète d'une course à pied ou d'un cross. Avec une interface graphique il permet de configurer et d'afficher les résultats de la course en détectant les arrivées par une cellule infrarouge.

Projet Chrono-cross



module de chronométrage (EC)

Présentation individuelle

Ma partie consiste dans un premier temps à réaliser une application permettant au client de pouvoir gérer une manifestation sportive dans notre cas le projet chrono-cross.

Cette application demande à l'organisateur d'inscrire des coureurs à une course. Pour cela il faut entrer le numéro de dossard de l'élève, le nom de l'élève, son prénom, sa classe puis la catégorie dans laquelle l'élève a choisi de participer.

Pour ma part, je dois assurer la saisie des données entrées sur l'application, l'enregistrement dans une base de données et l'affichage dans l'IHM. Pour la base de données, je dois écrire toute les requêtes SQL. Je dois aussi récupérer les temps des coureurs à l'aide du module de chronométrage fourni par l'étudiant EC et les enregistrer.



Planification

La Salle

Le diagramme de Gantt permet de visualiser et d'établir chronologiquement les tâches effectuées lors de la réalisation du projet.





La Salle

Ressources logicielles

Système d'exploitation du PC « Course »	GNU Linux
Environnement de développement (IR)	Qt Creator et Qt Designer
API GUI PC « Course »	Qt 4.8
Compilateurs	GNU g++ for Linux
Système de gestion de bases de données relationnelles	MySQL
Gestion et administration de bases de données	phpMyAdmin
Atelier de génie logiciel (IR)	bouml 7.5
Logiciel de gestion de versions (IR)	Client subversion
Générateurs de documentation (IR)	Doxygen version 1.8.11
Gestionnaire de projet (IR)	Planner



Travail personnel

Application Gestion-Cross

Diagramme de déploiement



Le PC « Course » héberge le serveur de base de données (MySQL)

Diagramme de cas d'utilisation Gestion-Cross



L'acteur humain de ce système est : "Organisateur". Il prépare les coureurs et inscrit les coureurs.



Diagramme de classe

Ce diagramme de classe retranscrit tous les attributs et toutes les méthodes de l'application Gestion-Cross.

Interventions : OStingList Interventions Interventions Interventions Interventions Intervention In
 iste Course: OShingList iste Course: OShingList<
<pre>isteCourser: CShingList isteCategorie: CShingList isteCategorie: CShingList isteMameCourser: CShingList isteMameCourser: CShingList isteMameCourser: CShingList isteMameCourser: CShingList isteMameCourser: CShingList courser: CShingListModel dosard: CShingList dot corresponder(CShing): Dool editorDosard: bool dot dosard: CShingList dot corresponder(CShing): Dool editorDosard: CShingList dot corresponder(CShing): Dool configure(Categorie): void getMamifestation): void dot chargeristECCategorie: (void getCategorie): Void dot afticherArbicitation: CShing): CShing dot afticherArbicitation: CShing): Void afticherArbicitation: Void dot afticherArbicitation: Void afticherArbicitation: Void dot afticherArbicitation: Void asperimeCourse): vo</pre>
 IstelCases : OShingList IstelCases : Void IstelCases : Void IstelCases : OShingList IstelCases : Void IstelCases : Void IstelCases : OShingList IstelCases : Void IstelCases : Void IstelCases : OShingList IstelCases : Void IstelCase : Intrevid IstelCase : Introvend IstelCase
<pre>isiteCatagorie: OShingList isiteNonCourse:: OShingLis</pre>
 istelworsCoursers : OStringList istelworsCoursers : OStringList istelworsCoursCourser : OStringList istelworsCoursCoursCoursCoursCoursCoursCoursCo
 istel:Aumero2DosandCourser: 0StringList istel:Aumero2DosandCourser: 0StringListModel dosand: 0StringListModel istel:Aumero2DosandCourser: 0String. istel:Aumero2D
<pre>istetGoureur: CStingListModel dosards : GStringListModel istetGoursesEvaluts : GStringList istetCoursesEvaluts : GStringList endicolOsard: bool - idCourseur : OString isterCourseSevaluts : GStringList endicolOsard: bool - idCourseur : OString - id</pre>
 courser: CStringLisModel dossant: OStringLisModel isteManifestationsResultats : OStringList isteManifestationsResultats : OStringList isteManifestation: OStringList isteManifestation: OStringList editionDossand: bool editionDossand: bool courser: (requete : OString, domees : OStringList: editionDossand: bool courser: (requete : OString, domees : OVector-OStringList: editionDossand: bool courser: (requete : OString, domees : OVector-OStringList: editionDossand: bool courser: (requete : OString, domees : OVector-OStringList:): i: editionDossand: (Source: : OString): void charger: (requete : OString, domees : OVector-OStringList:): i: editionEconsent): void editionEconsent(Sourcer: : OString): osid editionEconsent(Sourcer: : OString): void estectionnerManifestation): void est
dosands : QStringListModel isteManifestationsResultats : QStringList isteManifestationsResultats : QStringList manifestation : QStringList editionDosard : bool editionDosard : bool idCourser : QString : bool ecuperer(requete : QString, donnees : QVector-QString) : bool chargerListCateCourse(IdManifestation : OString) : void chargerListCateGories() : void getQCategories() : void getQCategories() : void getQCategories() : void effaceFracultats() : void effaceFracultats() : void effaceFracultats() : void effaceFracultats() : void effaceFracultats() : void effaceFracultats() : void estectionnerCourse() : void selectionnerCourse() : void estectionnerCourse() : vo
 isteManifestationsResultat: OStingList isteCourseResultat: OStingList isteCourseResultat: OStingList editonDossard: bool editonDossard: bool idCourse: OStingList iHMManifestation: OStingList editonDossard: bool editonDossard: CourseRes: OWingle: bool ecuperer(requée: OStingdonnees: OWector-OSting_List): i enterpresented enterpresented
 istaCoursesResultat: OStringList manifestation: OStringList editionDosard: bool editonDosard: bool editonDosard: Course: OWdogat) HHMManifestation() chargerListeCourses(Manifestations): void chargerListeCourses(Manifestations) cold editonDosard(Course): void getfaClases(): void editonPossand(course): void estectionRestation): void selectionRestation): void selectionRestation): void estectionRestation): void estectionRestation: void estectionRestation: void estectionRestation: void estectionRestation: void estectionRestation: void estectionRestation: void
 marifestation: QStringList editionDosard : bood idCoureu: : QString iHMMarifestation[arent: QWdget] HHMMarifestation[: void chargerListeRaseB: : void getIdClasseD: : OString editoration: : QString : context getIdClasseD: : OString getIdClasseD: : OString getIdClasseD: : OString getIdClasseD: : OString getIdClasseD: : void afficherConfigurationManifestation() : void afficherConfigurationManifestation() : void afficherConfigurationManifestation() : void afficherConfigurationManifestation() : void getIdClasseD: : void getI
 editionDossard : bool idCoureur : QString HHMAnifestation[parent : QWdget] HHMAnifestation[parent : QWdget] etargerListeManifestation[) chargerListeCarses[(MAnifestation : QString) : void chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation : QString] : void getIdCategorie] : QString chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation]) chargerListeCarses[(Manifestation]) cold getIdCategorie] : QString getIdCourse() : void getIdCourse() : void getIdCourse() : void effacerResultate[) : void effacerResultate[) : void afficherInscriptionCourse() : void afficherInscriptionCourse() : void selectionerConffuration() : void selectionerConffuration() : void selectionerCourse[) : void<
 idCourser: CString IHMAnifestation(parent: CWidget) IHMAnifestation() chargerListeCourses(idManifestation: CString): void chargerListeClasses(): void getIdClasse(): CString chargerListeClasses(): void getIdClasse(): CString verifierInformationsCourseur(): bool getIdClasse(): CString verifierInformationsCourseur(): bool getIdClasse(): void effloerResultats(): void effloerResultats(): void afficherVoifgurationNanifestation(): void afficherVoifgurationNanifestation(): void afficherVoifgutificationResultats(): void coreefAnifestation(): void selectonnerManifestation(): void selectonnerManifestation(): void selectonnerManifestation(): void selectonnerCourse(): void selectonnerCourse(): void adprimerCourse(): void adprimerCourse(): void adprimerCourse(): void adprimerCourse(): void adprimerCourse(): void adprimerCourse(): void creerCourse(): void creerCourse(): void adprimerCourse(): void adprimerCourse(): void creerCourse(): void creerCour
 HHMAmifestation() - HHMAmifestation() - dhargerListeCourse()(Manifestation: CSting): void - dhargerListeCourse()(Manifestation: CSting): void - dhargerListeCalsase(): void - gettQLase(): Soling - dhargerListeCalsase(): void - gettQLase(): Soling - verifierInformationsCoureur(): bool - gettQLase(): OSting, - verifierInformationsSuppressionCoureur(): bool - officerResultat(): void - afficherNeur(nom: OSting): OSting - verifierInformationsSuppressionCoureur(): bool - onfigurerTableResultat(): void - afficherNeur(nom: OSting): Void - afficherNeuropiet(Index: int): void - afficherNeuropiet(Index: int): void - afficherNeuropiet(Index: int): void - afficherNeuropiet(Index: int): void - selectonnerManifestation(): void - selectonnerManifestation(): void - selectonnerCourse(): void - selectonnerCourse(): void - selectonnerCourse(): void - afficherNeuropiet(Index: int): void - selectonnerCourse(): void - apprimerCourse(): void - ap
 HHMManifestation() chargerListeManifestations(): void chargerListeOcurses()(Manifestation : QString) : void chargerListeOcurses(): void getIdClasse(): QString verifierInformationsCoursur(): bool getIdCourceur(nom : QString) : QString verifierInformationsCoursur(): bool configurationAlanifestation(): void afficherInscriptionCourceur() : void afficherInscriptionCourceur() : void afficherInscriptionCourceur(): void afficherInscriptionCourceur(): void selectionnerOnglet(Index : Int): void creenAanifestation(): void selectionnerOnglet(Index : Int): void selectionnerOnglet(Index : Int): void afficherInscriptionCourceur(): void selectionnerOnglet(Index : Int): void selectionnerOnglet(In
 chargerListeQuartiestations(): void chargerListeQuartiestation: QString): void gettQClasse(): QString chargerListeQategories(): void gettQClasse(): QString verifierInformationsCourcer(): bool gettQCourser(nom: QString): QString): QString): QString gettQCourser(nom: QString): QString): QString wertifierInformationsCourcer(): bool configurerTableResultats(): void effacerResultats(): void afficherCourse(): void afficherCourse(): void afficherCourse(): void afficherCourse(): void selectionnerOnglet(index: int): void selectionnerManifestation(): void supprimerManifestation(): void selectionnerManifestation(): void se
 chargerListeCourse(iManifestation : QString) : void chargerListeCatases() : void getIdClass() : QString chargerListeCategories() : void getIdCategorie() : QString verifierInformationsCourse() : bool getIdCourse() : void getIdCourse() : void efficierInseriptionCourse() : void afficherPublicationResultats() : void accereCourse() : void supprimerCourse() : void supprimerCourse() : void afficherInscriptionCourse() : void supprimerCourse() : void supprimerCourse() : void supprimerCourse() : void afficherInscriptionCourse() : void supprimerCourse() : void supprimerCourse() : void afficherInscriptionCourse() : void apprimerCourse() : void
 chargerListeClasses(): void getldClasse(): GString chargerListeCategories(): void getldCategorie(): GString verliferInformationsCoureur(): bool getldCategorie(): GString: getldCategorie(): GString: GString: getldCategorie(): GString: GString: GString:
gettdClasse(): QString ohargerListeCategories(): Svoid gettdCategories(): CString verifierInformationsCourseur(): bool gettNumernDossard(idCourseur: QString): QString gettldCourseur(nom: QString): QString verifierInformationsCourseur(): bool configurerTableResultats(): void efficientPosciptionCourseur(): void afficherConfigurationManitestation(): void afficherProciptionCourseurs(): void afficherProciptionCourseurs(): void afficherProciptionCourseurs(): void afficherProciptionCourseurs(): void selectionnerConglet(index: int): void reree/Manifestation(): void selectionnerCourse(): void supprimerManifestation(): void supprimerCourse(): void
 chargerListeCategories(): void gettdCategories(): String verifierInformationsCoureur(): bool gettdCoureur(nom: QString): QString gettdCoureur(nom: QString): QString verifierInformationsSuppressionCoureur(): bool configuerTableResultats(): void effacerResultats(): void afficherConfigurationManifestation(): void afficherPublicationResultats(): void afficherPublicationResultats(): void afficherInscriptionCoureur(): void afficherInscriptionCoureur(): void selectionnerOnglet(index: inf): void selectionnerCoure(): void supprimerCoure(): void supprimerCoure(): void supprimerCoure(): void supprimerCoure(): void
 getldCategorie(): OString verifierInformationsCoureur(): bool getNumeroDossard(idCoureur: QString): QString): QString getldCoureur(nom: QString): QString verifierInformationsSuppressionCoureur(): bool configurerTableResultats(): void afficherConfigurationManifestation(): void afficherConfigurationManifestation(): void afficherPublicationResultats(): void getldCoureur(): void getldCategorie(): void coreerManifestation(): void supprimerManifestation(): void supprimerManifestation(): void supprimerManifestation(): void supprimerCourse(): void supprimerCourse(): void supprimerCourse(): void supprimerCourse(): void creerCourse(): void supprimerCourse(): void supprimerCourse(): void creerCourse(): void creerCourse(): void supprimerCourse(): void supprimerCourse(): void chargerListeCourseurs(): void modifierCourse(): void modifierCourse(): void modifierCourse(): void supprimerCourse(): void modifierCourse(): void modifierCourse(): void modifierCourse(): void modifierCourse(): void modifierCourse(): void
 verifierInformationsCoureur() : bool getINtmeroDossard(idCoureur : QString) : QString) : QString getIdCoureur(nom : QString) : QString verifierInformationsSuppressionCoureur() : bool configurerTableResultats() : void effacerResultats() : void afficherConfigurationManifestation() : void afficherInscriptionCoureurs() : void afficherInscriptionCoureurs() : void afficherInscriptionCoureurs() : void selectionnerOnglet(index : int) : void supprimerManifestation() : void supprimerManifestation() : void selectionnerCourse() : void supprimerCourse() : void supprimerCourse() : void selectionnerCourse() : void supprimerCourse() : void
 getNumeroDossard(idCoursur : QString) : QString) : QString getIdCoursur(nom : QString) : OString configureTableResultats() : void effacerResultats() : void afficherInscriptionCoursur() : void afficherInscriptionCoursur() : void afficherInscriptionCoursur() : void selectionnerConglet(index : int) : void selectionnerConglet(index : int) : void selectionnerManifestation() : void selectionnerManifestation() : void selectionnerConglet(index : int) : void selectionnerConglet(index : int) : void selectionnerConglet(index : int) : void selectionnerManifestation() : void supprimerManifestation() : void selectionnerCourse() : void selectionnerCourse() : void selectionnerCourse() : void selectionnerCourse() : void supprimerCourse() : void
 getldCoureur(nom : QString) : QString verifierInformationsSuppressionCoureur() : bool configurerTableResultats() : void effacerResultats() : void afficherConfigurationManifestation() : void afficherInscriptionCoureurs() : void afficherPublicationResultats() : void afficherRonfigurationManifestation() : void selectionnerOnglet(index : int) : void creerManifestation() : void supprimerManifestation() : void supprimerManifestation() : void supprimerManifestation() : void supprimerManifestation() : void selectionnerManifestation() : void supprimerManifestation() : void supprimerCourse() : void selectionnerCourse() : void creerCourse() : void selectionnerCourse() : void supprimerCourse() : void supprimerCourse() : void creerCourse() : void creerCourse() : void supprimerCourse() : void supprimerCourse() : void creerCourse() : void creerCourse() : void
 verifierInformationsSuppressionCourseur(): bool configurerTableResultats(): void afficherResultats(): void afficherConfigurationManifestation(): void afficherPublicationResultats(): void quitter(): void quitter(): void selectionnerOnglet(index: int): void creerManifestation(): void supprimerManifestation(): void selectionnerManifestation(): void selectionnerCourse(): void selectionnerCourse(): void creerCourse(): void
 configurerTableResultats() : void effacerResultats() : void afficherConfigurationManifestation() : void afficherInscriptionCoureurs() : void afficherPublicationResultats() : void quitter() : void selectionnerOnglet(index : int) : void creerManifestation() : void supprimerManifestation() : void selectionnerManifestation() : void selectionnerCourse() : void supprimerCourse() : void creerCourse() : void creerCourse() : void selectionnerCourse() : void creerCourse() : void supprimerCourse() : void supprimerCourse() : void
 effacerResultats() : void afficherConfigurationManifestation() : void afficherInscriptionCoureurs() : void afficherPublicationResultats() : void quitter() : void selectionnerOnglet(index : int) : void creerManifestation() : void modifierManifestation() : void selectionnerManifestation() : void selectionnerCourse() : void supprimerCourse() : void creerCourse() : void creerCourse() : void creerCourse() : void supprimerCourse() : void creerCourse() : void
 afficherConfigurationManifestation(): void afficherPublicationResultats(): void quitar(): void selectionnerOnglet(index: int): void creerManifestation(): void modifierManifestation(): void supprimerManifestation(): void selectionnerManifestation(): void creerCourse(): void selectionnerCourse(): void selectionnerCourse(): void creerCourse(): void supprimerCourse(): void supprimerCourse(): void
 afficherInscriptionCoureurs() : void afficherPublicationResultats() : void quitter() : void selectionnerOnglet(index : int) : void creerManifestation() : void supprimerManifestation() : void selectionnerManifestation() : void creerCourse() : void creerCourse() : void supprimerCourse() : void selectionnerCourse() : void selectionnerCourse() : void creerCourse() : void selectionnerCourse() : void supprimerCourse() : void creerCourse() : void choisirInscriptionCourse() : void creerCoureur() : void creerCoureur() : void creerCoureur() : void supprimerCoureur() : void supprimerCoureur() : void supprimerCoureur() : void
 afficherPublicationResultats(): void quitter(): void selectionnerOnglet(index : int): void creerManifestation(): void supprimerManifestation(): void selectionnerManifestation(): void selectionnerManifestation(): void creerCourse(): void modifierCourse(): void supprimerCourse(): void selectionnerCourse(): void selectionnerCourse(): void choisirInscriptionCourse(): void creerCoureur(): void creerCoureur(): void choisirInscriptionCourse(): void supprimerCourse(): void creerCoureur(): void creerCoureur(): void creerCoureur(): void supprimerCourse(): void supprimerCourse(): void
 quiter(): void selectionnerOnglet(index : int) : void creerManifestation() : void supprimerManifestation() : void selectionnerManifestation() : void creerCourse() : void modifierCourse() : void supprimerCourse() : void selectionnerCourse() : void selectionnerCourse() : void selectionnerCourse() : void selectionnerCourse() : void supprimerCourse() : void selectionnerCourse() : void supprimerCourse() : void choisirInscriptionCourse() : void modifierCourse() : void creerCourse() : void creerCourse() : void choisirInscriptionCourse() : void supprimerCourse() : void modifierCourse() : void supprimerCourse() : void modifierCourse() : void supprimerCourse() : void
 selectionnerOnglet(index : int) : void creerManifestation() : void supprimerManifestation() : void selectionnerManifestation() : void creerCourse() : void creerCourse() : void supprimerCourse() : void selectionnerCourse() : void choisirInscriptionCourse() : void creerCourse() : void creerCourse() : void selectionnerCourse() : void choisirInscriptionCourse() : void creerCourse() : void supprimerCourse() : void supprimerCourse() : void
 creerManifestation(): void supprimerManifestation(): void supprimerManifestation(): void selectionnerManifestation(): void creerCourse(): void modifierCourse(): void supprimerCourse(): void selectionnerCourse(): void creerCourse(): void creerCourse(): void creerCourse(): void supprimerCourse(): void creerCourse(): void
 modifierManifestation(): void supprimerManifestation(): void selectionnerManifestation(): void creerCourse(): void supprimerCourse(): void selectionnerCourse(): void choisirInscriptionCourse(): void creerCoureur(): void modifierCoureur(): void supprimerCoureur(): void supprimerCoureur(): void supprimerCoureur(): void
 supprimer/Manifestation(): void selectionner/Manifestation(): void creer/Course(): void modifier/Course(): void supprimer/Course(): void choisin/nscription/Course(): void creer/Course(): void creer/Course(): void supprimer/Course(): void choisin/nscription/Course(): void supprimer/Course(): void supprimer/Course(): void choisin/nscription/Course(): void
 selectionnerManifestation(): void creerCourse(): void supprimerCourse(): void selectionnerCourse(): void choisirInscriptionCourse(): void creerCoureur(): void modifierCoureur(): void supprimerCoureur(): void supprimerCoureur(): void chargerListeCoureur(): void
 creerCourse() : void modifierCourse() : void supprimerCourse() : void selectionnerCourse() : void choisirInscriptionCourse() : void creerCoureur() : void modifierCoureur() : void supprimerCoureur() : void chargerListeCoureur() : void
 modifierCourse() : void supprimerCourse() : void selectionnerCourse() : void choisirInscriptionCourse() : void creerCoureur() : void modifierCoureur() : void supprimerCoureur() : void chargerListeCoureur() : void
 supprimerCourse() : void selectionnerCourse() : void choisirInscriptionCourse() : void creerCoureur() : void modifierCoureur() : void supprimerCoureur() : void chargerListeCoureur() : void
 selectionnerCourse() : void choisirInscriptionCourse() : void creerCoureur() : void modifierCoureur() : void supprimerCoureur() : void chargerListeCoureurs() : void
 choisinfnscriptionCourse(): void creerCoureur(): void modifierCoureur(): void supprimerCoureur(): void chargerListeCoureurs(): void
- creerCoureur() : void - modifierCoureur() : void - supprimerCoureur() : void - chargerListeCoureurs() : void
- modifierCoureur() : void - supprimerCoureur() : void - chargerListeCoureurs() : void
- supprimerCoureur() : void - chargerListeCoureurs() : void
- chargerListeCoureurs() : void
- selectionnerListeGoureursCourse() : void
- afficherListeCoureurs() : void
 selectionnerCoureurListe(index : QModelIndex) : void
 selectionnerDossardListe(index : QModelIndex) : void
- demarrerInscriptionCoureur() : void
- inscrireCoureur() : void
- selectionnerCoureur() : void
- chargerListeInscriptionCourses() : void
- afficherResultats(idCourse : QString) : void
- activerImpression() : void
 ajouterResultat(ligne : int, unResultat : QStringList) : void
- selectionnerManifestationResultats() : void
- selectionnerCourseResultats() : void
- imprimerResultats() : void



IHM

Ici l'application GestionCross avec l'onglet "Inscription" permettant la création et l'inscription d'un coureur à une course :

m du coureur nom du coureur /01/2000				
nom du coureur /01/2000				
/01/2000	2			
*				
mère INE du couraur				
meno mue os solaresti				
1	:			
3M :				
:				
2				
eur	Dossard	: Numéro de dossard		
	13 M : odifier Supprimer supprimer :	13 M : bdifier Supprimer eur Dossard	13 M : bdifier Supprimer eur Dossard : Numéro de dossard	13 M : odifier Supprimer eur Dossard : Numéro de dossard

L'application Gestion-Cross a deux utilités la première partie sert à la création d'un coureur ainsi qu'à sa modification ou sa suppression. La deuxième partie sert à inscrire un coureur à une course en lui associant un numéro de dossard.



La Salle



Cette base de données est composée de 6 tables, Manifestation, Course, Inscrit, Coureur, Catégorie et Classe pour l'application Gestion-Cross, la première partie de l'application va utiliser les tables Coureur, Catégorie et Classe et la deuxième partie va utiliser Course, Inscrit et Coureur.



a Salle

Projet Chrono-cross

Cas d'utilisation : créer un coureur

Manifestation	iption Resultats
	= 1
Nom :	Nom du coureur 2
Prénom :	Prénom du coureur 3
Date de naissance :	01/01/2000
Sexe :	M 2 5
Numéro INE :	Numéro INE du coureur 6
Classe :	3E 1 : 7
Catégorie :	M13 M 🗧 🔞
Créer 9	Modifier Supprimer

1 - Il s'agit d'une liste déroulante avec tous les coureurs préalablement enregistrés pour pouvoir poursuivre sur la procédure de modification ou de suppression .

2- Nom du coureur.

- 3- Prénom du coureur.
- 4- Date de naissance du coureur

5- Sexe du coureur (option préalablement configurer pour deux choix unique Mâle ou Femelle)

6- Numéro INE (Identifiant National Etudiant) composé de 9 chiffres et de deux lettres en majuscules.

7-Classe du coureur.

8-Catégorie du coureur.

9-Bouton de création du coureur.

10-Bouton de modification d'un coureur préalablement sélectionné dans la liste (1).

11-Bouton de suppression d'un coureur préalablement sélectionné dans la liste (1).

La première partie va permettre de configurer toutes les données pour l'inscription d'un coureur qui seront ensuite envoyés dans une base de données dans laquelle sera stockées toutes les données .Ces dernières sont stockées grâce à une requête SQL tant pour la création, modification ou suppression.



Scénario Création d'un coureur

Pour la création d'un coureur, il faut remplir tous les champs nécessaires par la suite appuyer sur le bouton Créer. Une requête SQL va enregistrer toutes les infos rentrées dans une base de données Chrono-Cross.

Nom :	Pellizzoni
Prénom :	Corentin
Date <mark>de naissance</mark> :	20/03/1997
Sexe :	M ‡
Numéro INE :	235164987PC
Classe :	3E 3 🗧
Catégorie :	M13 M 🛟



Vue de la base de données : nous voyons bien que le coureur Pellizzoni n'existe pas dans la base de données.

2	4	7	123456789BB	MOUTARD	Camille	2004-01-08	F
16	3	7	234567891AA	PELIOT	Julien	2004-02-16	м
1	4	7	123456789AA	PERRICHON	Julia	2004-04-16	F
29	3	7	234567891NN	PIGNON	Jean	2004-10-20	м
15	4	7	12345678900	PREVOST	Emmie	2004-01-02	F
4	4	7	123456789DD	RIES	Clementine	2004-06-16	F
19	3	7	234567891DD	RIOUX	Clement	2004-07-07	м

Ici nous voyons que le coureur Pellizzoni existe dans la base de données.

chargerListeCoureurs()

2	4	7	123456789BB	MOUTARD	Camille	2004-01-08	F	
16	3	7	234567891AA	PELIOT	Julien	2004-02-16	м	
96	1	12	235164987PC	PELLIZZONI	Corentin	1997-03-20	м	-
1	4	7	123456789AA	PERRICHON	Julia	2004-04-16	F	
29	3	7	234567891NN	PIGNON	Jean	2004-10-20	м	



Scénario Modification d'un coureur

Pour modifier un coureur, il faut sélectionner le coureur dans la liste déroulante toute les informations vont s'afficher et il faudra modifier la donnée voulant l'être et cliquer sur le bouton modifier, une requête SQL est envoyé à la base de données Chrono-Cross.



La modification sera porté sur la catégorie du coureur .

96	5	12	235164987PC	PELLIZZONI	Corentin	1997-03-20	М
	1						

Nous pouvons constater que dans la base la catégorie a était modifié.

96	1	12	235164987PC	PELLIZZONI	Corentin	1997-03-20	М
	1						



Scénario Supprimer un coureur

Pour supprimer un coureur, il faut sélectionner le coureur dans la liste déroulante toute les informations vont s'afficher et il faudra cliquer sur le bouton supprimer, une requête SQL est envoyé à la base de données Chrono-Cross.



Nous voyons bien que le coureur PELLIZZONI a été supprimé de la base de données.

17	3	7	234567891BB	MOULARD	Charles	2004-03-08	м
2	4	7	123456789BB	MOUTARD	Camille	2004-01-08	F
16	3	7	234567891AA	PELIOT	Julien	2004-02-16	м
1	4	7	123456789AA	PERRICHON	Julia	2004-04-16	F
29	3	7	234567891NN	PIGNON	Jean	2004-10-20	м
15	4	7	12345678900	PREVOST	Emmie	2004-01-02	F
4	4	7	123456789DD	RIES	Clementine	2004-06-16	F



La Salle Avignon rères des Ecoles Chrétisenes

Cas d'utilisation : inscrire un coureur à une course

La deuxième partie va permettre d'inscrire un coureur à une course et de pouvoir lui associer un numéro de dossard pour la course sélectionnée.

Course :	: 1	
Coureur : Nom du coureur	Dossard : Numéro de dossard	3
	4 5	

- 1- Liste déroulante avec toutes les courses
- 2- Nom du coureur
- 3- Numéro de dossard du coureur
- 4- Vue de tous les coureurs
- 5- Vue des numéros de dossard

Prenons comme exemple la course moins de 15 ans féminines.

Coureur: Nom du coureur	Dossard : Nur	néro de dossard	
BIRE-HESLOUIS Maele	A	107	6
BODIN Alexia		108	
BOTIN Alexis	E		
CLEVOST Emile			
DURAND Lucien			
DUTOT Camille		114	
FORNES Marie		110	
FURLES Mario			
GOURLET Romane		112	
GOUSSET Romain			
HERROUIS Maurice	-		

Nous pouvons voir que les participants à cette course sont bien toutes féminines, Nous constatons que chaque coureur a un numéro de dossard attribué pour cette course.



La Salle Avignon rrires des Ecoles Chrétiennes

Application ChronoCrossClassement

Diagramme de déploiement



Le PC « Course » héberge le serveur de base de données (MySQL), le système embarqué Chronomètre est relié au PC-Course avec une liaison USB/RS232, le Chronomètre est géré dans la partie EC.

Diagramme de cas d'utilisation ChronoCrossClassement



L'acteur humain de ce système est : "Organisateur". Il démarre la course et classe les arrivées. L'acteur matériel « Chronomètre » interagit avec le système par l'échange de trames qui permettent d'obtenir les temps et l'ordre des arrivées.



Diagramme de classe

Ce diagramme de classe retranscrit tous les attributs et toutes les méthodes de l'application Gestion-Cross



- selectionnerCourse() : void



La Salle

Ici l'application ChronoCrossClassement permettant de démarrer une course et de classer les arrivées des coureurs :

Démarrer Top Départ Terminer Arrivées Nb Arrivées : Nb Classés : classement Dossard Temps	ourse :			
Arrivées: Nb Classés : Classement Dossard Temps	Démarrer	Тор	Départ	Terminer
coureurs: Nb Arrivées: Nb Classés:		Arr	rivées	
Classement Dossard Temps	coureurs :	Nb Arrivées :	Nb Classés :	
	Classement	Dossard	Tr	emps

La première partie sert à sélectionner puis de démarrer une course et la deuxième partie sert à afficher le nombre de coureurs inscrit à la course le nombre d'arrivées et le nombre de coureurs classés, afficher les temps reçu sur un tableau et ainsi pouvoir associer le numéro de dossard avec le temps correspondant.



La Salle



La même base de données va être utilisé mais en ajoutant la table **Arrivee**, ce qui va permettre d'enregistrer les temps d'arrivés des coureurs pour ensuite les attribués au numéro de dossard associés au Coureur.



Module Chronomètre

Pour la partie détection de temps j'utilise un chronomètre TAG heuer hl975. Ce chronomètre utilise le protocole THCOM08, celui qui sera implanté dans le module de l'étudiant EC.



Avec l'aide des documents fournis et d'un programme de test j'ai pu comprendre comment fonctionne le chronomètre tout d'abord j'ai paramétré la liaison entre le chronomètre et le PC avec une liaison USB/RS232. J'ai paramétré par la suite une connexion au port automatique. Une fois tout cela effectué, j'ai testé avec l'outil cutecom.

On utilise l'outil cutecom sur le fichier de périphérique /dev/hl975 et la configuration suivante : * 9600 bits/s

- * 8 bits de données
- * 1 bit de stop
- * pas de parité

Format

Le format d'une trame est le suivant : [Data] + TAB + CS16 + CR + LF

- Data : voir les messages décrits ci-dessous
- TAB : délimiteur du Checksum (code ASCII 0x09)
- CS16 : Checksum sur 16 bits (4 chiffres codés en hexadécimal)
- CR + LF : délimiteur de fin de trame (codes ASCII 0x0D + 0x0A)

Le format d'une trame sans Checksum sera le suivant : [Data] + TAB + CR + LF



Acquittement

Pour chaque trame envoyée vers l'appareil (*Device*), celui-ci répond à la machine émettrice (*Host*) par une trame AK d'acquittement qui a le format suivant : AK X

X = 'C' accepted, 'F' rejected, 'R' not supported

Messages

Serial number (ID) : ID NNNNN

N = Serial number (0 - 65535)

Exemple : ID 01145

Serial number request (#ID) : #ID

Serial number + Device type + soft version (SN) : SN_NNNNN_TTTTT_VVVV

N = Serial number (0 - 65535) T = Device type (CP540, HL440, HL940) V = Software version (example: VA05)

Exemple : SN 01145 HL975 VA10

Serial number and device type request (#SN) : #SN

R = Run number (1 - 99)
T = 'T' if the added Run is itself the addition of two other Runs '_' otherwise.
A = Added Run (1 - 99)
X = Name of the Timing Mode (Max 19 chars)

Closing of a Run (CL) : CL RR

R = Run number (1 - 99)

Time (TN): TN SSSS CC HH:MM:SS.FFFFF DDDDD

S = Sequential number (0 - 9999) C = Channel number (1 - 99) in case of manual entry (M1 - M4) 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



Exemples :

TN	1	2	3.71300 365
TN	2	2	8.00500 365
TN	3	2	1:01.48200 365

Read a parameters (#RP) : #RP III

I = parameter ID (000 - 999)

trame = "#RP 003"; // &P 003 XX YY 00:00:00 0

Parameter 003 : Run number

#RP 003
&P 003 XX YY HH:MM:SS DDDDD

X = Run number (00-99). 00 = no run open Y = Number of the next Run(00-99). H = Hours M = Minutes

S = Seconds

D = Date

Exemple: &P 003 89 90 00:00:00 0

Write a parameters (#WP) : #WP III

I = parameter ID (000 - 999)

Parameter 120 : HL975 Operating mode #WP 120 X &P 120 X

X = 0 -> PTB (Base de temps) X = 5 -> Clock (Horloge) <u>Exemple :</u>

#WP 120 5 &P 120 5

Write a command (#WC) : #WC III

Command 001 : Close a Run

#WC 001





Command 007 : Start a new synchro

#WC 007 TT HH:MM DD/XX/YY

T = Synchro Type 2 -> Manual Synchro

If Manual Synchro is chosen: H = Hour M = Minute D = Day X = Month

Y = Year

Exemple: #WC 007 02 00:00 00/00/01

Command 008 : Trigger a manual input pulse

#WC 008 XX

X = Input number ('1'-'2' for HL975)

Exemple : #WC 008 01

Liste des messages

Standard messages ID, from Device to Host :

- AK Acknowledge of a received command
- ID Serial number (optionnel)
- SN Serial number + device type + soft version (optionnel)
- OP Opening a Run
- CL Closing a Run
- TN New time
- &P Parameter

Standard messages ID (commands), from Host to Device :

- #ID Get serial number (optionnel)
- #SN Get serial number + device type (optionnel)
- #RP Read parameter
- #WP Write parameter
- #WC Write Command

<u>Échanges</u>

Démarrage d'une course

Host

Device

Pellizzoni Corentin

Projet Chrono-cross

BTS SN-IR 2018



Mode PTB

trame = "#WP 120 0"

Synchro manuel

trame = "#WC 007 02 00:00 00/00/01"

----->

----->

La synchronisation du temps sera déclenchée par l'entrée numéro 1. Il est possible de la déclencher logiciellement en envoyant la trame :

trame = "#WC 008 01"
----->
trame = "TS 00:00:00 00/00/01"
<-----</pre>

Il est aussi possible de demander le numéro de course :

trame = "#RP 003"
----->
trame = "&P 003 89 90 00:00:00 0"
<------</pre>

Ici, le numéro de course reçu est le 89. Si le numéro de course est 00, alors la course n'a pas encore été démarrée.

Temps à l'arrivée

Les temps à l'arrivée sont déclenchés par l'entrée numéro 2.



Cas d'utilisation : Démarrer une course

Manifestation : C	ross Fin Annee	BTS 1			
Course :		\sim	2		\bigcirc
	Démarrer	3	Top Départ	Terminer	5

- 1- Manifestation du jour
- 2- Liste déroulante avec toutes les courses
- 3- Démarre la course
- 4- Démarre la course à partir de l'application
- 5- Termine la course

La première partie va permettre de sélectionner une course dans la liste déroulante par rapport à la manifestation du jour. toutes les courses sont récupérés dans la base de données avec une requête SQL.Nous pouvons alors démarrer la course et la terminer.

Scénario Démarrer une course manuellement

Pour démarrer une course suite à l'action sur le bouton Démarrer une trame va être envoyée au chronomètre et va débuter la course .

La course Cross moins de 15 féminins à démarrer.



La trame #WP 120 5 va configurer le chronomètre et la trame,

#WC 007 02 00:00 00/00/01 va paramétrer le chronomètre au début de la course.

Pellizzoni Corentin



Suite à l'envoi de la trame le chronomètre affiche le début du temps annonçant le départ de la course.



Scénario Démarrer une course logiciellement

Manifestation : Cross Fin Annee BTS							
Course : Course BTS SN - 3800 m - 11:00:00			*				
Démarrer	Top Départ	Terminer					

Le bouton Top Départ va lancer le chronomètre "automatiquement"





Scénario Terminer une course

Pour terminer une course suite à l'action sur le bouton Terminer une nouvelle trame va être envoyée au chronomètre et donc va clôturer la course donc plus aucune données ne va être enregistrées par l'application.

Manifestation : Cross Fin Annee BTS			
Course : Course BTS SN - 3800 m - 11:00:00		*	
Démarrer	Top Départ	Terminer	



la trame #WC 001 va clôturer la course active



Cas d'utilisation : Chronométrer et classer les arrivées

	Arrivées		
Nb coureurs : 1	Nb Arrivées : 📿	Nb Classés : 3	
Classement	Dossard	Temps	
(4)	5	6	
			(7)
Quitter			Enregistrer

- 1- Affiche le nombre de coureurs
- 2- Affiche le nombre de coureurs arrivés
- 3- Affiche le nombre de coureurs classés
- 4- Affiche le classement des coureurs
- 5- Affiche le numéro de dossard
- 6- Affiche le temps au passage du capteur infrarouge
- 7- Enregistre

la deuxième partie va permettre de classer les arrivées des coureurs. A chaque passage devant le module infrarouge (module EC), une trame va être envoyée avec le temps puis le temps s'affiche dans la colonne 6, le numéro 2 va alors passé à 1 car il y aura eu une arrivée, ensuite il faut entrer le numéro de dossard qui associera au temps dans l'ordre.

Pour afficher le nombre de coureurs la requête SQL suivante est utilisé :

"SELECT idInscrit, NumeroDossard FROM Inscrit WHERE idCourse = "





Scénario Arrivée des coureurs

Nb Arrivées :	Nb Classés :	
Dossard	Temps	
		(with
	Nb Arrivées : Dossard	Nb Arrivées : Nb Classés : Dossard Temps

Une fois que le coureur franchit le capteur infrarouge (module EC), le chronomètre envoie une trame avec le temps correspond à la traversé du capteur infrarouge, le classement est fait dans l'ordre d'arrivée des temps, pour les dossards il faut sélectionner la case et entrez le numéro de dossard et valider et une requête SQL enverra alors le temps le numéro de dossard et le classement du coureur dans la base de données.



Pellizzoni Corentin

Projet Chrono-cross

00:00:50



Classement	Dossard	Temps
1	109	00:00:43
2	103	00:00:46
3	105	00:00:48

101

Le chronomètre va envoyer la trame suivante pour le temps:



4

La partie entourée dans la trame correspond au temps reçu au passage du capteur infrarouge. une fois le dossard associés au temps une requête SQL va envoyer les données dans la base de données avec le bouton Enregistrer



puis le deuxième élève IR va pouvoir afficher et imprimer les résultats.



<u>Recette</u>

Tests	OUI	NON
l'inscription des coureurs est possible	Х	
les associations coureurs/transpondeurs sont stockées dans la base de données	x	
les temps d'arrivée et le classement sont affichés sur l'écran	х	
les temps d'arrivée et le classement sont stockées dans la base de données		x
le démarrage d'une course est possible	X	

Bilan

Ce projet m'a permis d'aborder une première expérience d'un travail collaboratif sur la plateforme subversion et QT avec le langage C++.Cela ma permis aussi d'approcher les attentes d'une entreprise tant en qualité de communication que de gestion du temps et du travail .

A ce jour, l'application permet d'inscrire des coureurs à une course.

Démarrer et terminer une course est fonctionnel.

La base de données fonctionne parfaitement et toutes les informations peuvent être stockés. Plusieurs petites améliorations peuvent être apportées aux applications mais leurs fonctionnalités principales sont respectés par rapport au cahier des charges.



Annexes

Ressources

- <u>http://www.reliableracing.com/downloads/THCOM08.pdf</u> Chapitre 13 page 52
- http://tvaira.free.fr/projets/activites/activite-chrono-tagheuer.html

Mise en oeuvre du chronomètre TAG heuer hl975

Détection

```
$ dmesq
. . .
[12736.539217] USB Serial support registered for FTDI USB Serial Device
[12736.539450] ftdi_sio 4-2:1.0: FTDI USB Serial Device converter detected
[12736.539545] usb 4-2: Detected FT232RL
[12736.539551] usb 4-2: Number of endpoints 2
[12736.539557] usb 4-2: Endpoint 1 MaxPacketSize 64
[12736.539562] usb 4-2: Endpoint 2 MaxPacketSize 64
[12736.539567] usb 4-2: Setting MaxPacketSize 64
[12736.542416] usb 4-2: FTDI USB Serial Device converter now attached to ttyUSB0
[12736.542462] usbcore: registered new interface driver ftdi_sio
[12736.542467] ftdi_sio: v1.6.0:USB FTDI Serial Converters Driver
$ lsusb
. . .
Bus 004 Device 002: ID 0403:6001 Future Technology Devices International, Ltd FT232
USB-Serial (UART) IC
$ ls -1 /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 mars 15 14:01 /dev/ttyUSB0
Prise en charge automatique
$ sudo vim /etc/udev/rules.d/51.ttyusb.rules
# Adaptateur pour Mini Display HL 975
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", MODE="0666",
SYMLINK+="h1975"
$ ls -1 /dev/ttyUSB0
crw-rw-rw- 1 root dialout 188, 0 mars 15 14:27 /dev/ttyUSB0
pellizzonic@BTS-SN-ZUSE:~$ ls -1 /dev/h1975
lrwxrwxrwx 1 root root 7 mars 15 14:27 /dev/h1975 -> ttyUSB0
```



Programme de test du chronomètre

🛞 💿 Test Chrono TagHeuer	
Port : //dev/hl975 🛟 Transmission : RS232 🛟	Fermer
Commande : #WP 120 0	Envoyer Display Event
Ouverture du port /dev/hl975 réussie	
Mini-Display HL 975 : TN 9 2 1:00.18900 0 05D8	
Mini-Display HL 975 : TN 18 1 1:05.84300 0 05E9	
Fermeture du port /dev/hl975	
Ouverture du port /dev/hl975 reussie	
Mini-Display HL 975 : #11 0075	
Mini-Display HL 975 : #11 0075	
Mini-Display HL 975 : #11 0075	
Mini-Display HL 975 : #11 0075	
Trame envoyée : #MP 120.0 0100	
Mini-Display HL 975 : ok	
Mini-Display HL 975 : #IT 0075	
Mini-Display HL 975 : #17 0075	
Mini-Display HL 975 : #IT 0075	
Mini-Display HL 975 : #!T 0075	
Mini-Display HL 975 : #!T 0075	
Mini-Display HL 975 : #!T 0075	
Mini-Display HL 975 : #!T 0075	
Trame envoyée : #WC 007 01 01D2	
Mini-Display HL 975 : ok	
Mini-Display HL 975 : TS 00:00:35 00/00/01 0402	
Trame envoyée : #Wp 120 0 01CA	
Mini-Display HL 975 : erreur !	
Trame envoyée : #WP 120 0 01AA	
Mini-Display HL 975 : ok	
Mini-Display HL 975 : TS 00:00:00 00/00/01 03FA	N
Mini-Display HL 975 : TN 1 1 3.47900 0 0599	3
Mini-Display HL 975 : TN 2 1 4.41100 0 058D	
Mini-Display HL 975 : TN 3 1 5.35800 0 0599	
Mini-Display HL 975 : TN 1 2 10.99100 0 05A7	
Mini-Display HL 975 : TN 2 2 12.21100 0 0598	
Mini-Display HL 975 . 114 5 2 15.25400 0 05A2	