Algorithme en pseudo-code

# Note importance

Dans les pseudo-codes suivants, il existe une entité de type "AccèsServeur". Cette entité représente toutes les fonctions d'envoie au serveur. Le nom de la fonction correspond au rôle demandé au serveur.

Nous faisons le choix de ne pas décrire ces fonctions en pseudo-code car il renvoie à des mécaniques qui sont propre au système utilisé. Le pseudo-code devient ainsi plus claire et compréhensible.

# Inscription (dans l'AM)

structure InformationCoureur

String password1, password2, Tel, Email, Nom, Prenom, Photo, Ville

**inscriptionCoureur**()

**Tant que** (VRAI)

**Si** ( remplirQuestionnaire() = VRAI)

**Fin Boucle Tant Que**

**Fin Si**

**Fin Tant que**

**FinremplirQuestionnaire**() : Booléen

*//On récupère les informations données par l'utilisateur via un formulaire dans l'AM*

*//Si un champ est vide, la valeur est NULL*

InformationCoureur info = recupererInformationFormulaire()

*// On redemande à remplir le formulaire si un champ obligatoire est manquant*

**Si** ( info.prenom = NULL OU info.nom = NULL OU info.password1 = NULL OU info.password2 = NULL OU info.mail = NULL OU info.tel == NULL OU   
info.ville == NULL)

**Retourner** FAUX

**Fin Si**

*//AccèsServeur permet de communiquer avec le serveur*

AccèsServeur serveur

*// Vérification que le coureur est bien nouveau (le téléphone est la clef dans la base de donnée)*

**Si** ( serveur.coureurExiste( i.tel ) )

**Retourner** FAUX

**Fin Si**

*//Vérification des mots de passe*

**Si** (info.password1 info.password2)

**Retourner** FAUX

**Fin Si**

*//Les informations données sont bonnes et le coureur n'existe pas, le coureur est ajouté à la base de donnée et le serveur s'occupera d'envoyer un mail de confirmation au coureur*

serveur.ajouterCoureur(info)

**Retourner** VRAI

**Fin**

# Parcours coureur (dans l'AM)

structure Coureur

int tempsParcours *//représente le temps que le coureur à mis pour courir son tour*

int distanceParcouru, distanceValidée

CoordonnéeGPS dernierePosition

String villeInscrit

Les constantes suivante sont posées :

int TEMPS\_PARCOURS\_MAXIMAL, DISTANCE\_MAXIMALE,

TEMPS\_ATTENTE

parcoursCoureurDansUnTour()

Coureur c ← coureurAM() *//permet de récupérer les informations du coureur utilisant l'AM*

c.distanceParcouru ← 0

**Tant Que** ( estArrive( c ) = FAUX)

*//positionActuelle() est une méthode qui permet de récupérer les coordonnées GPS actuelle*

*// distanceGPS(CoordonnéeGPS GPS1, CoordonnéeGPS GPS2) calcule la distance entre les deux points GPS*

*// Exigence E33 : on actualise le compteur tout les kilomètres*

**Si** ( distanceGPS( c.dernierePosition, positionActuelle() > 1)

actualiserDistance( c )

*//La fonction suivante met en attente le programme pendant TEMPS\_ATTENTE*

dormir(TEMPS\_ATTENTE)

**Fin Tant Que**

**Fin**

**actualiserDistance**(Coureur c)

**Si** (c.tempsParcours > TEMPS\_PARCOURS\_MAXIMAL)

c.tempsParcours ← 0

c.distanceParcouru ← 0

**Sinon Si** (c.distanceParcours < DISTANCE\_MAXIMALE )

c.distanceParcourue ← c.distanceParcourue + 1

c.dernierePosition ← positionActuelle()

*//AccèsServeur permet de communiquer avec le serveur*

AccèsServeur serveur

serveur.actualiserCompteur(c)

**Fin Si**

**Fin**

**estArrive**(Coureur c) : Booléen

*//AccèsServeur permet de communiquer avec le serveur*

AccèsServeur serveur

**Si** (distanceGPS( c.dernierePosition, serveur.positionLigneArrive( c.villeInscrit ) > 0.25)

**Retourner** FAUX

**Sinon**

**Si** (c.tempsParcours > TEMPS\_PARCOURS\_MAXIMAL)

c.tempsParcours ← 0

c.distanceParcouru ← 0

**Retourner** FAUX

**Sinon Si** (c.distanceParcours MAXDISTANCE \* 1.15)

c.distanceValidé ← c.distanceValidé + c.distanceParcours

c.tempsParcours ← 0

c.distanceParcouru ← 0

**Retourner** VRAI

**Fin Si**

**Fin Si**

**Retourner** Faux

**Fin**

# Gestion d'évènement

## Coté administrateur (dans le SWA)

Exigence E15, E31

**creationEvenement**()

*//AccèsServeur permet de communiquer avec le serveur*

AccèsServeur serveur

Liste<String> villes ←serveur.recupererToutesLesVilles()

afficherListeVille( villes )

String villeChoisi ← recupererReponseVille()

**Si** ( serveur.estUneNouvelleVille(villeChoisi) )

// Requete de Base de donnée avec *valeurs incohérentes pour les autres champs*

serveur.creerEvenement( villeChoisi )

**Fin Si**

Liste< Coureur > coureurs ← serveur.coureurEligibleOrganisateur( villeChoisi )

afficherListeCoureur( coureurs )

Coureur coureurChoisi ← recupererReponseCoureur()

serveur.promouvoir( villeChoisi, coureurChoisi ) *//l'ancien organisateur, s'il existe, redevient simple coureur et le coureur C devient l'organisateur de la ville V)*

**Fin**

## Coté organisateur (dans le SWP)

Exigence E30 et E30.1

**modificationEvenement**() {

*//AccèsServeur permet de communiquer avec le serveur*

AccèsServeur serveur

Coureur c ← recupererInformationCompte()

*// Evenement possède toutes les caractéristique présent dans la Base de donnée,*

se référer au schéma de base de donné

Evenement evenementOrganise ← serveur.evenement( c.tel )

Boolean bienDefini ← VRAI

**Tant que** (Evenement.heureDebut n'est pas défini

OU Evenement.heureFin n'est pas défini

OU Evenement.date n'est pas défini

OU bienDefini = FAUX)

Evenement.heureDebut ← recupererHeureDebut()

Evenement.heureFin ← recupererHeureFin()

**Si** (Evenement.heureDebut < Evenement.heureFin )

bienDefini ← FAUX

**Sinon**

bienDefini ← VRAI

**Fin Si**

Evenement.date ← recupererDate()

**Fin Si**

Date aujourdhui ← serveur.onEstQuelJour()

Heure heureActuelle ← serveur.ilEstQuelHeure()

**Si** ( Evenement.date = aujourdhui

ET Evenement.heureDebut < heureActuelle < Evenement.heureFin )

*// Afficher une alerte signalant que l'organisateur doit attendre la fin de l'événement avant de pouvoir changer ses propriétés*

afficherAlerteImpossible()

*//Remplissage d'un formulaire permettant de modifier seulement le champ Information*

Evenement.information ← recupererFormulaireInformation()

**Sinon**

*// Remplissage d'un formulaire permettant de modifier toutes les propriétés SAUF la ville de l'événement et l'organisateur.*

Evenement ← recupererFormulaireEvenement()

**Fin Si**

*//Requête Base de Donnée pour modifierEvenement*

serveur.modificationEvenement ( Evenement )

**Fin**