Dossier technique de projet

-

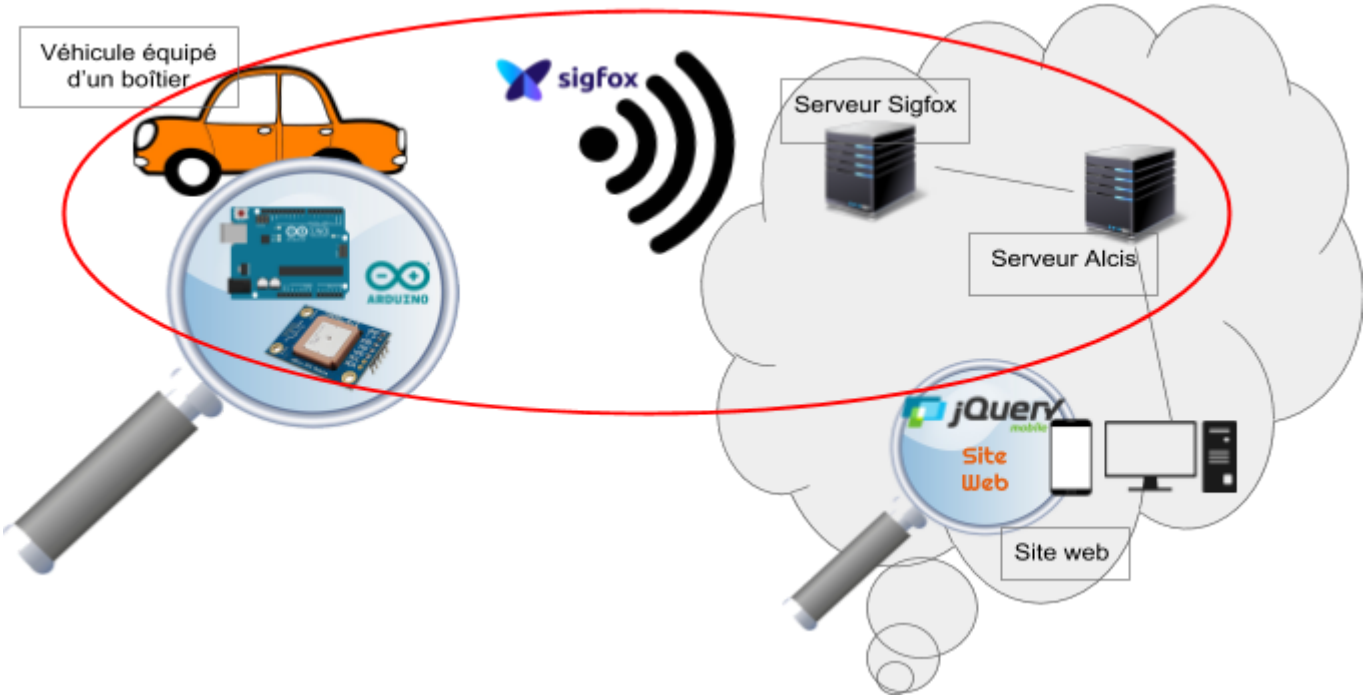
BTS SN-IR 2018

CROS Alexandre

Lycée international Victor Hugo - Colomiers

Sommaire

Identification du travail



Au sein du projet, mon travail est majoritairement situé autour de la base de données sur le serveur Alcis qui contiendra les informations envoyées depuis le serveur Sigfox et qui en permettra l'affichage sur le site web.
Depuis il s'agira de coder le tramage et détramage des messages côté boîtier (en C++) et du côté serveur (en Java).

Les données temps réel

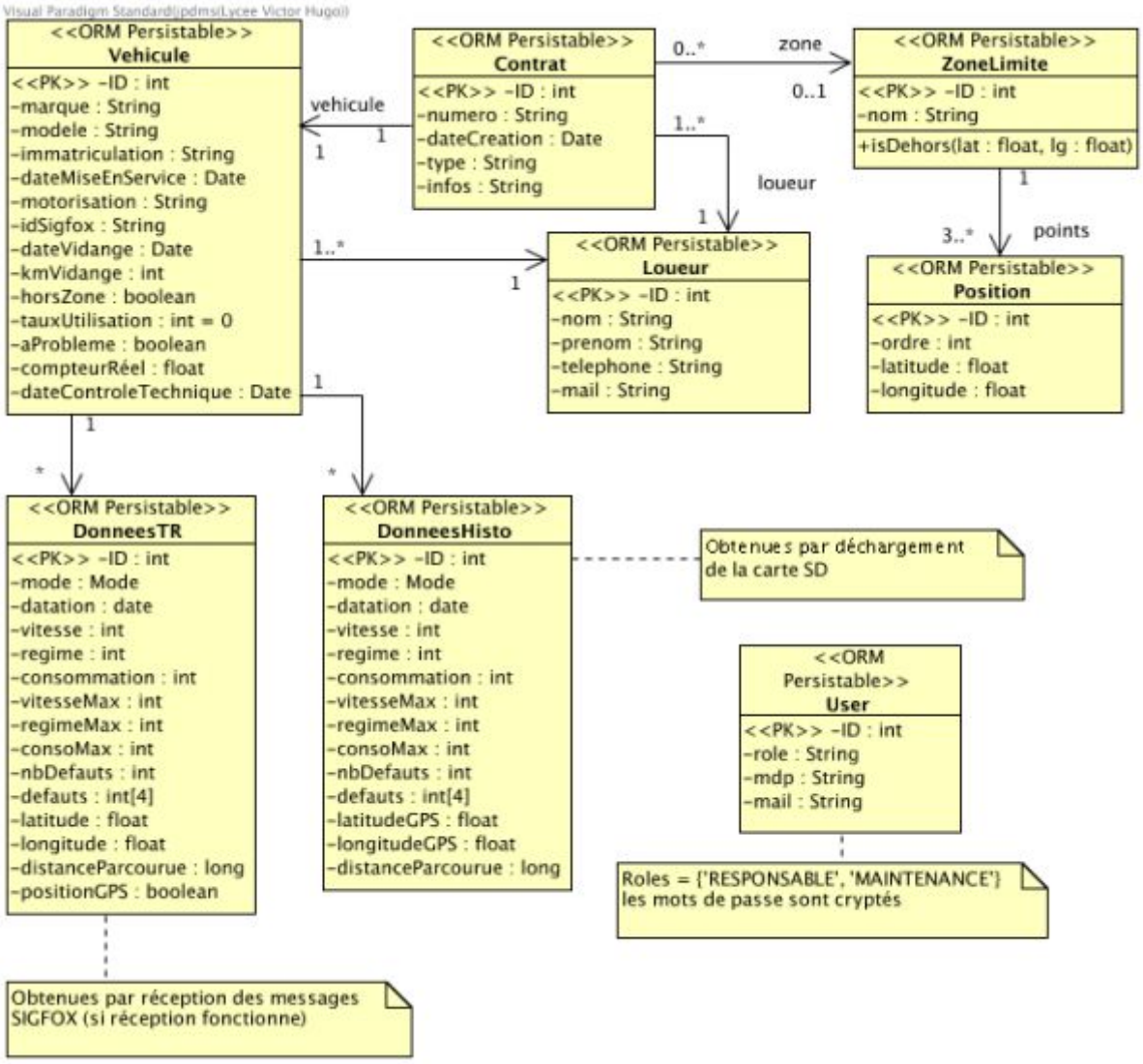
La première étape est donc la détermination des données temps réel à récupérer afin de concevoir la base de données.
Celles-ci sont donc choisies d'après le cahiers des charges et sont les suivantes :

Données temps réel	Description
Datation	Date et heure de réception de la donnée
Distance parcourue	Distance parcourue depuis la dernière réinitialisation du boîtier
Consommation	Consommation instantanée au moment de la capture de donnée
Consommation max	Consommation max dans l'intervalle de deux messages
Consommation moyenne	Consommation max dans l'intervalle de deux messages

Nombre de défauts	Nombre de défauts sur le véhicule (Jusqu'à 4)
Code de défaut 1	1er code de défaut
Code de défaut 2	2e code de défaut
Code de défaut 3	3e code de défaut
Code de défaut 4	4e code de défaut
Vitesse	Vitesse instantanée au moment de la capture de donnée
Vitesse max	Vitesse max dans l'intervalle de deux messages
Vitesse moyenne	Vitesse max dans l'intervalle de deux messages
Régime	Régime instantané au moment de la capture de donnée
Régime max	Régime max dans l'intervalle de deux messages
Régime moyenne	Régime max dans l'intervalle de deux messages

La base de donnée

Conception de la BD



Ce diagramme correspond à la base de données côté serveur.
Il va falloir développer des classes interfaces pour chacune de ces tables.
Au moment du contrat on choisit une zone autorisée prédéfinie dans une liste.
Les DonneesTR et DonneesHisto sont structurellement identiques mais obtenues respectivement depuis le serveur Sigfox et depuis la carte SD équipée dans le boîtier.

Vehicule.aProbleme donne une indication sur un véhicule qui suivant des critères à définir pose soucis.
Les User sont deux pour l'instant ; le responsable et la maintenance.
A chaque achat ou vente de véhicule la BD est mise à jour.
A chaque création ou fin de contrat, la BD est mise à jour.

Création du schéma de la BD

Afin de créer de manière rapide et simplifier le script schéma SQL permettant l'initialisation de la BD on utilise l'outil de développement UML Visual Paradigm. En effet ce dernier permet la création d'un schéma SQL à partir d'une diagramme UML de base de donnée.
Un extrait du schéma SQL permettant d'initialiser la BD Alfox est en annexe 1.

Conception des classes d'interface

Afin de gérer les données de la base de données et notamment les afficher sur le site web, on utilise des classes d'interface qui permettent de manipuler.
Chacune des classes d'interface seront construire à partir d'un schéma repris du projet antérieur SnDiscovery, projet de science participative. Elles seront par la suite adaptées en fonction des besoins du projet et de l'IHM.

Le schéma des classes d'interface est composé des méthodes suivantes :

Méthodes	Rôle
constructeur()	Crée un objet
create()	Crée une nouvelle entrée dans la BD à partir d'un objet
save()	Met à jour une entrée à partir d'un objet
delete()	Supprime une entrée de la table
getByChamp()	Récupère une ou des entrées en fonction d'une champ et en crée un ou des objets
creerParRequete()	Crée un objet à partir de résultat d'une commande SQL
assesseurs	Getters et setters qui permettent de récupérer la valeur d'un attribut de l'objet ou de la modifier.

Méthode create() et classe DonneesTR

La méthode create() de la classe DonneesTR est particulière. En effet en fonction du mode du boîtier, le message Sigfox ne sera pas détramé de la même manière.
Les trames des messages suivent le modèle suivant défini au préalable :

Mode du boîtier	Trame du message
NORMAL	TM K1 K2 K3 CD CD VX VM RX RM CX CM
GPS	TM K1 K2 K3 L1 L2 L3 L4 G1 G2 G3 G4
DMD_GPS	TM K1 K2 K3 L1 L2 L3 L4 G1 G2 G3 G4
DEGRADE	TM K1 K2 K3 00 00 00 00 00 00 00 00
MAINTENANCE	TM K1 K2 K3 00 00 00 00 00 00 00 00
DORMIR	TM K1 K2 K3 00 00 00 00 00 00 00 00
INIT	TM K1 K2 K3 00 00 00 00 00 00 00 00

La légende des trames est la suivante :

Nom	Description	Octets	Valeurs	Définition
TM	Type du message + Etat de la communication + Nombre de défauts	(1octet) 4 bits 2 bits 1 bit	0 à 255 0 à 15 0 à 003 0 à 001	Poids fort : (NORMAL, DEGRADE, DMD_GPS, GPS, MAINTENANCE, DORMIR, INIT) Poids faible haut :(BLUETOOTH_OFF, BLUETOOTH_ON, OBD2_OFF, OBD2_ON) Poids faible bas : Si il y a un défaut, 0 sinon
CD	Code défaut	2 quartets	0 à 15	(permet de transmettre 4 défauts dans la même trame)
KM	Kilométrage	3 octets	0 à 999 999	(en km)
CX	Consommation max	1 octet	0 à 255	(en dl/100km : max sur 10 mn)
CM	Consommation moyenne	1 octet	0 à 255	(en dl/100km : moyenne calculée sur 10 mn)
VX	Vitesse max	1 octet	0 à 255	(en km/h : max sur 10 mn)
VM	Vitesse moyenne	1 octet	0 à 255	(en km/h : moyenne calculée sur 10 mn)
RX	Régime moteur max	1 octet	0 à 255	(en t/100mn : max sur 10 mn)
RM	Régime moteur moyen	1 octet	0 à 255	(en t/100mn : moyenne calculée sur 10 mn)

Il existe donc trois cas de détramage possibles :

- Mode NORMAL
- Mode GPS, DMD_GPS
- Mode DEGRADE, MAINTENANCE, DORMIR, INIT

La méthode create de la classe DonneesTR se trouve donc en annexe 2.

Les tests unitaires JUnit des classes d'interface

Afin de vérifier le bon fonctionnement des méthodes de classes, on utilise le framework de test unitaire JUnit.
Le but est de tester les valeurs retournées, si elles correspondent à celle de la base de données alors les tests sont validés.

Les méthodes sont testées de la sorte :
(Ci-dessous la méthode create() de la classe Vehicule)

```
public void testCreate() throws Exception {
    System.out.println("create");
    // Initialisation de la connexion SQL
    Connection con = ConnexionMySQL.newConnexion();
    // Attribution de valeurs aux attributs de l'objet
    String marque = "Citroën";
    String modele = "DS5";
    String immatriculation = "DD-000-EE";
    Timestamp dateMiseEnService = Utils.stringToTimestamp("2018/03/26 00:00:00");
    String motorisation = "Diesel";
    String idSigfox = "0123";
    Timestamp dateVidange = Utils.stringToTimestamp("2018/05/26 00:00:00");
    int kmVidange = 0;
    boolean horsZone = false;
    int tauxUtilisation = 0;
    boolean aProbleme = false;
    double compteurReel = 50.0;
    Timestamp dateControleTechnique = Utils.stringToTimestamp("2020/03/26 00:00:00");
    // Appel de la méthode pour créer une entrée
    Vehicule result = Vehicule.create(con, marque, modele, immatriculation,
        dateMiseEnService, motorisation, idSigfox, dateVidange, kmVidange, horsZone,
        tauxUtilisation, aProbleme, compteurReel, dateControleTechnique);
    // On vérifie que le modèle correspond à la valeur donnée plus tôt
    assertEquals("DS5", result.getModele());
    // On supprime l'entrée créée à la fin du test
    result.delete(con);
    }
```

L'extrait de la fiche de test correspondant à l'exécution de la méthode create() ci-dessus est le suivant :

Description du test

Scénarios concernés	On teste les valeurs retournées par les méthodes.
Description	Cette classe sert à obtenir les données comme la date de la dernière donnée pour un véhicule, la liste des données pour un véhicule et pour une journée, l'âge moyen de la flotte, la consommation moyenne de la flotte, le kilométrage moyen de la flotte. Elle permet également de vérifier si un véhicule est en dehors de sa zone et depuis de temps il se trouve au garage le cas échéant.
Environnement nécessaire	IDE Netbeans
Situation initiale	La base données est créée et comporte des données.
Classe de test nécessaire	Vehicule.java
Nom du script	Vehicule.jaba

Description du script

N°	Traitement	Paramètres en entrée	Résultats attendus
1	Appel de la méthode testCreate()	Données d'un véhicule	Une entrée est bien crée dans la table 'vehicule'

Pour valider ses tests unitaires, la base de données est remplie dans un premier par des données qui ne sont pas réalistes et ne correspondent pas aux données de l'entreprise.

Adaptation des classes à l'IHM

Les classes d'interface étant créées à partir d'un schéma unique, il s'agit maintenant de créer des méthodes permettant d'adapter ses classes à l'IHM.

Les méthodes nécessaires ajoutés sont les suivantes :

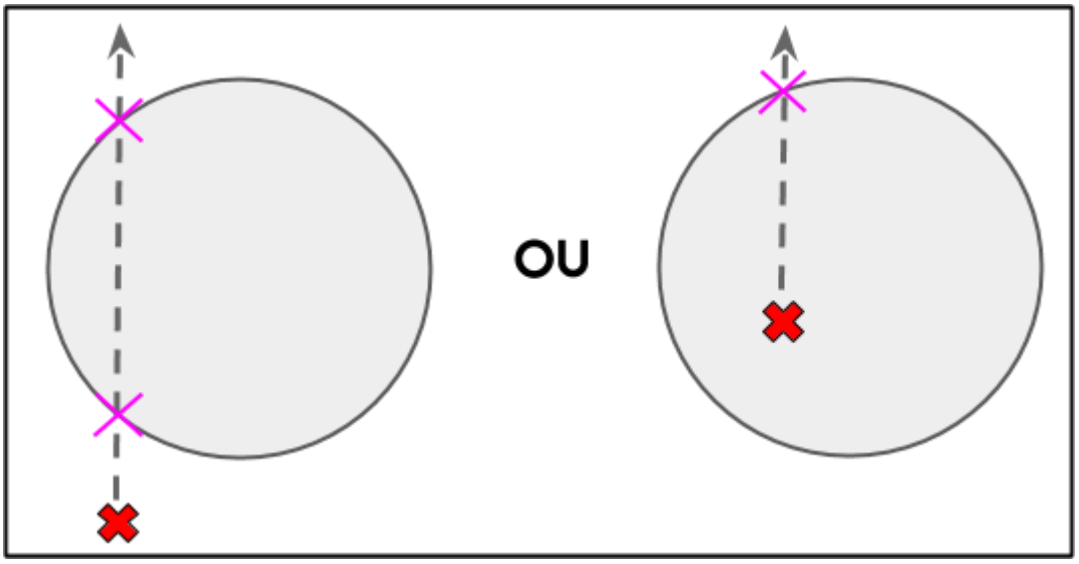
Classe	Vehicule	DonneesTR
Donnée(s) retournée(s) par la méthode	Kilométrage moyen de la flotte	Dernière donnée enregistrée d'un véhicule
	Kilométrage mensuel moyen de la flotte	Liste des données d'un même jour pour un véhicule
	Conso moyenne de la flotte	Liste de toutes les données pour un véhicule
	Conso mensuelle moyenne de la flotte	Détramer les messages et enregistrer les données dans la BD
	Âge moyen de la flotte	
	Le véhicule est-il hors zone ?	
	Nombre de véhicule(s) hors zone	
	Temps passé au garage Alcis	

Les deux méthodes les plus intéressantes sont les méthodes qui retournent si un véhicule est hors zone (isDehors()) et le temps passé au garage Alcis (getTempsAlcis()).

Méthode isDehors()

La méthode isDehors() vérifie si un véhicule est dans sa zone de transit et retourne 'false' si c'est le cas. La méthode retourne 'true' si le véhicule est hors zone.

Le fonctionnement suit le schéma suivant :



On récupère la position GPS obtenue par le service de localisation Sigfox.
On fait “avancer” le point dans une même direction.
Si le point traverse un nombre de fois pair la frontière de la zone limite alors cela signifie qu’il est à l’extérieur. Si le nombre de traversée est impair alors le point est dans la zone.
Le code correspondant à cette méthode est écrit en annexe 3.

Méthode saveData()

La méthode saveData() est appelé par la callback et permet le détramage des messages et l’enregistrement des données dans la BD.
Cette méthode modifie la valeur de chaque attribut de l’objet DonneesTR avec les valeurs des octets correspondant.
Ensuite elle appelle la méthode create() afin de créer une nouvelle entrée dans la table ‘donneesTR’.
De la même manière que la méthode create(), le détramage est différent en fonction du mode du boîtier.

Signe de la Longitude

Comme indiqué page 6, en mode GPS et DMD_GPS, la latitude et la longitude sont codées sur 4 octets chacune. Il a donc été choisi de donner 6 décimales de précision après la virgule.
On a donc au plus 8 chiffres pour la latitude (-90 à +90°) et 9 pour la longitude (-180° à +180°).
De cette manière on utilise un octet pour coder 2 chiffres, et un octet pour les 3 chiffres avant virgule de la longitude.
Il s’agit donc de trouver une solution afin de coder le signe positif ou non en binaire.

La méthode retenue est telle qu’on utilise la sixième décimale après la virgule. En effet cette dernière correspond à plus ou moins 1 centimètre de précision GPS.

Ainsi lorsque le signe sera positif, la sixième décimale sera paire. Si le signe est négatif, elle sera impaire.

On peut donc avoir les exemples similaires :

Signe avant traitement	Valeur de la sixième décimale	Signe souhaité	Valeur de la sixième décimale après traitement
+	6	-	7
-	5	+	6

Le code de cette astuce est situé en annexe 4.

Méthode getTempsAlcis()

Le but de la méthode getTempsAlcis() est de retourner le temps passé au garage Alcis par un véhicule.
En effet, il peut sembler important pour le personnel de maintenance de voir combien de véhicules sont présent et depuis combien de temps.

Pour ce faire, on utilise donc la position fournie pour chaque message Sigfox (avec un précision de 5 kilomètres).
On va donc vérifier si un véhicule se trouve dans une zone de 5 kilomètres autour du garage.
Si c'est le cas on récupère la dernière date à laquelle le véhicule ne s'y trouvait pas et on la compare avec la date actuelle. Le temps obtenu sera donc le temps passé dans la zone du garage.

Cependant, afin d'éviter que le véhicule soit indiqué comme présent au garage après un simple passage dans la zone, on vient vérifier que le temps retourner par la méthode est supérieur à 1 heure. Si c'est le cas on considère qu'il est présent et la méthode retourne la valeur de temps.
Dans le cas contraire on considère qu'il n'est pas présent au garage et la méthode retournera 0.

Le code de cette méthode est présent en annexe 5, cependant à l'heure actuelle celle-ci comporte une erreur et retourne une valeur de temps qui ne correspond pas aux données temps réel de la base de données.

Annexe 1 - Script schéma SQL

```
drop schema if exists alfox;
create schema alfox;
use alfox;

create table donneesHisto (
    ……
) ;

create table donneesTR (
    ID int(10) not null auto_increment,
    SeqNumber int(10),
    Mode varchar(255),
    Datation datetime,
    Vitesse int(10),
    Regime int(10),
    Consommation int(10),
    VitesseMax int(10),
    RegimeMax int(10),
    ConsoMax int(10),
    NbDefauts int(10),
    Defaut1 int(10),
    Defaut2 int(10),
    Defaut3 int(10),
    Defaut4 int(10),
    Latitude decimal(9,6),
    Longitude decimal(9,6),
    DistanceParcourue bigint(20),
    Snr decimal(5,2),
    Rssi decimal(5,2),
    AvgSnr decimal(5,2),
    Device varchar(255),
    primary key (ID)
) ;

create table vehicule (
    ID int(10) not null auto_increment,
    Marque varchar(255) not null,
    Modele varchar(255) not null,
    Immatriculation varchar(255) not null unique,
    DateMiseEnService datetime not null,
    Motorisation varchar(255) not null,
    IdSigfox varchar(255) unique,
    DateVidange datetime,
```


```
    KmVidange int(10),
    HorsZone tinyint(1) not null,
    TauxUtilisation int(10) not null,
    AProbleme tinyint(1) not null,
    CompteurReel decimal(9,3) not null,
    DateControleTechnique datetime not null,
    primary key (ID)
) ;

create table zoneLimite (
    ...
) ;

create table position (
    ...
) ;


create table loueur (
    ...
) ;

create table contrat (
    ...
) ;

create table `user` (
    ...
) ;

alter table donneesHisto
    add index FKDonneesHis493650 (VehiculeID),
    add constraint FKDonneesHis493650 foreign key (VehiculeID) references vehicule
(ID);

alter table donneesTR
    add index FKDonneesTR404677 (Device),
    add constraint FKDonneesTR404677 foreign key (Device) references vehicule
(IdSigfox);

alter table contrat
    add index FKContrat5291 (ZoneLimiteID),
    add constraint FKContrat5291 foreign key (ZoneLimiteID) references zoneLimite
(ID);
```


```
alter table position
    add index FKPosition217606 (ZoneLimiteID),
    add constraint FKPosition217606 foreign key (ZoneLimiteID) references zoneLimite
(ID);

alter table contrat
    add index FKContrat433214 (Device),
    add constraint FKContrat433214 foreign key (Device) references vehicule (ID);

alter table contrat
    add index FKContrat81038 (LoueurID),
    add constraint FKContrat81038 foreign key (LoueurID) references loueur (ID);

alter table loueur
    add constraint nomprenom unique (Nom, Prenom);
```


Annexe 2 – Méthode create() de la classe DonneesTR

```
static public DonneesTR create(Connection con, String mode, Timestamp datation,
            int vitesse, int regime, int consommation, int vitesseMax, int regimeMax,
            int consoMax, int nbDefauts, int defaut1, int defaut2, int defaut3,
            int defaut4, double latitude, double longitude,
            long distanceParcourue, int seqNumber, double snr, double rssi,
            double avgSnr, String device) throws Exception {
        DonneesTR donneesTR = new DonneesTR(mode, datation, vitesse, regime,
                consommation, vitesseMax, regimeMax, consoMax, nbDefauts, defaut1,
                defaut2, defaut3, defaut4, latitude, longitude, distanceParcourue,
                seqNumber, snr, rssi, avgSnr, device);

        String queryString;
        if (mode == "NORMAL") {
            queryString
                    = "insert into donneesTR (Mode, Datation, Vitesse, Regime, "
                    + "Consommation, VitesseMax, RegimeMax, ConsoMax, NbDefauts, "
                    + "Defaut1, Defaut2, Defaut3, Defaut4, Latitude, Longitude, "
                    + "DistanceParcourue, SeqNumber, Snr, Rssi, AvgSnr, Device) "
                    + "values ("
                    + Utils.toString(mode) + ", "
                    + Utils.toString(datation) + ", "
                    + Utils.toString(vitesse) + ", "
                    + Utils.toString(regime) + ", "
                    + Utils.toString(consommation) + ", "
                    + Utils.toString(vitesseMax) + ", "
                    + Utils.toString(regimeMax) + ", "
                    + Utils.toString(consoMax) + ", "
                    + Utils.toString(nbDefauts) + ", "
                    + Utils.toString(defaut1) + ", "
                    + Utils.toString(defaut2) + ", "
                    + Utils.toString(defaut3) + ", "
                    + Utils.toString(defaut4) + ", "
                    + Utils.toString(latitude) + ", "
                    + Utils.toString(longitude) + ", "
                    + Utils.toString(distanceParcourue) + ", "
                    + Utils.toString(seqNumber) + ", "
                    + Utils.toString(snr) + ", "
                    + Utils.toString(rssi) + ", "
                    + Utils.toString(avgSnr) + ", "
                    + Utils.toString(device)
                    + ")";
```


```
    } else if ((mode == "DMD_GPS") || (mode == "GPS")) {
        queryString
                = "insert into donneesTR (Mode, Datation, Vitesse, Regime, "
                + "Consommation, VitesseMax, RegimeMax, ConsoMax, NbDefauts, "
                + "Defaut1, Defaut2, Defaut3, Defaut4, Latitude, Longitude, "
                + "DistanceParcourue, SeqNumber, Snr, Rssi, AvgSnr, Device) "
                + "values ("
                + Utils.toString(mode) + ", "
                + Utils.toString(datation) + ", "
                + "NULL,NULL,"
                + "NULL,NULL,NULL,NULL,NULL,"
                + "NULL,NULL,NULL,NULL,"
                + Utils.toString(latitude) + ", "
                + Utils.toString(longitude) + ", "
                + Utils.toString(distanceParcourue) + ", "
                + Utils.toString(seqNumber) + ", "
                + Utils.toString(snr) + ", "
                + Utils.toString(rssi) + ", "
                + Utils.toString(avgSnr) + ", "
                + Utils.toString(device)
                + ")";
    } else {
        queryString
                = "insert into donneesTR (Mode, Datation, Vitesse, Regime, "
                + "Consommation, VitesseMax, RegimeMax, ConsoMax, NbDefauts, "
                + "Defaut1, Defaut2, Defaut3, Defaut4, Latitude, Longitude, "
                + "DistanceParcourue, SeqNumber, Snr, Rssi, AvgSnr, Device) "
                + "values ("
                + Utils.toString(mode) + ", "
                + Utils.toString(datation) + ", "
                + "NULL,NULL,"
                + "NULL,NULL,NULL,NULL,NULL,"
                + "NULL,NULL,NULL,NULL,NULL,NULL,"
                + Utils.toString(distanceParcourue) + ", "
                + Utils.toString(seqNumber) + ", "
                + Utils.toString(snr) + ", "
                + Utils.toString(rssi) + ", "
                + Utils.toString(avgSnr) + ", "
                + Utils.toString(device)
                + ")";
    }
    Statement lStat = con.createStatement();
    lStat.executeUpdate(queryString, Statement.NO_GENERATED_KEYS);
    return donneesTR;
}
```


```
public static void saveData(Connection con, String data,
        Timestamp datation, double latitude, double longitude,
        String device, int seqNumber, float snr, float rssi, float avgSnr)
        throws Exception {
    // conversion de la chaine hexa en tableau de byte
    HexBinaryAdapter adapter = new HexBinaryAdapter();
    byte[] bData = adapter.unmarshal(data);

    int vitesseMoy = 0, regimeMoy = 0, consoMoy = 0, vitesseMax = 0,
            regimeMax = 0, consoMax = 0;
    int defaut1 = 0, defaut2 = 0, defaut3 = 0, defaut4 = 0;
    long distanceParcourue = 0L;

    // décodage du 1er octet TM
    boolean bluetoothActif = false;
    bluetoothActif = (bData[0] & 0x80) == 0x80;
    boolean OBD2Actif = false;
    OBD2Actif = (bData[0] & 0x40) == 0x40;
    int nbDefauts = 0;
    boolean isDefaut = false;
    isDefaut = (bData[0] & 0x10 >> 4) != 0;
    String mode = "";
    mode = sMode[bData[0] & 0x0F];
    if (mode == "ERREUR") {
        // rien à faire
    } else if (mode == "NORMAL") {
        // NORMAL : TM K1 K2 K3 CD CD VX VM RX RM CX CM
        distanceParcourue = bData[1] * 10000 + bData[2] * 100 + bData[3];
        defaut1 = (bData[4] & 0xF0) >> 4;
        defaut2 = bData[4] & 0x0F;
        defaut3 = (bData[5] & 0xF0) >> 4;
        defaut4 = bData[5] & 0x0F;
        nbDefauts = ((defaut1 != 0) ? 1 : 0) + ((defaut2 != 0) ? 1 : 0)
                + ((defaut3 != 0) ? 1 : 0) + ((defaut4 != 0) ? 1 : 0);
        vitesseMax = bData[6];
        if (vitesseMax < 0) {
            vitesseMax = 256 + vitesseMax;
        }
        vitesseMoy = bData[7];
        if (vitesseMoy < 0) {
            vitesseMoy = 256 + vitesseMoy;
        }
        regimeMax = bData[8] * 100;
        regimeMoy = bData[9] * 100;
```


```
        consoMax = bData[10];
        if (consoMax < 0) {
            consoMax = 256 + consoMax;
        }
        consoMoy = bData[11];
        if (consoMoy < 0) {
            consoMoy = 256 + consoMoy;
        }
        DonneesTR donneesTR = create(con, mode, datation,
                vitesseMoy, regimeMoy, consoMoy, vitesseMax, regimeMax, consoMax,
                nbDefauts, defaut1, defaut2, defaut3, defaut4,
                latitude, longitude, distanceParcourue,
                seqNumber, snr, rssi, avgSnr, device);
    } else if ((mode == "DEGRADE") || (mode == "INIT") || (mode == "DORMIR")) {
        // TM K1 K2 K3 00 00 00 00 00 00 00 00
        distanceParcourue = bData[1] * 10000 + bData[2] * 100 + bData[3];
        DonneesTR donneesTR = create(con, mode, datation,
                vitesseMoy, regimeMoy, consoMoy, vitesseMax, regimeMax, consoMax,
                nbDefauts, defaut1, defaut2, defaut3, defaut4,
                latitude, longitude, distanceParcourue,
                seqNumber, snr, rssi, avgSnr, device);
    } else if ((mode == "DMD_GPS") || (mode == "GPS")) {
        // DMD_GPS : TM K1 K2 K3 LA LA LA LA LO LO LO LO
        boolean negLat = false;
        boolean negLg = false;

        distanceParcourue = bData[1] * 10000 + bData[2] * 100 + bData[3];

        // gestion du signe des données GPS.
        if ((bData[7] & 0x01) != 0) {
            bData[7] &= 0xFE;
            negLat = true;
        }
        if ((bData[11] & 0x01) != 0) {
            bData[11] &= 0xFE;
            negLg = true;
        }
        double lat = (float) bData[4] + (float) bData[5] / 100
                + (float) bData[6] / 10000 + (float) bData[7] / 1000000;
        if (negLat) {
            lat = -lat;
        }
        double lg = (float) bData[8] + (float) bData[9] / 100
                + (float) bData[10] / 10000 + (float) bData[11] / 1000000;
        if (negLg) {
```


```
                lg = -lg;
            }
            DonneesTR donneesTR = create(con, mode, datation,
                    vitesseMoy, regimeMoy, consoMoy, vitesseMax, regimeMax, consoMax,
                    nbDefauts, defaut1, defaut2, defaut3, defaut4,
                    lat, lg, distanceParcourue,
                    seqNumber, snr, rssi, avgSnr, device);
        }
    }
```


Annexe3 - Méthode isDehors()

```
// Obligation de passer par la méthode getLastDatation() avant
// Met à jour horsZone
public boolean isDehors(Connection con) throws Exception {
    //Récupération du tableau de position de la zone associée par le contrat au véhicule
    String queryString = "select * from position, zoneLimite where  ZoneLimiteID =
            zoneLimite.ID and Nom = (select Nom from contrat, vehicule, zoneLimite where
            Device = vehicule.IdSigfox and ZoneLimiteID = zoneLimite.ID and
            Immatriculation='" + immatriculation + "') order by Ordre";
    Statement lStat = con.createStatement( //peut générer une exception si problème avec
            la requête SQL
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);
    ResultSet lResult = lStat.executeQuery(queryString);
    // On met les points dans 2 collections Arraylist de Dloat
    // Dloat avec un D majuscule est une classe !)
    ArrayList<Double> xap = new ArrayList<>();
    ArrayList<Double> yap = new ArrayList<>();
    while (lResult.next()) {
        xap.add(lResult.getDouble("Latitude"));
        yap.add(lResult.getDouble("Longitude"));
    }
    // On transforme les collections en tableaux d'objets
    int nbPoints = xap.size();
    Double[] xtp = xap.toArray(new Double[0]);
    Double[] ytp = yap.toArray(new Double[0]);

    // Les tableaux sont maintenant des tableaux de doubles !
    // Récupération de la dernière latitude et longitude enregistrée
    String queryString2 = "select Latitude, Longitude from donneesTR, vehicule where
            Device = vehicule.IdSigfox and vehicule.Immatriculation = \""
            + this.immatriculation + "\" order by Datation desc limit 1;";
    Statement lStat2 = con.createStatement(
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);
    ResultSet req = lStat2.executeQuery(queryString2);
    double latitude = 0;
    double longitude = 0;
    // Si il y a une donnée on récupère la latitude et longitude
    if (req.next()) {
        latitude = req.getDouble("Latitude");
        longitude = req.getDouble("Longitude");
    }// Sinon on génère une exception
    else {
```


```
        throw new Exception("Aucune donnée TR");
    }
    // On vérifie que le point se situe dans la zone

    int i, j;
    boolean isDehors = false;
    for (i = 0, j = nbPoints - 1; i < nbPoints; j = i++) {
        if ((((ytp[i] <= longitude) && (longitude < ytp[j])) || ((ytp[j] <= longitude)
&&  ---------------  (longitude < ytp[i]))) && (latitude < (xtp[j] - xtp[i]) * (longitude
-  ------------------  ytp[i]) / (ytp[j] - ytp[i]) + xtp[i])) {
            isDehors = !isDehors;
        }
    }
    // On met à jour horsZone avec la valeur donnée par la méthode isDehors()
    horsZone = isDehors;
    return !isDehors;
}
```


Annexe 4 – saveData() et signes des positions GPS

```
public static void saveData(Connection con, String data, Timestamp datation,
        double latitude, double longitude, String device, int seqNumber, float snr,
        float rssi, float avgSnr) throws Exception {
    // conversion de la chaine hexa en tableau de byte
    HexBinaryAdapter adapter = new HexBinaryAdapter();
    byte[] bData = adapter.unmarshal(data);

    int vitesseMoy = 0, regimeMoy = 0, consoMoy = 0, vitesseMax = 0,
        regimeMax = 0, consoMax = 0;
    int defaut1 = 0, defaut2 = 0, defaut3 = 0, defaut4 = 0;
    long distanceParcourue = 0L;

    // décodage du 1er octet TM
    boolean bluetoothActif = false;
    bluetoothActif = (bData[0] & 0x80) == 0x80;
    boolean OBD2Actif = false;
    OBD2Actif = (bData[0] & 0x40) == 0x40;
    int nbDefauts = 0;
    boolean isDefaut = false;
    isDefaut = (bData[0] & 0x10 >> 4) != 0;
    String mode = "";
    mode = sMode[bData[0] & 0x0F];
    if (mode == "ERREUR") {
        // rien à faire
    } else if (mode == "NORMAL") {
        // NORMAL : TM K1 K2 K3 CD CD VX VM RX RM CX CM
        // ...Traitement...
    } else if ((mode == "DEGRADE") || (mode == "INIT") || (mode == "DORMIR")) {
        // TM K1 K2 K3 00 00 00 00 00 00 00 00
        // ...Traitement...
    } else if ((mode == "DMD_GPS") || (mode == "GPS")) {
        // DMD_GPS : TM K1 K2 K3 LA LA LA LA LO LO LO LO
        boolean negLat = false;
        boolean negLg = false;

        distanceParcourue = bData[1] * 10000 + bData[2] * 100 + bData[3];

        // gestion du signe des données GPS.
        if ((bData[7] & 0x01) != 0) {
            bData[7] &= 0xFE;
            negLat = true;
```


```
        }
        if ((bData[11] & 0x01) != 0) {
            bData[11] &= 0xFE;
            negLg = true;
        }
        double lat = (float) bData[4] + (float) bData[5] / 100
                + (float) bData[6] / 10000 + (float) bData[7] / 1000000;
        if (negLat) {
            lat = -lat;
        }
        double lg = (float) bData[8] + (float) bData[9] / 100
                + (float) bData[10] / 10000 + (float) bData[11] / 1000000;
        if (negLg) {
            lg = -lg;
        }
        DonneesTR donneesTR = create(con, mode, datation, vitesseMoy, regimeMoy,
            consoMoy, vitesseMax, regimeMax, consoMax, nbDefauts, defaut1, defaut2,
            defaut3, defaut4, lat, lg, distanceParcourue, seqNumber, snr, rssi, avgSnr,
            device);
    }
}
```


Annexe 5 - Méthode getTempsAlcis()

```
public static int getTempsAlcis(Connection con, String idSigfox) throws Exception {
    long tempsAlcisEnMs = 0;
    int tempsAlcisEnM = 0;
    // On récupère la date et heure actuelle
    Date now = new java.util.Date();
    // On défini la position d'ALCIS (5km*5km)
    double latMin = 43.555692;
    double latMax = 43.646065;
    double lgMin = 1.465021;
    double lgMax = 1.591707;
    // Récupération des donnéesTR associées au véhicule passé en paramètre
    ArrayList<DonneesTR> lstDonneesTR = DonneesTR.getDonneesVehicule(con, idSigfox);
    // On regarde si la dernière donnée donne le véhicule chez Alcis
    double lat0 = lstDonneesTR.get(0).getLatitude();
    double lg0 = lstDonneesTR.get(0).getLongitude();
    if ((lat0 >= latMin) && (lat0 <= latMax) && (lg0 >= lgMin) && (lg0 <= lgMax)) {
        // Si oui on récupère la dernière date à laquelle le véhicule était en dehors
        d'Alcis
        int i = 1;
        while ((i < lstDonneesTR.size()) && (lstDonneesTR.get(i).getLatitude() >=
latMin)            && (lstDonneesTR.get(i).getLatitude() <= latMax)
            && (lstDonneesTR.get(i).getLongitude() >= lgMin)
            && (lstDonneesTR.get(i).getLongitude() <= lgMax)) {
            i++;
        }
        if (i >= lstDonneesTR.size())
            i--;
        // On récupère la date
        Timestamp dateDernierePosAlcis = lstDonneesTR.get(i).getDatation();
        tempsAlcisEnMs = now.getTime() - dateDernierePosAlcis.getTime();
        tempsAlcisEnM = (int)(tempsAlcisEnMs / 60000);
        if (tempsAlcisEnM < 60) {
            // Si le véhicule y est depuis moins d'une heure on ne le considère pas
            tempsAlcisEnM = 0;
        }
    }
    else {
        tempsAlcisEnM = 0;
    }
    return tempsAlcisEnM;
}
```