SAI

1.0

Généré par Doxygen 1.7.6.1

Jeudi Juin 9 2016 21 :44 :28

Table des matières

1 Page principale du projet SAI

1.1 Introduction

Aider à la régulation du trafic en temps réel par l'exploitant.

Les fonctionnalités principales du SAI seront donc :
- la géolocalisation des bus ;
- la visualisation des données d'exploitation en temps réel ;
- la gestion des alertes ;
- l'envoi de messages vers la flotte de bus.

1.2 Table des matières

Dépôt SVN : https://svn.riouxsvn.com/saeiv

2 Changelog

---

r121 | dirar | 2016-06-09 16 :08 :19 +0200 (jeu. 09 juin 2016) | 1 ligne

Mise à jour de la base de donnée et modification de la carte

---

r116 | dirar | 2016-06-04 14 :40 :03 +0200 (sam. 04 juin 2016) | 1 ligne

Mofification selectionnerBus

---

r115 | dirar | 2016-06-04 04 :40 :26 +0200 (sam. 04 juin 2016) | 1 ligne

modification de la focntion selectionner bus pour afficher icone pour l'arret selectionné

---

r114 | dirar | 2016-06-03 17 :06 :05 +0200 (ven. 03 juin 2016) | 1 ligne

Ajout fonction pour interagir avec la selection du bus

---

r111 | dirar | 2016-06-03 12 :25 :17 +0200 (ven. 03 juin 2016) | 1 ligne

ajout slot selectionnerBus

---

r110 | dirar | 2016-06-02 22 :07 :40 +0200 (jeu. 02 juin 2016) | 1 ligne

Modification des fonction afficher

---

r109 | dirar | 2016-06-02 19 :34 :07 +0200 (jeu. 02 juin 2016) | 1 ligne

Modification de afficherSuivi() pour recuperer historique des heures et ajout de l'icone bus pour differencier les bus en retardet en avance

---

r108 | dirar | 2016-05-27 16 :13 :34 +0200 (ven. 27 mai 2016) | 1 ligne

Modification du IHM

---

r107 | dirar | 2016-05-27 14 :16 :45 +0200 (ven. 27 mai 2016) | 1 ligne

Modiffication de la methode afficherServicesVitesse()

---

r103 | dirar | 2016-05-26 15 :59 :39 +0200 (jeu. 26 mai 2016) | 1 ligne

Modification de la class serveur

---

r98 | tvaira | 2016-05-25 13 :23 :14 +0200 (mer. 25 mai 2016) | 1 ligne

Chemin icone dans Doxyfile

---

r94 | dirar | 2016-05-23 18 :33 :17 +0200 (lun. 23 mai 2016) | 1 ligne

Documentation classe IHM

---

r85 | tvaira | 2016-05-20 20 :11 :01 +0200 (ven. 20 mai 2016) | 1 ligne

Mise à jour documentation classe Serveur

---

r79 | tvaira | 2016-05-19 11 :28 :00 +0200 (jeu. 19 mai 2016) | 1 ligne

Ajout des icones pour la documentation

---

r78 | tvaira | 2016-05-19 11 :22 :20 +0200 (jeu. 19 mai 2016) | 1 ligne

Initialisation de la documentation SIV à compléter pour doxygen

---

r77 | tvaira | 2016-05-19 11 :12 :11 +0200 (jeu. 19 mai 2016) | 1 ligne

Initialisation de la documentation à compléter pour doxygen

---

r75 | dirar | 2016-05-18 15 :11 :29 +0200 (mer. 18 mai 2016) | 1 ligne

modification fonction decoderMessage

---

r74 | dirar | 2016-05-13 22 :25 :10 +0200 (ven. 13 mai 2016) | 1 ligne

modification de la fonction afficherSuivi et recupererSuivi

---

r73 | tvaira | 2016-05-13 06 :57 :26 +0200 (ven. 13 mai 2016) | 1 ligne

SAI : vérifie si un vehicule d'un service pour un itineraire existe

---

r69 | dirar | 2016-05-04 19 :19 :12 +0200 (mer. 04 mai 2016) | 1 ligne

Modification de la fonction afficherSuiviHeure() pour réafficher les anciennes heures sauvegardé + Calcul des avances/retards

---

r68 | dirar | 2016-05-04 17 :17 :37 +0200 (mer. 04 mai 2016) | 1 ligne

modification de la fonction calculduree et de l'affichage des suivis aves les heures

---

r53 | dirar | 2016-05-01 02 :20 :44 +0200 (dim. 01 mai 2016) | 1 ligne

ajout de la fonction calculerDuree() pour calculer l'avance ou le retard

---

r51 | dirar | 2016-04-30 20 :35 :54 +0200 (sam. 30 avril 2016) | 1 ligne

ajout de la fonction afficherSuiviHeure() qui affiche l'heure d'arrivée et l'heure de départ réel dans le suivi

---

r50 | dirar | 2016-04-30 18 :44 :10 +0200 (sam. 30 avril 2016) | 1 ligne

modification de la fonction afficherArretServices() pour afficher le prochain arret

---

r48 | dirar | 2016-04-29 16 :14 :01 +0200 (ven. 29 avril 2016) | 1 ligne

Modification de la fonction SelectionnerVehicule

---

r42 | dirar | 2016-04-28 21 :18 :43 +0200 (jeu. 28 avril 2016) | 1 ligne

ajout requete pour recuperer la direction dans la fonction selectionnerVehicule()

---

r38 | dirar | 2016-04-28 16 :11 :19 +0200 (jeu. 28 avril 2016) | 1 ligne

Ajout de la fonction actualiserLocalisation

---

r37 | dirar | 2016-04-27 14 :38 :20 +0200 (mer. 27 avril 2016) | 1 ligne

Ajout de la classe serveur sur le depot

---

r31 | dirar | 2016-04-27 08 :06 :19 +0200 (mer. 27 avril 2016) | 1 ligne

Ajout de la classe serveur sur le depot

---

r27 | dirar | 2016-04-22 17 :41 :12 +0200 (ven. 22 avril 2016) | 1 ligne

Ajout de la classe serveur et fonction pour receptionner les message

---

r20 | dirar | 2016-04-21 10 :22 :16 +0200 (jeu. 21 avril 2016) | 1 ligne

Modification recupererTraces()

---

r16 | dirar | 2016-04-20 16 :13 :56 +0200 (mer. 20 avril 2016) | 1 ligne

Ajout des itinéraires et des traces

---

r12 | dirar | 2016-03-23 17 :15 :12 +0100 (mer. 23 mars 2016) | 1 ligne

Ajout carte ,modification des requettes

---

r10 | dirar | 2016-03-23 11 :17 :04 +0100 (mer. 23 mars 2016) | 1 ligne

Ajout de qmapcontrol (fichiers oublies)

---

r9 | dirar | 2016-03-23 11 :15 :36 +0100 (mer. 23 mars 2016) | 1 ligne

Ajout de qmapcontrol (carte)

---

r8 | dirar | 2016-03-23 11 :05 :21 +0100 (mer. 23 mars 2016) | 1 ligne

Ajout architecture logicielle de base (sans la carte)

---

3 Configuration

`configuration.sh`

Informations sur le poste de développement

- Distribution : Ubuntu 12.04.5 LTS
- OS : GNU/Linux
- Noyau : Linux
- Version : 3.8.0-44-generic
- Machine : x86_64
- Processeur : Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
- Mémoire RAM : 8129984 kB

Liste des paquets Qt nécessaires

- libqt4-sql libqtgui4 libqtcore4

Liste des autres paquets nécessaires

- libc6 libstdc++6-armel-cross gcc-4.6-arm-linux-gnueabi libc6-armel-cross libfontconfig1 libaudio2 libglib2.0-0 libpng12-0 zlib1g libfreetype6 libglib2.0-0 libsm6 libice6 libxi6 libxrender1 libxext6 libx11-6 libc6-armel-cross libc6-armel-cross libc6-armel-cross libc6 libexpat1 libxt6 libxau6 libpcre3 libffi6 libuuid1 libxcb1 libxdmcp6

génèrè le jeudi 19 mai 2016, 07 :30 :44 (UTC+0200)

Informations de version sur les outils

`version.sh`

- g++ (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
- QMake version 2.01a
- GNU Make 3.81
- GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
- svn, version 1.6.17 (r1128011)
- geany 1.24.1 (construit le May 20 2014 avec GTK 2.24.10, GLib 2.32.4)
- sqlite3 3.7.9 2011-11-01 00 :52 :41 c7c6050ef060877ebe77b41d959e9df13f8c9b5e
- Qt Meta Object Compiler version 63 (Qt 4.8.1)
- Qt Creator 2.4.1 based on Qt 4.8.1
- cppunit-config 1.12.1
- doxygen 1.7.6.1
- bouml Bouml 4.23
- papyrus Papyrus 1.1.3

génèrè le jeudi 19 mai 2016, 07 :31 :08 (UTC+0200)

4 Manuel d'installation

Fabrication de l'exécutable :
- `qmake`
- `make`

5 Recette

Étudiant 4 : DIRA Rialy (SAI)

Le protocole de communication entre le serveur SAI et un client SIV est spécifié et mis en oeuvre

Une communication avec le SIV ou un simulateur fourni est possible

La prise et la fin de service d'un conducteur est détectée et prise en compte

Le suivi d'au moins une course d'un véhicule est possible dans le bandeau Service

La carte avec au moins la géolocalisation d'un véhicule est affichée et actualisée périodiquement

L'écart entre l'horaire théorique et réel (avance/retard) est calculé et affiché pour le suivi d'au moins un véhicule

Une journalisation des messages est réalisée

6 Base de données

PRAGMA foreign_keys=OFF ; BEGIN TRANSACTION ; CREATE TABLE ‘conducteur‘ (‘codeConducteur‘ smallint(4) NOT NULL, ‘nomConducteur‘ varchar(160) DEFAULT NULL, PRIMARY KEY (‘codeConducteur‘)) ; INSERT INTO "conducteur" VALUES(1,’denis darmon’) ; INSERT INTO "conducteur" VALUES(195,’richard boue’) ; INSERT INTO "conducteur" VALUES(1234,’fabrice allaire’) ; INSERT INTO "conducteur" VALUES(1962,’nicolas lepretre’) ; INSERT INTO "conducteur" VALUES(1993,’jean lapierre’) ; INSERT INTO "conducteur" VALUES(2121,’cyril salomon’) ; INSERT INTO "conducteur" VALUES(3082,’olivier piquet’) ; INSERT INTO "conducteur" VALUES(3553,’laurine thibault’) ; INSERT INTO "conducteur" VALUES(6340,’marc gross’) ; INSERT INTO "conducteur" VALUES(7275,’fanny delaunay’) ; CREATE TABLE ‘organisme‘ (‘idOrganisme‘ bigint(20) NOT NULL, ‘nomOrganisme‘ varchar(160) DEFAULT NULL, ‘telephone‘ varchar(10) DEFAULT NULL, ‘fuseauHoraire‘ varchar(160) DEFAULT NULL, ‘langue‘ varchar(160) DEFAULT NULL, ‘url‘ varchar(160) DEFAULT NULL, PRIMARY KEY (‘idOrganisme‘)) ; INSERT INTO "organisme" VALUES(6192449487677451,’Tisséo’,’0561417070’,’Europe/Paris’,’fr’,’http://www.tisseo.fr’) ; CREATE TABLE ‘vehicule‘ (‘idVehicule‘ smallint(3) NOT NULL, ‘typeVehicule‘ varchar(160) DEFAULT NULL, ‘capacite‘ smallint(3) DEFAULT NULL, PRIMARY KEY (‘idVehicule‘)) ; INSERT INTO "vehicule" VALUES(82,’articulé’,148) ; INSERT INTO "vehicule" VALUES(123,’standard’,100) ; INSERT INTO "vehicule" VALUES(155,’standard’,100) ; INSERT INTO "vehicule" VALUES(171,’articulé’,148) ; INSERT INTO "vehicule" VALUES(193,’standard’,100) ; INSERT INTO "vehicule" VALUES(204,’articulé’,148) ; INSERT INTO "vehicule" VALUES(287,’midibus’,88) ; INSERT INTO "vehicule" VALUES(345,’midibus’,88) ; INSERT INTO "vehicule" VALUES(615,’articulé’,148) ; INSERT INTO "vehicule" VALUES(679,’midibus’,88) ; INSERT INTO "vehicule" VALUES(766,’midibus’,88) ; INSERT INTO "vehicule" VALUES(986,’articulé’,148) ; CREATE TABLE ‘ligne‘ (‘idLigne‘ bigint(20) NOT NULL DEFAULT ’0’, ‘nomLong‘ varchar(160) DEFAULT NULL, ‘nomCourt‘ int(11) DEFAULT NULL, ‘description‘ varchar(160) DEFAULT NULL, ‘type‘ varchar(160) DEFAULT NULL, ‘couleurLigne‘ varchar(6) DEFAULT ’000000’, ‘couleurTexte‘ varchar(6) DEFAULT ’000000’, ‘idOrganisme‘ bigint(20) NOT NULL, PRIMARY KEY (‘idLigne‘), CONSTRAINT fk_ligne_idOrganisme FOREIGN KEY (idOrganisme) REFERENCES organisme(idOrganisme)) ; INSERT INTO "ligne" VALUES(11821949021891617,’St Cyprien - République / Oncopole’,3,’Ligne St Cyprien - République / Oncopole’,’3’,’660099’,’FFFFFF’,6192449487677451) ; INSERT INTO "ligne" VALUES(11821949021891635,’Colomiers Gare SNCF / Brax le Château’,32,’Ligne Colomiers Gare - SNCF / Brax le Château’,’3’,’8e4a05’,’FFFFFF’,6192449487677451) ; INSERT INTO "ligne" VALUES(11821949021891637,’Argoulets / Union Somport’,39,’Ligne Argoulets / Union Somport’,’3’,’00c62d’,’FFFFFF’,6192449487677451) ; CREATE TABLE ‘calendrier‘ (‘idCalendrier‘ bigint(20) NOT NULL, ‘lundi‘ binary(1) DEFAULT NULL, ‘mardi‘ binary(1) DEFAULT NULL, ‘mercredi‘ binary(1) DEFAULT NULL, ‘jeudi‘ binary(1) DEFAULT NULL, ‘vendredi‘ binary(1) DEFAULT NULL, ‘samedi‘ binary(1) DEFAULT NULL, ‘dimanche‘ binary(1) DEFAULT NULL, ‘dateDebut‘ date DEFAULT NULL, ‘dateFin‘ date DEFAULT NULL, PRIMARY KEY (‘idCalendrier‘)) ; INSERT INTO "calendrier" VALUES(4503603928071438,1,1,1,1,1,1,0,’2015-01-01’,’2016-12-31’) ; INSERT INTO "calendrier" VALUES(4503599631512310,1,1,1,1,1,0,0,’2015-01-01’,’2016-12-31’) ; INSERT INTO "calendrier" VALUES(4503603925963979,0,0,0,0,0,1,0,’2015-01-01’,’2016-12-31’) ; INSERT INTO "calendrier" VALUES(4503603928071505,1,1,1,1,1,0,0,’2015-01-01’,’2016-12-31’) ; CREATE TABLE ‘arret‘ (‘idArret‘ bigint(20) NOT NULL, ‘heureArrivee‘ time DEFAULT NULL, ‘heureDepart‘ time DEFAULT NULL, ‘numeroSequence‘ int(11) DEFAULT NULL, ‘informationVoyageur‘ int(11) DEFAULT NULL, ‘idItineraire‘ bigint(20) NOT NULL DEFAULT ’0’, PRIMARY KEY (‘idArret‘,‘idItineraire‘), CONSTRAINT fk_arret_idItineraire FOREIGN KEY (idItineraire) REFERENCES itineraire(idItineraire)) ; INSERT INTO "arret" VALUES(3377699720880909,’20 :25 :00’,’20 :25 :00’,0,0,4503603928074023) ;

CREATE TABLE ‘lieu‘ (‘idArret‘ bigint(20) NOT NULL, ‘codeArret‘ int(11) NOT NULL, ‘nomLieu‘ varchar(160) DEFAULT NULL, ‘latitude‘ varchar(160) DEFAULT NULL, ‘longitude‘ varchar(160) DEFAULT NULL, ‘typeLieu‘ binary(1) DEFAULT NULL, ‘stationParente‘ bigint(20) NOT NULL, PRIMARY KEY (‘idArret‘)) ; INSERT INTO "lieu" VALUES(3377699720880908,6760,’St Cyprien - République’,’43.5986’,’1.43027’,0,1970324837184772) ; INSERT INTO "lieu" VALUES(3377699720880909,6763,’St Cyprien - République’,’43.5975’,’1.43093’,0,1970324837184772) ; INSERT INTO "lieu" VALUES(3377699720880912,6765,’St Cyprien - République’,’43.5979’,’1.43175’,0,1970324837184772) ; INSERT INTO "lieu" VALUES(3377699720880914,6761,’St Cyprien - République’,’43.5971’,’1.43122’,0,1970324837184772) ; INSERT INTO "lieu" VALUES(3377699720880940,6840,’Palais de Justice’,’43.5932’,’1.44495’,0,1970324837184792) ; INSERT INTO "lieu" VALUES(3377699720880988,423,’Bagatelle’,’43.5802’,’1.41168’,0,1970324837184809) ; INSERT INTO "lieu" VALUES(3377699720880989,420,’Bagatelle’,’43.5797’,’1.41181’,0,1970324837184809) ; INSERT INTO "lieu" VALUES(3377699720881173,2431,’Esquirol’,’43.6005’,’1.44393’,0,1970324837184892) ; INSERT INTO "lieu" VALUES(3377699720881174,2430,’Esquirol’,’43.6004’,’1.4447’,0,1970324837184892) ; INSERT INTO "lieu" VALUES(3377699720881193,5591,’Pont Neuf’,’43.5999’,’1.4415’,0,1970324837184897) ; INSERT INTO "lieu" VALUES(3377699720881194,5590,’Pont Neuf’,’43.5997’,’1.44091’,0,1970324837184897) ; INSERT INTO "lieu" VALUES(3377699720881231,221,’Arsenal’,’43.6056’,’1.43366’,0,1970324837184915) ; INSERT INTO "lieu" VALUES(3377699720881298,3183,’Grand Rond’,’43.5964’,’1.45401’,0,1970324837184947) ; INSERT INTO "lieu" VALUES(3377699720881299,3180,’Grand Rond’,’43.5939’,’1.45238’,0,1970324837184947) ; INSERT INTO "lieu" VALUES(3377699720881301,3181,’Grand Rond’,’43.595’,’1.45364’,0,1970324837184947) ; INSERT INTO "lieu" VALUES(3377699720881404,1728
Cours Dillon’,’43.5988’,’1.43744’,0,1970324837184995) ; INSERT INTO "lieu" VALUES(3377699720881409,1721,’Cours Dillon’,’43.5988’,’1.43722’,0,1970324837184995) ; INSERT INTO "lieu" VALUES(3377699720881427,2831,’Fontaine Lestang’,’43.5878’,’1.41857’,0,1970324837185007) ; INSERT INTO "lieu" VALUES(3377699720881428,2830,’Fontaine Lestang’,’43.5878’,’1.41871’,0,1970324837185007) ; INSERT INTO "lieu" VALUES(3377699720881489,3410,’Héraclès’,’43.6086’,’1.42952’,0,1970324837185029) ; INSERT INTO "lieu" VALUES(3377699720881490,3411,’Héraclès’,’43.6086’,’1.42901’,0,197032483718502

INSERT INTO "lieu" VALUES(3377699720881911,7021,'Trois Fours','43.5875','1.46147',0,1970324837185312) ; INSERT INTO "lieu" VALUES(3377699720881912,7020,'Trois Fours','43.5872','1.46192',0,1970324837185312) ; INSERT INTO "lieu" VALUES(3377699720881918,5581,'Montaudran','43.5716','1.4977',0,1970324837185315) ; INSERT INTO "lieu" VALUES(3377699720881919,5580,'Montaudran','43.5708','1.49923',0,1970324837185315) ; INSERT INTO "lieu" VALUES(3377699720882004,4121,'Leclerc','43.6094','1.43143',0,1970324837185358) ; INSERT INTO "lieu" VALUES(3377699720882275,6743,'François Verdier','43.6005','1.45144',0,1970324837185541) ; INSERT INTO "lieu" VALUES(3377699720882277,6742,'François Verdier','43.6006','1.45119',0,1970324837185541) ; INSERT INTO "lieu" VALUES(3377699720882283,5870,'Quartier Général','43.598','1.45215',0,1970324837185542) ; INSERT INTO "lieu" VALUES(3377699720882287,2762,'Fer à Cheval','43.5939','1.43375',0,1970324837185543) ; INSERT INTO "lieu" VALUES(3377699720882288,2763,'Fer à Cheval','43.5938','1.43371',0,1970324837185543) ; INSERT INTO "lieu" VALUES(3377699720882315,1750,'Crampel','43.5881','1.45927',0,1970324837185566) ; INSERT INTO "lieu" VALUES(3377699720882316,1751,'Crampel','43.5883','1.45895',0,1970324837185566) ; INSERT INTO "lieu" VALUES(3377699720882317,1711,'Courrège','43.5842','1.46923',0,1970324837185567
INSERT INTO "lieu" VALUES(3377699720882318,1710,'Courrège','43.5841','1.46952',0,1970324837185567) ; INSERT INTO "lieu" VALUES(3377699720882750,6980,'Teinturiers','43.5954','1.43244',0,1970324837185912) ; INSERT INTO "lieu" VALUES(3377699720882751,6981,'Teinturiers','43.5957','1.43244',0,1970324837185912) ; INSERT INTO "lieu" VALUES(3377699720882752,2140,'Delpy','43.5934','1.43087',0,1970324837185913) ; INSERT INTO "lieu" VALUES(3377699720882753,1851,'Cugnaux','43.5956','1.43143',0,1970324837185914) ; INSERT INTO "lieu" VALUES(3377699720882754,6401,'Ste-Lucie','43.5939','1.42958',0,1970324837185915) ; INSERT INTO "lieu" VALUES(3377699720882755,6111,'Rodin','43.5925','1.42744',0,197032483718
INSERT INTO "lieu" VALUES(3377699720882756,6110,'Rodin','43.5924','1.42739',0,1970324837185916) ; INSERT INTO "lieu" VALUES(3377699720882757,7110,'Valats','43.5911','1.4249',0,1970324837185917) ; INSERT INTO "lieu" VALUES(3377699720882758,7111,'Valats','43.5912','1.42514',0,1970324837185917) ; INSERT INTO "lieu" VALUES(3377699720882761,1971,'Déodat de Séverac','43.5898','1.42253',0,1970324837185918) ; INSERT INTO "lieu" VALUES(3377699720882762,1970,'Déodat de Séverac','43.59','1.42277',0,1970324837185918) ; INSERT INTO "lieu" VALUES(3377699720882763,3020,'Gamelin','43.5888','1.42123',0,1970324837185919) ; INSERT INTO "lieu" VALUES(3377699720882764,3021,'Gamelin','43.5889','1.42142',0,1970324837185919) ; INSERT INTO "lieu" VALUES(3377699720882766,4700,'Mont Dore','43.5846','1.41579',0,1970324837185920) ; INSERT INTO "lieu" VALUES(3377699720882767,6920,'Tellier','43.5857','1.41371',0,1970324837185921) ;
INSERT INTO "lieu" VALUES(3377699720882768,6921,'Tellier','43.5854','1.41241',0,1970324837185921) ; INSERT INTO "lieu" VALUES(3377699720882775,4270,'Place du Morvan','43.5807','1.40946',0,1970324837185925) ; INSERT INTO "lieu" VALUES(3377699720882776,4271,'Place du Morvan','43.5809','1.40894',0,1970324837185925) ; INSERT INTO "lieu" VALUES(3377699720882777,3281,'Guyenne','43.5772','1.40984',0,1970324837185926) ; INSERT INTO "lieu" VALUES(3377699720882778,3280,'Guyenne','43.577','1.41047',0,1970324837185926) ; INSERT INTO "lieu" VALUES(3377699720882779,3510,'Ile de France','43.576','1.41165',0,1970324837185927) ; INSERT INTO "lieu" VALUES(3377699720882780,3511,'Ile de France','43.5764','1.41223',0,1970324837185927) ; INSERT INTO "lieu" VALUES(3377699720882781,671,'Bigorre','43.5748','1.41107',0,1970324837185928) ; INSERT INTO "lieu" VALUES(3377699720882782,1661,'Ch. Papus','43.5742','1.41195',0,1970324837185929) ; INSERT INTO "lieu" VALUES(3377699720882785,5620,'Ile du Ramier','43.5923','1.44044',0,197032484053
INSERT INTO "lieu" VALUES(3377699720882786,5621,'Ile du Ramier','43.5922','1.44035',0,1970324840530407) ; INSERT INTO "lieu" VALUES(3377699720882787,3651,'Jardin Royal','43.5948','1.44961',0,1970324837185933) ; INSERT INTO "lieu" VALUES(3377699720882788,3650,'Jardin Royal','43.5948','1.4495',0,1970324837185933) ; INSERT INTO "lieu" VALUES(3377699720882798,91,'Amidonniers','43.6061','1.42818',0,1970324837185937) ; INSERT INTO "lieu" VALUES(3377699720882799,90,'Amidonniers','43.6066','1.42821',0,1970324837185937) ; INSERT INTO "lieu" VALUES(3377699720882800,11,'Les Abattoirs','43.6009','1.42851',0,1970324837185938) ; INSERT INTO "lieu" VALUES(3377699720882871,1961 Demouilles','43.5893','1.45665',0,1970324837185971) ; INSERT INTO "lieu" VALUES(3377699720882872,1960,'Demouilles','43.5888','1.45763',0,1970324837185971) ; INSERT INTO "lieu" VALUES(3377699720882873,2911,'Frizac','43.5908','1.45354',0,197032483718
INSERT INTO "lieu" VALUES(3377699720882874,2910,'Frizac','43.5905','1.4538',0,1970324837185972) ; INSERT INTO "lieu" VALUES(3377699720882875,3641,'Jardin des Plantes','43.5928','1.4532',0,1970324837185973) ; INSERT INTO "lieu" VALUES(3377699720882876,3640,'Jardin des Plantes','43.5925','1.45265',0,1970324837185973) ; INSERT INTO "lieu" VALUES(3377699720882906,720,'Bordelongue','43.5709','1.41678',0,1970324837185987) ; INSERT INTO "lieu" VALUES(3377699720882907,721,'Bordelongue','43.5711','1.41724',0,1970324837185987) ; INSERT INTO "lieu" VALUES(3377699720883024,451,'Barcelone Leclerc','43.607','1.43069',0,1970324837186054) ; INSERT INTO "lieu" VALUES(3377699720883103,6913,'Tahiti','43.5799','1.48064',0,1970324837186092) ; INSERT INTO "lieu" VALUES(3377699720883112,6611,'Sciences Sociales','43.6063','1.43473',0,1970324837186096) ; INSERT INTO "lieu" VALUES(3377699720883336,3130,'Gironis','43.5638','1.42527',0,1970324837186218) ; INSERT INTO "lieu" VALUES(3377699720883337,3131,'Gironis','43.5637','1.42535',0,1970324837186218) ; INSERT INTO "lieu" VALUES(3377699720883338,4451,'Marchant','43.5594','1.42474',0,1970324837186219) ; INSERT INTO "lieu" VALUES(3377699720883339,4450,'Marchant','43.5594','1.42474',0,197032483718621
INSERT INTO "lieu" VALUES(3377699720884466,6421,'Dufour','43.5855','1.4659',0,1970324837186812) ; INSERT INTO "lieu" VALUES(3377699720884467,6420,'Dufour','43.585','1.4672',0,1970324837186812) ; INSERT INTO "lieu" VALUES(3377699720884468,1700,'Cottages','43.5833','1.47143',0,1970324837186813) ; INSERT INTO "lieu" VALUES(3377699720884469,1701,'Cottages','43.5835','1.47093',0,1970324837186813) ; INSERT INTO "lieu" VALUES(3377699720884470,801,'Buissonnets','43.5827','1.47301',0,1970324837186814) ; INSERT INTO "lieu" VALUES(3377699720884471,3680,'Jean-Paul Laurens','43.5822','1.47417',0,1970324837186815) ; INSERT INTO "lieu" VALUES(3377699720884472,3681,'Jean-Paul Laurens','43.5819','1.47489',0,1970324837186815) ; INSERT INTO "lieu" VALUES(3377699720884473,6700,'Six Avril','43.5809','1.-

47746',0,1970324837186816) ; INSERT INTO "lieu" VALUES(3377699720884474,6701,'Six Avril','43.5807','1.47823',0,1970324837186816)
INSERT INTO "lieu" VALUES(3377699720884475,7341,'Clinique Languedoc','43.5777','1.48561',0,1970324837186817) ; -
INSERT INTO "lieu" VALUES(3377699720884476,7340,'Clinique Languedoc','43.5775','1.4857',0,1970324837186817) ; I-
NSERT INTO "lieu" VALUES(3377699720884477,5151,'Ormeau','43.5791','1.48331',0,1970324837186818) ; INSERT IN-
TO "lieu" VALUES(3377699720884478,2261,'Ecole Montaudran','43.5766','1.48724',0,1970324837186819) ; INSERT INT-
O "lieu" VALUES(3377699720884479,850,'Bajac','43.5758','1.48847',0,1970324837186820) ; INSERT INTO "lieu" VALUE-
S(3377699720884480,851,'Bajac','43.5757','1.48869',0,1970324837186820) ; INSERT INTO "lieu" VALUES(3377699720884481,7331,'-
Lafaurie','43.5745','1.49048',0,1970324837186821) ; INSERT INTO "lieu" VALUES(3377699720884482,7330,'Lafaurie','43.-
5738','1.4915',0,1970324837186821) ; INSERT INTO "lieu" VALUES(3377699720884483,6241,'Route de Revel','43.5689','1.-
50366',0,1970324837186822) ; INSERT INTO "lieu" VALUES(3377699720884484,6240,'Route de Revel','43.5693','1.-
50271',0,1970324837186822) ; INSERT INTO "lieu" VALUES(3377699720888538,10,'Les Abattoirs','43.6012','1.42847',0,197032483718593
INSERT INTO "lieu" VALUES(3377699720888764,6914,'Tahiti','43.5795','1.48176',0,1970324837186092) ; INSERT INT-
O "lieu" VALUES(3377699721872252,4350,'Boulbonne','43.6006','1.44749',0,1970324838318971) ; INSERT INTO "lieu" -
VALUES(3377699721872253,4351,'Boulbonne','43.6006','1.44799',0,1970324838318971) ; INSERT INTO "lieu" VALU-
ES(3377699721902064,2561,'Compans-Caffarelli','43.6099','1.43488',0,1970324837184671) ; INSERT INTO "lieu" VA-
LUES(3377699721902071,6843,'Palais de Justice','43.5928','1.44408',0,1970324837184792) ; INSERT INTO "lieu" V-
ALUES(3377699722238902,23101,'Ch. des Martyrs','43.5705','1.41991',0,1970324839637228) ; INSERT INTO "lieu" -
VALUES(3377699723132164,1660,'Ch. Papus','43.574','1.41266',0,1970324837185929) ; INSERT INTO "lieu" VALUE-
S(3377699723132175,23040,'Hubert Curien','43.5578','1.42701',0,1970324839578894) ; INSERT INTO "lieu" VALUE-
S(3377699723132177,23041,'Hubert Curien','43.5578','1.42701',0,1970324839578894) ; INSERT INTO "lieu" VALUE-
S(3377699723135387,23051,'Oncopole','43.5542','1.43185',0,1970324839361574) ; INSERT INTO "lieu" VALUES(3377699723135401,670,
Bigorre','43.5748','1.41099',0,1970324837185928) ; INSERT INTO "lieu" VALUES(3377699723135406,23020,'Landes','43.-
5729','1.41694',0,1970324839582125) ; INSERT INTO "lieu" VALUES(3377699723135408,23021,'Landes','43.5731','1.-
41612',0,1970324839582125) ; INSERT INTO "lieu" VALUES(3377699723154117,23030,'Courtois de Viçose','43.5698','1.-
42287',0,1970324839600836) ; INSERT INTO "lieu" VALUES(3377699723154118,23031,'Courtois de Viçose','43.5698','1.-
42274',0,1970324839600836) ; INSERT INTO "lieu" VALUES(3377699723190509,23100,'Ch. des Martyrs','43.5705','1.-
41978',0,1970324839637228) ; INSERT INTO "lieu" VALUES(3377699723937619,20270,'Maison Commune','43.5575','1.-
43009',0,1970324840384338) ; INSERT INTO "lieu" VALUES(3377699723937620,20271,'Maison Commune','43.5575','1.-
43043',0,1970324840384338) ; INSERT INTO "lieu" VALUES(3377699724083641,6582,'Rond-Point Langlade','43.5668','1.-
42501',0,1970324837185088) ; INSERT INTO "lieu" VALUES(3377699724083642,6583,'Rond-Point Langlade','43.5668','1.-
425',0,1970324837185088) ; INSERT INTO "lieu" VALUES(3377699724118456,4120,'Leclerc','43.6094','1.43161',0,1970324837185358) ;
INSERT INTO "lieu" VALUES(3377704015495172,1130,'Cher','43.5816','1.40795',0,1970329131941891) ; INSERT INT-
O "lieu" VALUES(3377704015495173,1131,'Cher','43.582','1.40782',0,1970329131941891) ; INSERT INTO "lieu" VALUE-
S(3377704015495204,6150,'Ronsard','43.5833','1.40695',0,1970329131941910) ; INSERT INTO "lieu" VALUES(3377704015495205,6151,'-
Ronsard','43.5834','1.40692',0,1970329131941910) ; INSERT INTO "lieu" VALUES(3377704015495208,6432,'St-Gaudens','43.-
5851','1.41021',0,1970329131941911) ; INSERT INTO "lieu" VALUES(3377704015495209,6433,'St-Gaudens','43.5851','1.-
41035',0,1970329131941911) ; INSERT INTO "lieu" VALUES(3377704015495511,27552,'Malepère','43.5671','1.50643',0,197032483718685
INSERT INTO "lieu" VALUES(3377704015495519,27551,'Malepère','43.567','1.50573',0,1970324837186852) ; INSER-
T INTO "lieu" VALUES(3377699722724038,19961,'Brax le Château','43.6169','1.2403',0,1970324837185068) ; INSE-
RT INTO "lieu" VALUES(3377699722724037,19951,'La Chauge','43.6155','1.25',0,1970324837190345) ; INSERT IN-
TO "lieu" VALUES(3377699722724036,19941,'Château Cru','43.6127','1.26828',0,1970324837188856) ; INSERT INT-
O "lieu" VALUES(3377699722724035,19931,'Stade','43.6153','1.27652',0,1970324837185071) ; INSERT INTO "lieu" -
VALUES(3377699722724034,19921,'Basilique','43.6177','1.28334',0,1970324837185072) ; INSERT INTO "lieu" VALUE-
S(3377699722724033,19911,'Tuilerie','43.6162','1.29006',0,1970324837188855) ; INSERT INTO "lieu" VALUES(3377699721034355,19901,
Lycée International','43.6122','1.31031',0,1970324837185074) ; INSERT INTO "lieu" VALUES(3377699722724032,19891,'-
Piquemil','43.6123','1.31955',0,1970324837189860) ; INSERT INTO "lieu" VALUES(3377699720881634,16240,'Passerelle','43.-
6117','1.33032',0,1970324837186554) ; INSERT INTO "lieu" VALUES(3377699721158291,19735,'Colomiers Gare S-
NCF','43.6042','1.3347',0,1970324837184752) ; INSERT INTO "lieu" VALUES(3377699720888416,343,'Argoulets','43.-
6245','1.47742',0,1970324837190453) ; INSERT INTO "lieu" VALUES(3377699720881250,1800,'Cambard','43.6294','1.-
47642',0,1970324837184924) ; INSERT INTO "lieu" VALUES(3377699720881248,7250,'Vasseur','43.6363','1.47279',0,1970324837184923)
INSERT INTO "lieu" VALUES(3377699720881247,5160,'Ohnet','43.6389','1.471',0,1970324837184922) ; INSERT INTO
"lieu" VALUES(3377699720881245,1900,'Caunes','43.6403','1.46965',0,1970324837184921) ; INSERT INTO "lieu" VALUE-
S(3377699720881234,6570,'Atlanta','43.6424','1.46744',0,1970324837184917) ; INSERT INTO "lieu" VALUES(3377699720888422,19070,'-
St-Caprais','43.6475','1.47079',0,1970324837190454) ; INSERT INTO "lieu" VALUES(3377699720881242,10110,'Bayonne','43.-
6503','1.47231',0,1970324837184920) ; INSERT INTO "lieu" VALUES(3377699720880759,10160,'Ctre Cial Union','43.-
6565','1.47985',0,1970324837184690) ; INSERT INTO "lieu" VALUES(3377699720881237,10180,'Union Somport','43.-
655','1.48246',0,1970324837184918) ; INSERT INTO "lieu" VALUES(3377699720881238,10190,'Montcalm','43.6535','1.-
48441',0,1970324837184919) ; INSERT INTO "lieu" VALUES(3377699720881239,10192,'Montcalm','43.6533','1.48471',0,19703248371849
INSERT INTO "lieu" VALUES(3377699720881236,10181,'Union Somport','43.6553','1.4819',0,1970324837184918) ; INS-
ERT INTO "lieu" VALUES(3377699720880758,10163,'Ctre Cial Union','43.6566','1.47932',0,1970324837184690) ; INSE-
RT INTO "lieu" VALUES(3377699720881241,10111,'Bayonne','43.6492','1.47125',0,1970324837184920) ; INSERT INTO

"lieu" VALUES(3377699720881235,6571,'Atlanta','43.6433','1.46783',0,1970324837184917) ; INSERT INTO "lieu" VALUE-
S(3377699720881244,1901,'Caunes','43.6405','1.46937',0,1970324837184921) ; INSERT INTO "lieu" VALUES(3377699720881246,5161,'-
Ohnet','43.6384','1.47152',0,1970324837184922) ; INSERT INTO "lieu" VALUES(3377699720881249,7251,'Vasseur','43.-
6351','1.47323',0,1970324837184923) ; INSERT INTO "lieu" VALUES(3377699720881251,1801,'Cambard','43.6293','1.-
47632',0,1970324837184924) ; INSERT INTO "lieu" VALUES(3377699720887720,19890,'Piquemil','43.6125','1.31949',0,197032483718986
INSERT INTO "lieu" VALUES(3377699720881548,19900,'Lycée International','43.6124','1.3103',0,1970324837185074) ; I-
NSERT INTO "lieu" VALUES(3377699720886670,19910,'Tuilerie','43.6165','1.29011',0,1970324837188855) ; INSERT INTO
"lieu" VALUES(3377699720881546,19920,'Basilique','43.6179','1.28327',0,1970324837185072) ; INSERT INTO "lieu" VALUE-
S(3377699720881545,19930,'Stade','43.6154','1.27604',0,1970324837185071) ; INSERT INTO "lieu" VALUES(3377699720886671,19940,'-
Château Cru','43.6129','1.26819',0,1970324837188856) ; INSERT INTO "lieu" VALUES(3377699720888294,19950,'La -
Chauge','43.6157','1.24983',0,1970324837190345) ; CREATE TABLE itineraire ("idItineraire" BIGINT(20) NOT NULL, "desti-
nation" VARCHAR(160) DEFAULT ('NULL'), "direction" BINARY(1) DEFAULT ('NULL'), "idCalendrier" BIGINT(20) DEFAULT
('NULL'), "idLigne" BIGINT(20) DEFAULT ('NULL')) ; INSERT INTO "itineraire" VALUES(4503603928074023,'Oncopole T-
OULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074024,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074025,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074026,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074027,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074028,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074029,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074030,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074031,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074032,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074033,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074034,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074035,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074036,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074037,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074038,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074039,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074040,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074041,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074042,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074043,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074044,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074045,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074046,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074047,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074048,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074049,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074050,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074051,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074052,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074053,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074054,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074055,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074056,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074057,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074058,'Oncopole
TOULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074059,'Oncopole T-
OULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074060,'Oncopole TO-
ULOUSE',1,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074122,'Saint Cyprien -
République TOULOUSE',0,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUES(4503603928074123,'-
Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891617) ; INSERT INTO "itineraire" VALUE-
S(4503603928074124,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891617) ; INSERT INTO "iti-
neraire" VALUES(4503603928074125,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891617) ; I-
NSERT INTO "itineraire" VALUES(4503603928074126,'Saint Cyprien - République TOULOUSE',0,4503603928071438,1182194902189161
INSERT INTO "itineraire" VALUES(4503603928074127,'Saint Cyprien - République TOULOUSE',0,4503603928071438,118219490218916
INSERT INTO "itineraire" VALUES(4503603928074128,'Saint Cyprien - République TOULOUSE',0,4503603928071438,118219490218916
INSERT INTO "itineraire" VALUES(4503603928074129,'Saint Cyprien - République TOULOUSE',0,4503603928071438,118219490218916
INSERT INTO "itineraire" VALUES(4503603928074130,'Saint Cyprien - République TOULOUSE',0,4503603928071438,118219490218916
INSERT INTO "itineraire" VALUES(4503603928074131,'Saint Cyprien - République TOULOUSE',0,4503603928071438,118219490218916

INSERT INTO "itineraire" VALUES(4503603928074132,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074133,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074134,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074135,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074136,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074137,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074138,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074139,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074140,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074141,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074142,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074143,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074144,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074145,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074146,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074147,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074148,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074149,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074150,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074151,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074152,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074153,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074154,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074155,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074156,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074157,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074158,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074159,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928074160,'Saint Cyprien - République TOULOUSE',0,4503603928071438,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673401,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673402,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673403,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673404,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673405,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673406,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673407,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673408,'Colomiers Gare SNCF COLOMIERS',0,4503599631512310,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603922673413,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; I-
NSERT INTO "itineraire" VALUES(4503603922673414,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; IN-
SERT INTO "itineraire" VALUES(4503603922673415,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; INS-
ERT INTO "itineraire" VALUES(4503603922673416,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; INSE-
RT INTO "itineraire" VALUES(4503603922673417,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; INSER-
T INTO "itineraire" VALUES(4503603922673418,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; INSERT
INTO "itineraire" VALUES(4503603922673419,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; INSERT I-
NTO "itineraire" VALUES(4503603922673420,'Brax le Château BRAX',1,4503599631512310,11821949021891635) ; INSERT IN-
TO "itineraire" VALUES(4503603925963979,'Colomiers Gare SNCF COLOMIERS',0,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963980,'Colomiers Gare SNCF COLOMIERS',0,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963981,'Colomiers Gare SNCF COLOMIERS',0,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963982,'Colomiers Gare SNCF COLOMIERS',0,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963983,'Brax le Château BRAX',1,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963984,'Brax le Château BRAX',1,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963985,'Brax le Château BRAX',1,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603925963986,'Brax le Château BRAX',1,4503603925963979,11821949021891635) ;
INSERT INTO "itineraire" VALUES(4503603928072201,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072202,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072203,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072204,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072205,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072206,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072207,'Union Somport UNION',1,4503603928071505,11821949021891637) ;
INSERT INTO "itineraire" VALUES(4503603928072208,'Union Somport UNION',1,4503603928071505,11821949021891637) ;

INSERT INTO "itineraire" VALUES(4503603928072209,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072210,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072211,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072212,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072213,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072214,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072215,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072216,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072217,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072218,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072219,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072220,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072221,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072222,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072223,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072224,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072225,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072226,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072227,'Union Somport UNION',1,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072228,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072229,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072230,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072231,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072232,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072233,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072234,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072235,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072236,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072237,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072238,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072239,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072240,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072241,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072242,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072243,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072244,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072245,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072246,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072247,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072248,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072249,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072250,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072251,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072252,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072253,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; INSERT INTO "itineraire" VALUES(4503603928072254,'Argoulets TOULOUSE',0,4503603928071505,11821949021891637) ; CREATE TABLE ‘service‘ (‘idService‘ bigint(20) NOT NULL, ‘codeConducteur‘ smallint(4) DEFAULT NULL, ‘idVehicule‘ smallint(3) DEFAULT NULL, ‘dateDebut‘ date DEFAULT NULL, ‘heureDebut‘ time DEFAULT NULL, ‘dateFin‘ date DEFAULT NULL, ‘heureFin‘ time DEFAULT NULL, ‘effectue‘ int(1) NOT NULL DEFAULT 0, PRIMARY KEY (‘idService‘), CONSTRAINT fk_service_codeConducteur FOREIGN KEY (codeConducteur) REFERENCES conducteur(codeConducteur), CONSTRAINT fk_service_idVehicule FOREIGN KEY (idVehicule) REFERENCES vehicule(idVehicule)) ; INSERT INTO "service" VALUES(1,1234,155,'2015-07-11','07 :00 :00',",",0) ; INSERT INTO "service" VALUES(2,195,155,'2015-07-11','10 :00 :00',",",0) ; INSERT INTO "service" VALUES(3,1993,123,'2015-07-11','07 :00 :00',",",0) ; INSERT INTO "service" VALUES(4,2121,123,'2015-07-11','09 :00 :00',",",0) ; INSERT INTO "service" VALUES(5,7275,193,'2015-07-11','07 :00 :00',",",0) ; INSERT INTO "service" VALUES(6,6340,193,'2015-07-11','09 :00 :00',",",0) ; CREATE TABLE ‘course‘ (‘idItineraire‘ bigint(20) NOT NULL, ‘idService‘ smallint(3) DEFAULT NULL, ‘effectue‘ int(1) NOT NULL DEFAULT 0, PRIMARY KEY (‘idItineraire‘, ‘idService‘), CONSTRAINT fk_course_idItineraire FOREIGN KEY (idItineraire) REFERENCES itineraire(idItineraire), CONSTRAINT fk_course_idService FOREIGN KEY (idService) REFERENCES service(idService)) ; INSERT INTO "course" VALUES(4503603928074122,1,0) ; INSERT INTO "course" VALUES(4503603928074024,1,0) ; INSERT INTO "course" VALUES(4503603928074132,1,0) ; INSERT INTO "course" VALUES(4503603928074032,1,0) ; INSERT INTO "course" VALUES(4503603928074135,1,0) ; INSERT INTO "course" VALUES(4503603928074035,1,0) ; INSERT INTO "course" VALUES(4503603928074146,1,0) ; INS-

ERT INTO "course" VALUES(4503603928074052,1,0) ; INSERT INTO "course" VALUES(4503603928074149,1,0) ; INSERT INTO "course" VALUES(4503603928074055,1,0) ; INSERT INTO "course" VALUES(4503603928074131,1,0) ; INSERT INTO "course" VALUES(4503603928074138,2,0) ; INSERT INTO "course" VALUES(4503603928074038,2,0) ; INSERT INTO "course" VALUES(4503603928074141,2,0) ; INSERT INTO "course" VALUES(4503603928074041,2,0) ; INSERT INTO "course" VALUES(4503603928074144,2,0) ; INSERT INTO "course" VALUES(4503603928074044,2,0) ; INSERT INTO "course" VALUES(4503603928074153,2,0) ; INSERT INTO "course" VALUES(4503603928074050,2,0) ; INSERT INTO "course" VALUES(4503603928074157,2,0) ; INSERT INTO "course" VALUES(4503603928074057,2,0) ; INSERT INTO "course" VALUES(4503603928074129,1,0) ; INSERT INTO "course" VALUES(4503603922673401,3,0) ; INSERT INTO "course" VALUES(4503603922673413,3,0) ; INSERT INTO "course" VALUES(4503603922673402,3,0) ; INSERT INTO "course" VALUES(4503603922673414,3,0) ; INSERT INTO "course" VALUES(4503603922673403,3,0) ; INSERT INTO "course" VALUES(4503603922673415,3,0) ; INSERT INTO "course" VALUES(4503603922673404,3,0) ; INSERT INTO "course" VALUES(4503603922673416,3,0) ; INSERT INTO "course" VALUES(4503603922673405,3,0) ; INSERT INTO "course" VALUES(4503603922673417,3,0) ; INSERT INTO "course" VALUES(4503603922673406,4,0) ; INSERT INTO "course" VALUES(4503603922673418,4,0) ; INSERT INTO "course" VALUES(4503603922673407,4,0) ; INSERT INTO "course" VALUES(4503603922673419,4,0) ; INSERT INTO "course" VALUES(4503603922673408,4,0) ; INSERT INTO "course" VALUES(4503603922673420,4,0) ; INSERT INTO "course" VALUES(4503603928072228,5,0) ; INSERT INTO "course" VALUES(4503603928072201,5,0) ; INSERT INTO "course" VALUES(4503603928072229,5,0) ; INSERT INTO "course" VALUES(4503603928072202,5,0) ; INSERT INTO "course" VALUES(4503603928072230,5,0) ; INSERT INTO "course" VALUES(4503603928072203,5,0) ; INSERT INTO "course" VALUES(4503603928072231,5,0) ; INSERT INTO "course" VALUES(4503603928072204,5,0) ; INSERT INTO "course" VALUES(4503603928072232,5,0) ; INSERT INTO "course" VALUES(4503603928072205,5,0) ; INSERT INTO "course" VALUES(4503603928072233,5,0) ; INSERT INTO "course" VALUES(4503603928072206,5,0) ; INSERT INTO "course" VALUES(4503603928072252,6,0) ; INSERT INTO "course" VALUES(4503603928072225,6,0) ; INSERT INTO "course" VALUES(4503603928072253,6,0) ; INSERT INTO "course" VALUES(4503603928072226,6,0) ; INSERT INTO "course" VALUES(4503603928072254,6,0) ; INSERT INTO "course" VALUES(4503603928072227,6,0) ; CREATE TABLE ‘trace‘ (‘idTrace‘ bigint(20) NOT NULL, ‘pt_lat‘ real DEFAULT NULL, ‘pt_lon‘ real DEFAULT NULL, ‘pt_sequence‘ int(11) NOT NULL DEFAULT 0, PRIMARY KEY (‘idTrace‘, ‘pt_sequence‘)) ; INSERT INTO "trace" VALUES(4503603928031644,43.5975,1.43099,0) ;

CREATE TABLE ‘traces‘ (‘idItineraire‘ bigint(20) NOT NULL, ‘idTrace‘ bigint(20) NOT NULL, PRIMARY KEY (‘idItineraire‘, ‘idTrace‘), CONSTRAINT fk_traces_idItineraire FOREIGN KEY (idItineraire) REFERENCES itineraire(idItineraire), CONSTRAINT fk_traces_idTrace FOREIGN KEY (idTrace) REFERENCES trace(idTrace)) ; INSERT INTO "traces" VALUES(4503603928074023,4503603928031644) ;

COMMIT ;

7 A propos

Auteur

Rialy Farel Dira <rialy23@gmail.com>

Version

1.0

Date

2016

8 Licence GPL

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

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

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

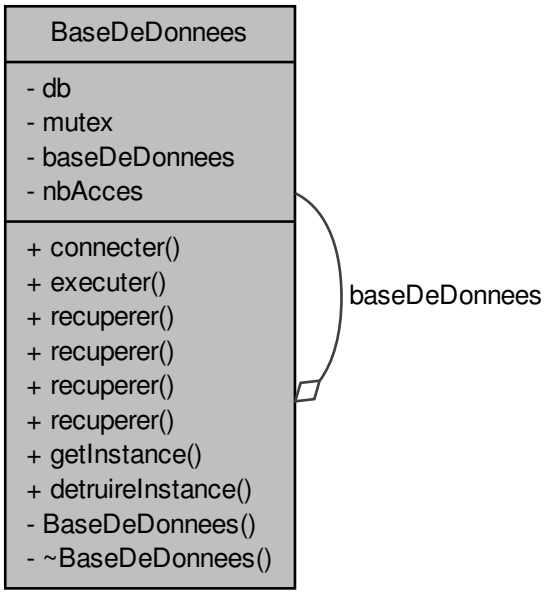
9 Documentation des classes

9.1 Référence de la classe BaseDeDonnees

Classe singleton + ressource critique protégée par un mutex.

`#include <basededonnees.h>`

Graphe de collaboration de BaseDeDonnees :



Fonctions membres publiques

- bool connecter (QString nomReferentiel="referentiel-sai.sqlite")
- bool executer (QString requete)
  Pour les requêtes SQL UPDATE, INSERT et DELETE.
- bool recuperer (QString requete, QVector< QString > &donnees)
  Requête SQL SELECT pour récupérer un seul champ de plusieurs enregistrements : le champ des différents enregistrements est stocké dans un QVector de QString.
- bool recuperer (QString requete, QVector< QStringList > &donnees)
  Requête SQL SELECT pour récupérer plusieurs champs de plusieurs enregistrements : les différents champs des différents enregistrements sont stockés dans un QVector de QStringList.
- bool recuperer (QString requete, QString &donnees)
  Requête SQL SELECT pour récupérer un seul champ d'un seul enregistrement : le champ est stocké dans un QString.
- bool recuperer (QString requete, QStringList &donnees)
  Requête SQL SELECT pour récupérer plusieurs champs d'un seul enregistrement : les différents champs sont stockés dans un QStringList.

Fonctions membres publiques statiques

- static BaseDeDonnees * getInstance ()
- static void detruireInstance ()

Fonctions membres privées

- BaseDeDonnees ()
- ~BaseDeDonnees ()

Attributs privés

– QSqlDatabase db
– QMutex mutex

Attributs privés statiques

– static BaseDeDonnees * baseDeDonnees = NULL
– static int nbAcces = 0

9.1.1 Description détaillée

Avertissement

Il faut ajouter QT += sql au fichier .pro

9.1.2 Documentation des constructeurs et destructeur

9.1.2.1 BaseDeDonnees : :BaseDeDonnees ( ) [private]

Références db.

Référencé par getInstance().

```
{
    #ifdef DEBUG_BASEDEDONNEES_1
    qDebug() << "<BaseDeDonnees::BaseDeDonnees()>";
    #endif
    db = QSqlDatabase::addDatabase("QSQLITE");
}
```

9.1.2.2 BaseDeDonnees : :~BaseDeDonnees ( ) [private]

```
{
    #ifdef DEBUG_BASEDEDONNEES_1
    qDebug() << "<BaseDeDonnees::~BaseDeDonnees()>";
    #endif
}
```

9.1.3 Documentation des fonctions membres

9.1.3.1 BaseDeDonnees : :connecter (QString *nomReferentiel* = "referentiel-sai.sqlite")

Paramètres

nomReferentiel	QString

Renvoie

bool

Références db, et mutex.

Référencé par IHM : :demarrer().

```
{
    QMutexLocker verrou(&mutex);

    if(db.isOpen())
    {
        #ifdef DEBUG_BASEDEDONNEES
        qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()> connexion
       active");
        #endif

        return true;
    }

    db.setDatabaseName(QCoreApplication::applicationDirPath() + "/referentiel/"
       + nomReferentiel);
    db.open();
    if(!db.isOpen())
    {
```

```
        //qDebug() << db.lastError().text();
        qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()> erreur :
       impossible de se connecter à la base de données %1 !").arg(
      QCoreApplication::applicationDirPath() + "/referentiel/" + nomReferentiel);

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

        return false;
    }

    #ifdef DEBUG_BASEDEDONNEES
    qDebug() << QString::fromUtf8("<BaseDeDonnees::connecter()> connexion
       réussie");
    #endif

    return true;
}
```

9.1.3.2 BaseDeDonnees : :detruireInstance ( ) [static]

Références baseDeDonnees, et nbAcces.

```
{
    // instance ?
    if(baseDeDonnees != NULL)
    {
        nbAcces--;
        #ifdef DEBUG_BASEDEDONNEES_1
        qDebug() << "<BaseDeDonnees::detruireInstance()> nbAcces restants = " <
      < nbAcces;
        #endif
        // dernier ?
        if(nbAcces == 0)
            delete baseDeDonnees;
    }
}
```

9.1.3.3 BaseDeDonnees : :executer (QString *requete*)

Paramètres

requete	QString

Renvoie

bool

Références db, et mutex.

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

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

            return false;
        }
    }
    else
        return false;

}
```

9.1.3.4 BaseDeDonnees : :getInstance ( ) [static]

Renvoie

BaseDeDonnees

Références BaseDeDonnees(), baseDeDonnees, et nbAcces.

Référencé par IHM : :demarrer().

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

    nbAcces++;
    #ifdef DEBUG_BASEDEDONNEES_1
    qDebug() << "<BaseDeDonnees::getInstance()> nbAcces = " << nbAcces;
    #endif

    return baseDeDonnees;
}
```

9.1.3.5 BaseDeDonnees : :recuperer (QString *requete,* QVector< QString > & *donnees*)

Paramètres

requete	QString
donnees	QVector<QString>

Renvoie

bool

Références db, et mutex.

Référencé par IHM : :afficherServicesVitesse(), IHM : :estEnCourse(), IHM : :recupererArret(), IHM : :recupererArrets(), IHM-: :recupererServices(), IHM : :recupererSuivi(), et IHM : :recupererTraces().

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

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

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

                // on stocke l'enregistrement dans le QVector
                donnees.push_back(data);
            }
            #ifdef DEBUG_BASEDEDONNEES
            qDebug() << "<BaseDeDonnees::recuperer(QString, QVector<QString>)>
       enregistrement -> " << donnees;
            #endif
            return true;
        }
        else
        {
            qDebug() << QString::fromUtf8("<BaseDeDonnees::recuperer(QString,
       QVector<QString>)> erreur : %1 pour la requête %2").arg(r.lastError().text()).arg
      (requete);

            return false;
        }
```

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

9.1.3.6 BaseDeDonnees : :recuperer (QString *requete,* QVector< QStringList > & *donnees*)

Paramètres

requete	QString
donnees	QVector<QStringList>

Renvoie

bool

Références db, et mutex.

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

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

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

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

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

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

9.1.3.7 BaseDeDonnees : :recuperer (QString *requete,* QString & *donnees*)

Paramètres

requete	QString
donnees	QString

Renvoie

bool

Références db, et mutex.

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

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

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

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

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

9.1.3.8 BaseDeDonnees : :recuperer (QString *requete,* QStringList & *donnees*)

Paramètres

requete	QString
donnees	QStringList

Renvoie

bool

Références db, et mutex.

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

    if(db.isOpen())
    {
        retour = r.exec(requete);
        #ifdef DEBUG_BASEDEDONNEES
```

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

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

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

9.1.4 Documentation des données membres

9.1.4.1 BaseDeDonnees * BaseDeDonnees : :baseDeDonnees = NULL [static, private]

TODO

Référencé par detruireInstance(), et getInstance().

9.1.4.2 QSqlDatabase BaseDeDonnees : :db [private]

TODO

Référencé par BaseDeDonnees(), connecter(), executer(), et recuperer().

9.1.4.3 QMutex BaseDeDonnees : :mutex [private]

TODO

Référencé par connecter(), executer(), et recuperer().

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

TODO

Référencé par detruireInstance(), et getInstance().

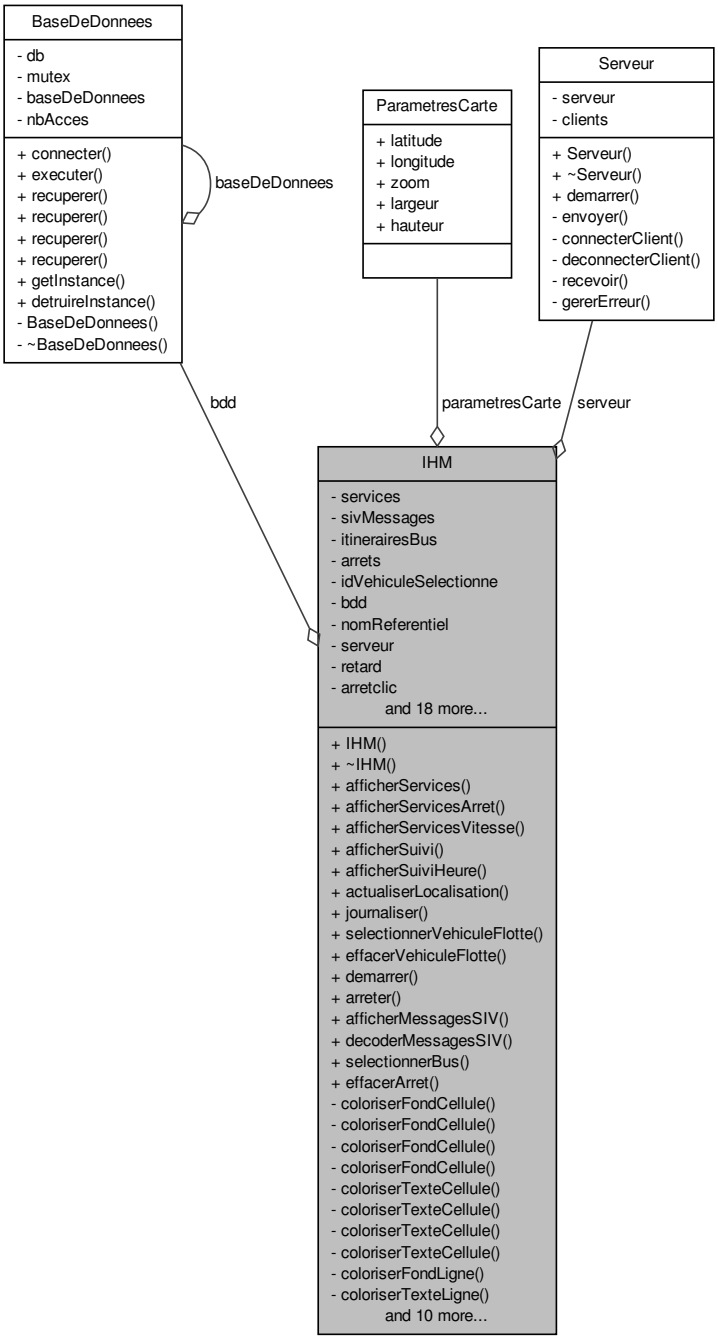
La documentation de cette classe a été générée à partir des fichiers suivants :
– basededonnees.h
– basededonnees.cpp

9.2 Référence de la classe IHM

Class.

`#include <ihm.h>`

Graphe de collaboration de IHM :



Connecteurs publics

- void selectionnerVehiculeFlotte (QTableWidgetItem ∗item)
  La fonction qui est appelée lorsqu'on selectionne le vehicule.
- void effacerVehiculeFlotte (QTableWidgetItem ∗item)
  La fonction qui est appelée apres un double click.
- void demarrer ()
  Pour demmarrer le serveur.
- void arreter ()
  Pour arreter le serveur.
- void afficherMessagesSIV (QString messageSIV)
  Pour afficher le message envoyé par le SIV.

- void decoderMessagesSIV (QString messageSIV)
  Pour decoder le message envoyé par le SIV.
- void selectionnerBus (Geometry *layer, QPoint point)
  La fonction qui est appelée lorsqu'on selectionne le vehicule sur la carte.
- void effacerArret ()

Fonctions membres publiques

- IHM (QWidget *parent=0)
  Contructeur par défaut.
- ~IHM ()
  Le destructeur de la classe.
- void afficherServices (QVector< QStringList > services)
  Pour afficher les services de la journée.
- void afficherServicesArret (QStringList arret)
  Pour afficher l'arrêt en cours et le prochain arret.
- void afficherServicesVitesse (QStringList vitesse)
  Pour afficher la vitesse la direction et l'état du service.
- void afficherSuivi (QString idVehicule, QVector< QStringList > suivi)
  Pour afficher le suivi d'une course.
- void afficherSuiviHeure (QString idVehicule, QStringList arret)
  Pour afficher l'heure d'arrivée réel et départ réel et l'avance retard.
- void actualiserLocalisation (QStringList localisation)
  Pour actualiser les positions du bus sur la carte.
- void journaliser (const QString &message)
  Pour afficher dans la partie journalisation.

Fonctions membres privées

- void coloriserFondCellule (QTableWidgetItem *cellule, const QColor &couleur)
- void coloriserFondCellule (QTableWidgetItem *cellule, const QString &couleur)
- void coloriserFondCellule (QTableWidget *tableWidget, int ligne, int colonne, const QColor &couleur)
- void coloriserFondCellule (QTableWidget *tableWidget, int ligne, int colonne, const QString &couleur)
- void coloriserTexteCellule (QTableWidgetItem *cellule, const QColor &couleur)
- void coloriserTexteCellule (QTableWidgetItem *cellule, const QString &couleur)
- void coloriserTexteCellule (QTableWidget *tableWidget, int ligne, int colonne, const QColor &couleur)
- void coloriserTexteCellule (QTableWidget *tableWidget, int ligne, int colonne, const QString &couleur)
- void coloriserFondLigne (QTableWidget *tableWidget, int ligne, const QString &couleur)
- void coloriserTexteLigne (QTableWidget *tableWidget, int ligne, const QString &couleur)
- QVector< QStringList > recupererServices ()
- QVector< QStringList > recupererSuivi (QString idItineraire)
  pour recuperer le suivi d'une course
- bool recupererArrets (QString idItineraire, QVector< QStringList > &arrets)
  Pour recuperer l'ensembles des arrets pour un itineraire.
- bool recupererArret (QString nomLieu, QStringList &arret)
  Pour recuperer un arret dans une liste.
- void recupererTraces (QString idItineraire, QVector< QStringList > &trace)
  Pour recuperer les traces d'un itineraire.
- double calculerDuree (QString heureReel, QString heureTheorique)
  Pour calculer la durée entre 2 heures.
- void geocaliserVehicule (QString longitude, QString latitude, QString idVehicule)
  Pour geolocaliser le vehicule sur la carte.
- void dessinerCourse (QString idVehicule, QString idItineraire)
  Pour dessiner le tracé d'un itineraire.
- void lireParametresCarte ()
  Pour recuperer les données pour localiser la ville sur la carte.
- bool estEnService (QString idVehicule, QString idService, QString idItineraire)
  Pour verifier si le vehicule est en service.
- bool estEnCourse (QString idService, QString idItineraire)
  Pour verifier si on est en course.

Attributs privés

- QVector< QStringList > services
- QVector< QStringList > sivMessages
- QMap< QString, QString > itinerairesBus
- QMap< QString, QVector < QStringList > > arrets
- QString idVehiculeSelectionne
- BaseDeDonnees * bdd
- QString nomReferentiel
- Serveur * serveur

- double retard
- bool arretclic
- QTimer * timerArret
- QTableWidget * informationsFlotte
- QTableWidget * suivisCourse
- QLabel * labelInformationsFlotte
- QLabel * labelsuivisCourse
- QTextEdit * journal
- QLabel * nomClicCarte
- QPushButton * bDemarrer
- QPushButton * bArreter
- ParametresCarte parametresCarte
- MapControl * mc
- TileMapAdapter * mapadapter
- Layer * l
- Layer * layerParcours
- QMap< QString, Layer * > mapParcours
- Layer * layerarret
- Layer * layerBus
- QMap< QString, Layer * > mapLayer
- QMap< QString, QList< Point * > > mapPoint

9.2.1 Description détaillée

Auteur

Rialy Farel Dira <rialy23@gmail.com>

9.2.2 Documentation des constructeurs et destructeur

9.2.2.1 IHM : :IHM (QWidget * *parent* = 0)

Paramètres

parent	

Références arretclic, arreter(), bArreter, bDemarrer, demarrer(), effacerArret(), effacerVehiculeFlotte(), ParametresCarte : :hauteur, idVehiculeSelectionne, informationsFlotte, journal, journaliser(), l, labelInformationsFlotte, labelsuivisCourse, ParametresCarte : :largeur, ParametresCarte : :latitude, lireParametresCarte(), ParametresCarte : :longitude, mapadapter, mc, nomClicCarte, nomReferentiel, parametresCarte, selectionnerVehiculeFlotte(), serveur, suivisCourse, timerArret, et ParametresCarte : :zoom.

```
                        : QWidget( parent )
{
    // les widgets
    labelInformationsFlotte = new QLabel(this);
    labelInformationsFlotte->setText(QString::fromUtf8("Services"));
    labelInformationsFlotte->setAlignment(Qt::AlignHCenter);
    informationsFlotte = new QTableWidget(0, 11, this);
    //informationsFlotte->setRowCount(0);
    informationsFlotte->setEditTriggers(QAbstractItemView::NoEditTriggers);
    informationsFlotte->setHorizontalHeaderLabels(QStringList() << "État" << "
      Bus" << "Ligne" << "Conducteur" << "Destination" << "Arrêt" << "Prochain" << "A/R
      " << "km/h" << "Heure" << "Service" );
    informationsFlotte->verticalHeader()->setHidden(true);
    informationsFlotte->setFixedHeight(200);
    informationsFlotte->setColumnWidth(0, 120);
    informationsFlotte->setColumnWidth(1, 40);
    informationsFlotte->setColumnWidth(2, 50);
    informationsFlotte->setColumnWidth(3, 150);
    informationsFlotte->setColumnWidth(4, 150);
    informationsFlotte->setColumnWidth(5, 80);
    informationsFlotte->resizeColumnToContents(6);
    informationsFlotte->resizeColumnToContents(7);
    informationsFlotte->resizeColumnToContents(8);
    informationsFlotte->resizeColumnToContents(9);

    QHeaderView * headerViewFlotte = informationsFlotte->horizontalHeader();
    //headerViewFlotte->setResizeMode(QHeaderView::ResizeToContents);
    headerViewFlotte->setStretchLastSection(true);
    //headerViewFlotte->setResizeMode(QHeaderView::Stretch);

    //boutons pour connecter les bus
    nomClicCarte = new QLabel(this);
    bDemarrer  = new QPushButton(QString::fromUtf8("Démarrer"), this);
    bDemarrer->setEnabled(true);
    bArreter   = new QPushButton(QString::fromUtf8("Arrêter"), this);
    bArreter->setEnabled(false);

    // insertion tableau Suivis
```

```
labelsuivisCourse = new QLabel(this);
labelsuivisCourse->setText(QString::fromUtf8("Suivis"));
labelsuivisCourse->setAlignment(Qt::AlignHCenter);
suivisCourse = new QTableWidget(0, 8, this);
suivisCourse->setEditTriggers(QAbstractItemView::NoEditTriggers);
suivisCourse->setHorizontalHeaderLabels(QStringList() << "N" << "Arret" <<
  "Départ théorique" << "Départ Réel" << "Av(-)/Rd(+)" << "Arrivé théorique" << "
  Arrivée Réel" << "Av(-)/Rd(+)" );
suivisCourse->verticalHeader()->setHidden(true);
suivisCourse->setColumnWidth(0, 30);
suivisCourse->setColumnWidth(1, 200);
suivisCourse->setColumnWidth(2, 150);
suivisCourse->setColumnWidth(3, 90);
suivisCourse->setColumnWidth(4, 100);
suivisCourse->setColumnWidth(5, 150);
suivisCourse->resizeColumnToContents(6);
suivisCourse->resizeColumnToContents(7);

QHeaderView * headerViewSuivi = suivisCourse->horizontalHeader();
//headerViewSuivi->setResizeMode(QHeaderView::Stretch);
headerViewSuivi->setStretchLastSection(true);
//headerViewSuivi->setResizeMode(QHeaderView::ResizeToContents);

//Pour la journalisation
journal = new QTextEdit(this);
journal->setReadOnly(true);
journal->setFixedHeight(150);

lireParametresCarte();

// On instancie la map
mc = new MapControl(QSize(parametresCarte.largeur,parametresCarte.hauteur))
  ;
mc->showScale(true);
mapadapter = new TileMapAdapter("otile1.mqcdn.com", "
  /tiles/1.0.0/osm/%1/%2/%3.png", 256, 0, 17);
l = new Layer("Custom Layer", mapadapter, Layer::MapLayer);
mc->addLayer(l);
mc->setView(QPointF(parametresCarte.latitude, parametresCarte.longitude));
  //Toulouse !
mc->setZoom(parametresCarte.zoom);
mc->setFixedWidth((parametresCarte.largeur)+1);
mc->setFixedHeight((parametresCarte.hauteur)+1);
QPushButton* zoomin = new QPushButton("+");
QPushButton* zoomout = new QPushButton("-");
zoomin->setMaximumWidth(50);
zoomout->setMaximumWidth(50);

// Connection avec les slots
connect(zoomin, SIGNAL(clicked(bool)), mc, SLOT(zoomIn()));
connect(zoomout, SIGNAL(clicked(bool)), mc, SLOT(zoomOut()));

QVBoxLayout* innerlayout = new QVBoxLayout;
innerlayout->addWidget(zoomin);
innerlayout->addWidget(zoomout);
innerlayout->addStretch(1);
mc->setLayout(innerlayout);


// mise ne place des layout

QHBoxLayout *hLayout0 = new QHBoxLayout;
QHBoxLayout *hLayout1 = new QHBoxLayout;
QHBoxLayout *hLayout2 = new QHBoxLayout;
QHBoxLayout *hLayout3 = new QHBoxLayout;
QHBoxLayout *hLayout4 = new QHBoxLayout;
QHBoxLayout *hLayout5a = new QHBoxLayout;
QHBoxLayout *hLayout5 = new QHBoxLayout;
QHBoxLayout *hLayout6 = new QHBoxLayout;

QVBoxLayout* layoutGauche = new QVBoxLayout;
QVBoxLayout* layoutDroit = new QVBoxLayout;
QHBoxLayout *mainLayout = new QHBoxLayout;

hLayout0->addWidget(informationsFlotte);
hLayout1->addWidget(labelInformationsFlotte);
hLayout2->addWidget(suivisCourse);
hLayout3->addWidget(labelsuivisCourse);
hLayout4->addWidget(journal);
layoutGauche->addWidget(mc);
hLayout5a->addWidget(nomClicCarte);
hLayout5->addWidget(bDemarrer);
hLayout5->addWidget(bArreter);

layoutGauche->addSpacing(15);
layoutGauche->addLayout(hLayout5a);
layoutGauche->addLayout(hLayout5);

layoutGauche->addStretch(1);
layoutDroit->addLayout(hLayout0);
layoutDroit->addLayout(hLayout1);
```

```
    layoutDroit->addLayout(hLayout2);
    layoutDroit->addLayout(hLayout3);
    layoutDroit->addLayout(hLayout4);
    //layoutDroit->addStretch(1);
    mainLayout->addLayout(layoutGauche);
    mainLayout->addLayout(layoutDroit);
    setLayout(mainLayout);

    /* démarrage plein écran */
    const int width = qApp->desktop()->width();
    const int height = qApp->desktop()->height();
    resize(width, height);
    /* reste en avant plan */
    //setWindowFlags(Qt::WindowStaysOnTopHint | Qt::FramelessWindowHint);

    // Initialisation
    serveur = NULL;

    QString message = "Démarrage SAI";
    journaliser(message);

    connect(informationsFlotte, SIGNAL(itemDoubleClicked(QTableWidgetItem*)),
      this, SLOT(effacerVehiculeFlotte(QTableWidgetItem *)));
    connect(informationsFlotte, SIGNAL(itemClicked(QTableWidgetItem*)), this,
      SLOT(selectionnerVehiculeFlotte(QTableWidgetItem *)));
    connect(informationsFlotte,SIGNAL(itemActivated(QTableWidgetItem*)),this,
      SLOT(selectionnerVehiculeFlotte(QTableWidgetItem*)));
    connect(bDemarrer, SIGNAL(clicked()), this, SLOT(demarrer()));
    connect(bArreter, SIGNAL(clicked()), this, SLOT(arreter()));

    idVehiculeSelectionne = "";
    nomReferentiel = "referentiel-sai.sqlite";

    QVector<QStringList> suivi;
    arretclic = false;
    timerArret = new QTimer(this);
    timerArret->setInterval(5000);
    connect(timerArret, SIGNAL(timeout()), this, SLOT(effacerArret()));

}
```

9.2.2.2 IHM : :~IHM ( )

```
{
}
```

9.2.3 Documentation des fonctions membres

9.2.3.1 IHM : :actualiserLocalisation (QStringList *localisation*)

Paramètres

localisation	

Références COLONNE_NUMERO_BUS, dessinerCourse(), geocaliserVehicule(), et informationsFlotte.

Référencé par decoderMessagesSIV().

```
{
    // Extraction des données utiles
    QString idBus = localisation.at(3);
    QString latitude = localisation.at(7);
    QString longitude = localisation.at(9);
    QString data;
    QStringList service;
    bool trouve = false;
    //int ligne = -1;

    // Identifie la ligne concernée dans la table à partir du numéro de bus
    int nb = informationsFlotte->rowCount();
    for(int i = 0; i < nb; i++)
    {
        if(informationsFlotte->item(i, COLONNE_NUMERO_BUS)->text() == idBus)
        {
            //ligne = i;
            trouve = true;
            break;
        }
    }

    if(trouve == true)
    {
        geocaliserVehicule(longitude,latitude,idBus);
        dessinerCourse(idBus, localisation.at(6));
```

```
    }
}
```

9.2.3.2 IHM : :afficherMessagesSIV (QString *messageSIV*) [slot]

Paramètres

messageSIV	

Références journaliser().

Référencé par arreter(), et decoderMessagesSIV().

```
{
   QString message = "<IHM::afficherDonnee()> donnees recues : " + messageSIV;
   journaliser(message);
}
```

9.2.3.3 IHM : :afficherServices (QVector< QStringList > *services*)

Paramètres

services	

Références COLONNE_AVANCE_RETARD, COLONNE_DESTINATION, COLONNE_ETAT, COLONNE_HORODATAGE, COLONNE_NOM_ARRET, COLONNE_NOM_CONDUCTEUR, COLONNE_NUMERO_BUS, COLONNE_NUMERO_LIGNE, COLONNE_NUMERO_SERVICE, COLONNE_PROCHAIN_ARRET, COLONNE_VITESSE, informationsFlotte, SERVICE_HEURE_DEBUT, SERVICE_ID_SERVICE, SERVICE_ID_VEHICULE, et SERVICE_NOM_CONDUCTEUR.

Référencé par demarrer().

```
{
    QString data;
    QStringList service;

    // Efface la table
    int nb = informationsFlotte->rowCount();
    for(int i = 0; i < nb; i++)
    {
        informationsFlotte->removeRow(0);
    }

    nb = informationsFlotte->rowCount();
    for(int i = 0; i < services.size(); i++)
    {
        informationsFlotte->setRowCount(++nb);
        service = services.at(i);
        #ifdef DEBUG_IHM
        qDebug() << "<IHM::afficherServices()> service : " << service;
        #endif

        data = "En attente";
        //data = "<span style=\" color:#ff0000;\">En attente</span>";
        informationsFlotte->setItem(nb-1, COLONNE_ETAT, new QTableWidgetItem(
      data));
        informationsFlotte->item(nb-1, COLONNE_ETAT)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = service.at(SERVICE_ID_VEHICULE);
        informationsFlotte->setItem(nb-1, COLONNE_NUMERO_BUS, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_NUMERO_BUS)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        informationsFlotte->setItem(nb-1, COLONNE_NUMERO_LIGNE, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_NUMERO_LIGNE)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = service.at(SERVICE_NOM_CONDUCTEUR);
        informationsFlotte->setItem(nb-1, COLONNE_NOM_CONDUCTEUR, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_NOM_CONDUCTEUR)->
      setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        informationsFlotte->setItem(nb-1, COLONNE_DESTINATION, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_DESTINATION)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        informationsFlotte->setItem(nb-1, COLONNE_NOM_ARRET, new
      QTableWidgetItem(data));
```

```
        informationsFlotte->item(nb-1, COLONNE_NOM_ARRET)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        informationsFlotte->setItem(nb-1, COLONNE_PROCHAIN_ARRET, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_PROCHAIN_ARRET)->
      setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        informationsFlotte->setItem(nb-1, COLONNE_AVANCE_RETARD, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_AVANCE_RETARD)->setTextAlignment
      (Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        informationsFlotte->setItem(nb-1, COLONNE_VITESSE, new QTableWidgetItem
      (data));
        informationsFlotte->item(nb-1, COLONNE_VITESSE)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = service.at(SERVICE_HEURE_DEBUT);
        informationsFlotte->setItem(nb-1, COLONNE_HORODATAGE, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_HORODATAGE)->setTextAlignment(
      Qt::AlignHCenter|Qt::AlignVCenter);
        data = service.at(SERVICE_ID_SERVICE);
        informationsFlotte->setItem(nb-1, COLONNE_NUMERO_SERVICE, new
      QTableWidgetItem(data));
        informationsFlotte->item(nb-1, COLONNE_NUMERO_SERVICE)->
      setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
    }
}
```

9.2.3.4 IHM : :afficherServicesArret (QStringList *arret*)

Paramètres

arret	

Références COLONNE_AVANCE_RETARD, COLONNE_ETAT, COLONNE_HORODATAGE, COLONNE_NOM_ARRET, COLONNE_NUMERO_BUS, COLONNE_PROCHAIN_ARRET, COLONNE_VITESSE, informationsFlotte, recupererSuivi(), retard, et SERVICE_ARRET.

Référencé par decoderMessagesSIV().

```
{
    // Exemple de QStringList arret :
    // ("ARRET", "27/04/2016", "06:23:20.000", "155", "1", "11821949021891623",
       "4503603928075026", "Jean Jaurès", "EA")

    // Extraction des données utiles
    QString date = arret.at(1);
    QString heure = arret.at(2);
    QString idBus = arret.at(3);
    QString idService = arret.at(4);
    QString nomArret = arret.at(7);
    QString idItineraire = arret.at(6);
    QString data;
    QStringList service;
    QStringList ligneTrouve;
    QString prochainArret;
    bool trouve = false;
    int ligne = -1;

    // Identifie la ligne concernée dans la table à partir du numéro de bus
    int nb = informationsFlotte->rowCount();
    for(int i = 0; i < nb; i++)
    {
        if(informationsFlotte->item(i, COLONNE_NUMERO_BUS)->text() == idBus)
        {
            ligne = i;
            trouve = true;
            break;
        }
    }

    QVector<QStringList> suivi = recupererSuivi(idItineraire);

    for(int i = 0; i < suivi.size()-1; i++)
    {
        QStringList ligneTrouve = suivi.at(i);
        QString arret = ligneTrouve.at(SERVICE_ARRET);
        if(nomArret == arret )
        {
            ligneTrouve = suivi.at(i+1);
            arret = ligneTrouve.at(SERVICE_ARRET);
            prochainArret = arret;
            #ifdef DEBUG_IHM
            qDebug() << "<IHM::afficherServicesArret()> arret trouvé : " <<
```

```
      prochainArret;
            #endif
        }
    }

    if(trouve == true)
    {
        informationsFlotte->item(ligne, COLONNE_HORODATAGE)->setText(heure);
        informationsFlotte->item(ligne, COLONNE_NOM_ARRET)->setText(nomArret);
        informationsFlotte->item(ligne, COLONNE_PROCHAIN_ARRET)->setText(
      prochainArret);
        informationsFlotte->item(ligne, COLONNE_VITESSE)->setText("0");
        informationsFlotte->item(ligne, COLONNE_ETAT)->setText("A l'arrêt");
        informationsFlotte->item(ligne, COLONNE_AVANCE_RETARD)->setText(
      QString::number(retard));
     }
}
```

9.2.3.5 IHM : :afficherServicesVitesse (QStringList *vitesse*)

Paramètres

vitesse	

Références bdd, COLONNE_DESTINATION, COLONNE_ETAT, COLONNE_NUMERO_BUS, COLONNE_NUMERO_LIGNE, - COLONNE_VITESSE, informationsFlotte, et BaseDeDonnees : :recuperer().

Référencé par decoderMessagesSIV().

```
{
    // Exemple de QStringList vitesse :
    // ("ARRET", "27/04/2016", "06:23:20.000", "155", "1", "11821949021891623",
       "4503603928075026", "Jean Jaurès", "EA")
    // Extraction des données utiles
    QString idBus = localisation.at(3);
    QString vitesse = localisation.at(12);
    QString data;
    QStringList service;
    QString direction = localisation.at(11);
    QString idItineraire = localisation.at(6);
    QString idLigne = localisation.at(5);
    QString NomLigne;

    bool trouve = false;
    int ligne = -1;

    // Identifie la ligne concernée dans la table à partir du numéro de bus
    int nb = informationsFlotte->rowCount();
    for(int i = 0; i < nb; i++)
    {
        if(informationsFlotte->item(i, COLONNE_NUMERO_BUS)->text() == idBus)
        {
            ligne = i;
            trouve = true;
            break;
        }
    }

    QString requeteDirection = " SELECT destination FROM itineraire WHERE
       idItineraire =" + idItineraire;
    bdd->recuperer(requeteDirection, direction);
    QString requeteNomLigne = " Select nomCourt  FROM ligne WHERE  idLIgne =" +
       idLigne;
    bdd->recuperer(requeteNomLigne, NomLigne);
    if(trouve == true)
    {
        informationsFlotte->item(ligne, COLONNE_DESTINATION)->setText(direction
      );
        informationsFlotte->item(ligne, COLONNE_VITESSE)->setText(vitesse);
        informationsFlotte->item(ligne, COLONNE_NUMERO_LIGNE)->setText(NomLigne
      );
        if (informationsFlotte->item(ligne, COLONNE_ETAT)->text()== "En attente
      ")
        {
            informationsFlotte->item(ligne, COLONNE_ETAT)->setText("Prise de
       service");
        }
        else
        {
            informationsFlotte->item(ligne, COLONNE_ETAT)->setText("En
       déplacement");
        }
    }
}
```

9.2.3.6 IHM : :afficherSuivi (QString *idVehicule,* QVector< QStringList > *suivi*)

Paramètres

idService	
suivi	

Références arrets, calculerDuree(), COLONNE_AVANCE, COLONNE_SEQUENCE, COLONNE_SERVICE_ARRET, COLONNE_SERVICE_DATE_DEBUT_REEL, COLONNE_SERVICE_HEURE_ARRIVE_REEL, COLONNE_SERVICE_HEURE_ARRIVE_THEORIQUE, COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE, COLONNE_TEMPS_ARRIVE, coloriserFondLigne(), retard, SEQUENCE, SERVICE_ARRET, SERVICE_HEURE_ARRIVE_THEORIQUE, SERVICE_HEURE_DEBUT_THEORIQUE, et suivisCourse.

Référencé par selectionnerBus(), et selectionnerVehiculeFlotte().

```
{

    QString data;
    QStringList arret;
    int nb;

    // Efface la table
    nb = suivisCourse->rowCount();
    for(int i = 0; i < nb; i++)
    {
        suivisCourse->removeRow(0);
    }

    nb = suivisCourse->rowCount();
    for(int i = 0; i < suivi.size(); i++)
    {
        suivisCourse->setRowCount(++nb);
        arret = suivi.at(i);
        #ifdef DEBUG_IHM
        qDebug() << Q_FUNC_INFO << arret;
        #endif
        data = arret.at(SEQUENCE);
        suivisCourse->setItem(nb-1, COLONNE_SEQUENCE, new QTableWidgetItem(data
    ));
        suivisCourse->item(nb-1, COLONNE_SEQUENCE)->setTextAlignment(
    Qt::AlignHCenter|Qt::AlignVCenter);
        data = arret.at(SERVICE_ARRET);
        suivisCourse->setItem(nb-1, COLONNE_SERVICE_ARRET, new QTableWidgetItem
    (data));
        suivisCourse->item(nb-1, COLONNE_SERVICE_ARRET)->setTextAlignment(
    Qt::AlignHCenter|Qt::AlignVCenter);
        data = arret.at(SERVICE_HEURE_DEBUT_THEORIQUE);
        suivisCourse->setItem(nb-1, COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE, new
    QTableWidgetItem(data));
        suivisCourse->item(nb-1, COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE)->
    setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        suivisCourse->setItem(nb-1, COLONNE_SERVICE_DATE_DEBUT_REEL, new
    QTableWidgetItem(data));
        suivisCourse->item(nb-1, COLONNE_SERVICE_DATE_DEBUT_REEL)->
    setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        suivisCourse->setItem(nb-1, COLONNE_AVANCE, new QTableWidgetItem(data))
    ;
        suivisCourse->item(nb-1, COLONNE_AVANCE)->setTextAlignment(
    Qt::AlignHCenter|Qt::AlignVCenter);
        data = arret.at(SERVICE_HEURE_ARRIVE_THEORIQUE);
        suivisCourse->setItem(nb-1, COLONNE_SERVICE_HEURE_ARRIVE_THEORIQUE, new
     QTableWidgetItem(data));
        suivisCourse->item(nb-1, COLONNE_SERVICE_HEURE_ARRIVE_THEORIQUE)->
    setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        suivisCourse->setItem(nb-1, COLONNE_SERVICE_HEURE_ARRIVE_REEL, new
    QTableWidgetItem(data));
        suivisCourse->item(nb-1, COLONNE_SERVICE_HEURE_ARRIVE_REEL)->
    setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);
        data = "";
        suivisCourse->setItem(nb-1, COLONNE_TEMPS_ARRIVE, new QTableWidgetItem(
    data));
        suivisCourse->item(nb-1, COLONNE_TEMPS_ARRIVE)->setTextAlignment(
    Qt::AlignHCenter|Qt::AlignVCenter);
    }

    if(!arrets.contains(idVehicule))
        return;
    else
    {
        QVector<QStringList> tablesuivi;
        tablesuivi = arrets[idVehicule] ;
        for (int i = 0; i < tablesuivi.size() ;i++ )
        {
```

```
            QStringList ligne = tablesuivi.at(i);
            QString heure = ligne.at(2);
            suivisCourse->item(i, COLONNE_SERVICE_DATE_DEBUT_REEL)->setText(
      heure);
            suivisCourse->item(i, COLONNE_SERVICE_HEURE_ARRIVE_REEL)->setText(
      heure);
            //on re-calcule  l'avance retard entre les heures d'arrives reel et
       théoriques qu'on a sauvegardé
            QString Htheorique = suivisCourse->item(i,
      COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE)->text();
            double duree = calculerDuree(heure,Htheorique);
            retard = duree;
            QString dureeH = QString::number(duree);
            suivisCourse->item(i, COLONNE_AVANCE)->setText(dureeH);
            suivisCourse->item(i, COLONNE_TEMPS_ARRIVE)->setText(dureeH);
            coloriserFondLigne(suivisCourse, tablesuivi.size()-1, "#fff000");
        }
     }
    //suivisCourse->resizeColumnsToContents();
}
```

9.2.3.7 IHM : :afficherSuiviHeure (QString *idVehicule,* QStringList *arret*)

Paramètres

idVehicule	
arret	

Références arrets, calculerDuree(), COLONNE_AVANCE, COLONNE_SERVICE_ARRET, COLONNE_SERVICE_DATE_DEBUT_REEL, COLONNE_SERVICE_HEURE_ARRIVE_REEL, COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE, COLONNE_TEMPS_ARRIVE, coloriserFondLigne(), idVehiculeSelectionne, retard, et suivisCourse.

Référencé par decoderMessagesSIV().

```
{
    //QMap<QString,QVector<QStringList> > arrets;
    if(arrets.contains(idVehicule))
    {
        arrets[idVehicule].push_back(arret);
    }
    else
    {
        QVector<QStringList> a;
        a.push_back(arret);
        arrets[idVehicule] = a;
    }
    #ifdef DEBUG_IHM
    qDebug() << Q_FUNC_INFO << arrets;
    #endif

    // Mettre à jour l'affichage pour ce vehicule à suivre ?
    if(idVehicule != idVehiculeSelectionne)
        return;
    else
    {
        QVector<QStringList> suivi;
        suivi = arrets[idVehicule] ;
        for (int i = 1; i < suivi.size() ;i++ )
        {
            QStringList ligne = suivi.at(i);
            QString heure = ligne.at(2);
            //on re-calcule  l'avance retard entre les heures d'arrives reel et
       théoriques qu'on a sauvegardé
            QString Htheorique = suivisCourse->item(i,
      COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE)->text();
            double duree = calculerDuree(heure,Htheorique);
            retard = duree;
            QString dureeH = QString::number(duree);
            //suivisCourse->item(i,
       COLONNE_SERVICE_DATE_DEBUT_REEL)->setText(heure);
            //suivisCourse->item(i,
       COLONNE_SERVICE_HEURE_ARRIVE_REEL)->setText(heure);
            //suivisCourse->item(i, COLONNE_TEMPS_ARRIVE)->setText(dureeH);
            //suivisCourse->item(i, COLONNE_AVANCE)->setText(dureeH);
         }
    }

    // Extraction des données utiles
    QString heureReel = arret.at(2);
    QString nomArret = arret.at(7);

    bool trouve = false;
    int ligne = -1;

    int nb = suivisCourse->rowCount();
```

```
    for(int i = 0; i < nb; i++)
    {
        if(suivisCourse->item(i, COLONNE_SERVICE_ARRET)->text() == nomArret)
        {
            ligne = i;
            trouve = true;
            break;
        }
    }

    QString heureTheorique;
    if(trouve == true)
    {
        heureTheorique = suivisCourse->item(ligne,
      COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE)->text();
    }

    QString avance;
    if (!heureTheorique.isEmpty())
    {
        double calcul = calculerDuree(heureReel,heureTheorique);
        avance = QString::number(calcul);
    }

    if(trouve == true)
    {
        suivisCourse->item(ligne, COLONNE_SERVICE_DATE_DEBUT_REEL)->setText(
      heureReel);
        suivisCourse->item(ligne, COLONNE_SERVICE_HEURE_ARRIVE_REEL)->setText(
      heureReel);
        suivisCourse->item(ligne, COLONNE_AVANCE)->setText(avance);
        suivisCourse->item(ligne, COLONNE_TEMPS_ARRIVE)->setText(avance);
        coloriserFondLigne(suivisCourse, ligne-1, "#ffffff");
        coloriserFondLigne(suivisCourse, ligne, "#fff000");
    }
}
```

9.2.3.8 IHM : :arreter ( ) [slot]

Références afficherMessagesSIV(), bArreter, bDemarrer, et serveur.

Référencé par IHM().

```
{
   if(serveur != NULL)
   {
      qDebug() << Q_FUNC_INFO;
      disconnect(serveur, SIGNAL(donneesRecues(QString)), this, SLOT(
      afficherMessagesSIV(QString)));
      delete serveur;
      bArreter->setEnabled(false);
      bDemarrer->setEnabled(true);
   }
}
```

9.2.3.9 IHM : :calculerDuree (QString *heureReel,* QString *heureTheorique*) [private]

Paramètres

heureReel	
heureTheorique	

Renvoie

double

Référencé par afficherSuivi(), et afficherSuiviHeure().

```
{
    QStringList tempsreel = heureReel.split(':');
    QStringList tempstheorique = heureTheorique.split(':');
    double duree = 0.;

    #ifdef DEBUG_IHM
    qDebug() << "<IHM::calculerDuree()> tempsreel : " << tempsreel <<"
      tempstheorique"<<tempstheorique;
    #endif

    if(tempsreel.size() != 3)
        return duree;

    if(tempstheorique.size() != 3)
```

```
        return duree;

    double heureR = tempsreel.at(0).toDouble();
    double minuteR = tempsreel.at(1).toDouble();
    //double secondeR = tempsreel.at(2).toDouble();

    double heureT = tempstheorique.at(0).toDouble();
    double minuteT = tempstheorique.at(1).toDouble();
    //double secondeT = tempstheorique.at(2).toDouble();

    duree = ((heureR - heureT)*60.) + (minuteR - minuteT);

    #ifdef DEBUG_IHM
    qDebug() << "<IHM::calculerDuree()> duree : " << duree;
    #endif

    return duree;
}
```

9.2.3.10 IHM : :coloriserFondCellule (QTableWidgetItem * *cellule*, const QColor & *couleur*) [private]

Paramètres

cellule	
couleur	

Référencé par coloriserFondCellule(), et coloriserFondLigne().

```
{
    cellule->setBackgroundColor(couleur);
}
```

9.2.3.11 IHM : :coloriserFondCellule (QTableWidgetItem * *cellule*, const QString & *couleur*) [private]

Paramètres

cellule	
couleur	

Références coloriserFondCellule().

```
{
    QColor _couleur;
    _couleur.setNamedColor(couleur);
    coloriserFondCellule(cellule, _couleur);
}
```

9.2.3.12 IHM : :coloriserFondCellule (QTableWidget * *tableWidget*, int *ligne*, int *colonne*, const QColor & *couleur*) [private]

Paramètres

tableWidget	
ligne	
colonne	
couleur	

```
{
    tableWidget->item(ligne, colonne)->setBackgroundColor(couleur);
}
```

9.2.3.13 IHM : :coloriserFondCellule (QTableWidget * *tableWidget*, int *ligne*, int *colonne*, const QString & *couleur*) [private]

Paramètres

tableWidget	
ligne	
colonne	
couleur	

Références coloriserFondCellule().

```
{
    QColor _couleur;
    _couleur.setNamedColor(couleur);
    coloriserFondCellule(tableWidget, ligne, colonne, _couleur);
}
```

9.2.3.14 IHM : :coloriserFondLigne (QTableWidget * *tableWidget*, int *ligne*, const QString & *couleur*) [private]

Paramètres

tableWidget	
ligne	
couleur	

Références coloriserFondCellule().

Référencé par afficherSuivi(), afficherSuiviHeure(), et selectionnerVehiculeFlotte().

```
{
    QColor _couleur;
    _couleur.setNamedColor(couleur);
    for(int i = 0; i < tableWidget->columnCount(); i++)
        coloriserFondCellule(tableWidget, ligne, i, _couleur);
}
```

9.2.3.15 IHM : :coloriserTexteCellule (QTableWidgetItem * *cellule*, const QColor & *couleur*) [private]

Paramètres

cellule	
couleur	

Référencé par coloriserTexteCellule(), et coloriserTexteLigne().

```
{
    cellule->setForeground(couleur);
}
```

9.2.3.16 IHM : :coloriserTexteCellule (QTableWidgetItem * *cellule*, const QString & *couleur*) [private]

Paramètres

cellule	
couleur	

Références coloriserTexteCellule().

```
{
    QColor _couleur;
    _couleur.setNamedColor(couleur);
    coloriserTexteCellule(cellule, _couleur);
}
```

9.2.3.17 IHM : :coloriserTexteCellule (QTableWidget * *tableWidget*, int *ligne*, int *colonne*, const QColor & *couleur*) [private]

Paramètres

tableWidget	
ligne	
colonne	
couleur	

```
{
    tableWidget->item(ligne, colonne)->setForeground(couleur);
}
```

9.2.3.18 IHM : :coloriserTexteCellule (QTableWidget * *tableWidget*, int *ligne*, int *colonne*, const QString & *couleur*) [private]

Paramètres

tableWidget	
ligne	
colonne	
couleur	

Références coloriserTexteCellule().

```
{
    QColor _couleur;
    _couleur.setNamedColor(couleur);
    coloriserTexteCellule(tableWidget, ligne, colonne, _couleur);
}
```

9.2.3.19 IHM : :coloriserTexteLigne (QTableWidget * *tableWidget*, int *ligne*, const QString & *couleur*) [private]

Paramètres

tableWidget	
ligne	
couleur	

Références coloriserTexteCellule().

```
{
    QColor _couleur;
    _couleur.setNamedColor(couleur);
    for(int i = 0; i < tableWidget->columnCount(); i++)
        coloriserTexteCellule(tableWidget, ligne, i, _couleur);
}
```

9.2.3.20 IHM : :decoderMessagesSIV (QString *messageSIV*) [slot]

Paramètres

messageSIV	

Références actualiserLocalisation(), afficherMessagesSIV(), afficherServicesArret(), afficherServicesVitesse(), afficherSuiviHeure(), estEnService(), itinerairesBus, et journaliser().

Référencé par demarrer().

```
{
    QString idVehicule;
    QString idService;
    QString idItineraire;

    afficherMessagesSIV(messageSIV);

    // extraction messages des données reçues :
    // chaque message se termine par \r\n (cf. protocole)
    QStringList messages = messageSIV.split("\r\n");
    qDebug() << Q_FUNC_INFO << messages;

    // pour chaque message reçu
    for(int i = 0; i < messages.size(); i++)
    {
        if(!messages[i].isEmpty())
        {
            QStringList champs = messages[i].split(';');
            qDebug() << Q_FUNC_INFO << champs;

            //QVector<QStringList> champsLocalisation;
            //QStringList champsArret;

            if(champs.length() != 0)
            {
                idVehicule = champs.at(3);
                idService = champs.at(4);
                idItineraire = champs.at(6);

                // est-ce que ce véhicule pour ce service sur cet itineraire
   existe ?
                if(!estEnService(idVehicule, idService, idItineraire))
                {
                    // NON :
                    QString messageErreur = "<b><font color=red>Le vehicule " +
```

```
        idVehicule + " du service " + idService + " pour l'itineraire " + idItineraire
       + " n'est pas connu du SAI !</font></b>";
                journaliser(messageErreur);
                continue; // on ignore et on passe au message suivant
            }
            // OUI :
            itinerairesBus[idVehicule] = idItineraire;

            #ifdef DEBUG_IHM
            qDebug() << "<IHM::decoderMessagesSIV()> idVehicule : " <<
    idVehicule << " idService : " << idService << " idItineraire : " << idItineraire;
            QString message = "<IHM::decoderMessagesSIV()> idVehicule : " +
     idVehicule + " idService : " + idService + " idItineraire : " + idItineraire;
            journaliser(message);
            #endif
        }

        const QString typeTrameLocalisation = "LOCALISATION";
        const QString typeTrameArret = "ARRET";
        const QString typeTrameAlerte = "ALERTE";

        if(champs.length() != 0)
        {
            if(messages[i].startsWith(typeTrameLocalisation))
            {
                qDebug() << "Information : trame LOCALISATION recue";
                actualiserLocalisation(champs);
                afficherServicesVitesse(champs);
            }
            else if(messages[i].startsWith(typeTrameArret))
            {
                qDebug() << "Information : trame ARRET recue";
                afficherSuiviHeure(idVehicule, champs);
                afficherServicesArret(champs);
            }
            else if(messages[i].startsWith(typeTrameAlerte))
            {
                qDebug() << "Information : trame ALERTE recue";
            }
            else
                qDebug() << "Erreur : trame inconnue !";
        }
        else
            qDebug() << "Erreur : trame vide !";
      }
    }
}
```

9.2.3.21 IHM : :demarrer ( ) [slot]

Références afficherServices(), bArreter, bdd, bDemarrer, BaseDeDonnees : :connecter(), decoderMessagesSIV(), Serveur : :demarrer(), BaseDeDonnees : :getInstance(), nomReferentiel, recupererServices(), retard, serveur, et services.

Référencé par IHM().

```
{
   if(serveur == NULL || serveur != NULL)
   {
      //Connection a la base de donnée
      bdd = BaseDeDonnees::getInstance();
      bdd->connecter(nomReferentiel);
      retard=0;
      services = recupererServices();
      afficherServices(services);
      qDebug() << Q_FUNC_INFO;
      serveur = new Serveur;
      serveur->demarrer();
      bArreter->setEnabled(true);
      bDemarrer->setEnabled(false);
      connect(serveur, SIGNAL(donneesRecues(QString)), this, SLOT(
      decoderMessagesSIV(QString)));
   }
}
```

9.2.3.22 IHM : :dessinerCourse (QString *idVehicule*, QString *idItineraire*) [private]

Paramètres

idVehicule	
idItineraire	

Références arrets, journaliser(), layerParcours, mapadapter, mapParcours, mc, recupererArrets(), recupererTraces(), et selectionnerBus().

Référencé par actualiserLocalisation().

```
{
    QVector<QStringList> arrets;
    QVector<QStringList> trace;
    bool ok;

    recupererTraces(idItineraire, trace);

    QString message = "<IHM::dessinerCourse()> idItineraire : " + idItineraire;
    journaliser(message);

    if(trace.isEmpty())
    {
        return;
    }

    if (mapParcours.contains(idVehicule))
    {
        return;
    }
    else
    {
        layerParcours = new GeometryLayer("Parcours " + idItineraire, mapadapter
      );
        mapParcours[idVehicule] = layerParcours;
        connect(mapParcours.value(idVehicule), SIGNAL(geometryClicked(Geometry*
      , QPoint)), this, SLOT(selectionnerBus(Geometry*, QPoint)));
    }

    ok = recupererArrets(idItineraire, arrets);

    QPen* pen = new QPen(QColor(QColor(255, 0, 0, 100))); // on trace avec la
       couleur indiquée par le référentiel
    pen->setWidth(5);

    QList<Point*> points;
    // l'itinéraire détaillé
    for (int i = 0; i < trace.size(); ++i)
    {
       QString latitude = trace.at(i).at(0);
       QString longitude = trace.at(i).at(1);
       points.append(new Point(longitude.toDouble(), latitude.toDouble(), "",
      Point::Middle));
    }

    // on trace l'itinéraire détaillé sur la carte
    layerParcours->addGeometry(new LineString(points, "itineraire", pen));

    if(ok)
    {
        // le départ
        points.clear();
        QString arret = arrets.at(0).at(1);
        QString latitude = arrets.at(0).at(4);
        QString longitude = arrets.at(0).at(5);
        points.append(new ImagePoint(longitude.toDouble(), latitude.toDouble(),
       QCoreApplication::applicationDirPath() + "/images/bus-icone.png", arret,
      Point::Middle));
        layerParcours->addGeometry(new LineString(points, "depart", pen));

        // les arrêts
        QPen* pointpen = new QPen(QColor(0,0,0));
        pointpen->setWidth(3);
        for(int i=1; i < arrets.size()-1; i++)
        {
            points.clear();
            arret = arrets.at(i).at(1);
            latitude = arrets.at(i).at(4);
            longitude = arrets.at(i).at(5);
            points.append(new CirclePoint(longitude.toDouble(), latitude.
      toDouble(), 8, arret, Point::Middle, pointpen));
            layerParcours->addGeometry(new LineString(points, arret, pen));

        }
        // l'arrivée
        points.clear();
        arret = arrets.at(arrets.size()-1).at(1);
        latitude = arrets.at(arrets.size()-1).at(4);
        longitude = arrets.at(arrets.size()-1).at(5);
        points.append(new ImagePoint(longitude.toDouble(), latitude.toDouble(),
       QCoreApplication::applicationDirPath() + "/images/flag.png", arret,
      Point::Middle));
        layerParcours->addGeometry(new LineString(points, "arrivee", pen));
    }

    mc->addLayer(layerParcours);
    mc->update();

    message = "<IHM::dessinerCourse()> idVehicule : " + idVehicule;
```

```
    journaliser(message);
}
```

9.2.3.23 IHM : :effacerArret ( ) [slot]

Références arretclic, layerarret, mc, et timerArret.

Référencé par IHM().

```
{
    if (arretclic == true)
    {
        layerarret->clearGeometries();
        arretclic = false;
        mc->update();
        timerArret->stop();
    }
}
```

9.2.3.24 IHM : :effacerVehiculeFlotte (QTableWidgetItem * *item*) [slot]

Paramètres

item	

Références COLONNE_ETAT, COLONNE_NUMERO_BUS, informationsFlotte, et journaliser().

Référencé par IHM().

```
{
    int ligne = item->row();

    QString idVehicule = informationsFlotte->item(ligne, COLONNE_NUMERO_BUS)->
      text();
    QString etat = informationsFlotte->item(ligne, COLONNE_ETAT)->text();
    //if(etat == "Fin de service")
    {
        informationsFlotte->removeRow(ligne);
        #ifdef DEBUG_IHM
        qDebug() << "<IHM::effacerVehiculeFlotte()> idVehicule : " <<
      idVehicule << " etat : " << etat;
        QString message = "<IHM::effacerVehiculeFlotte()> idVehicule : " +
      idVehicule + " etat : " + etat;
        journaliser(message);
        #endif
    }
}
```

9.2.3.25 IHM : :estEnCourse (QString *idService*, QString *idItineraire*) [private]

Paramètres

idService	
idItineraire	

Renvoie

bool

Références bdd, et BaseDeDonnees : :recuperer().

Référencé par estEnService().

```
{
    bool retour = false;
    QString effectue;
    QString requete = "SELECT effectue FROM course WHERE idItineraire='" +
      idItineraire + "' AND idService='" + idService + "'";

    retour = bdd->recuperer(requete, effectue);
    if(retour != false)
    {
        if(!effectue.isEmpty() && effectue == "0")
        {
            return true;
        }
        else
        {
```

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

9.2.3.26 IHM : :estEnService (QString *idVehicule,* QString *idService,* QString *idItineraire*) [private]

Paramètres

idVehicule	
idService	
idItineraire	

Renvoie

bool

Références estEnCourse(), SERVICE_ID_SERVICE, SERVICE_ID_VEHICULE, et services.

Référencé par decoderMessagesSIV().

```
{
    QStringList service;

    for(int i = 0; i < services.size(); i++)
    {
        service = services.at(i);
        if(service.at(SERVICE_ID_VEHICULE) == idVehicule && service.at(
      SERVICE_ID_SERVICE) == idService)
        {
            if(estEnCourse(idService, idItineraire))
                return true;
        }
    }
    return false;
}
```

9.2.3.27 IHM : :geocaliserVehicule (QString *longitude,* QString *latitude,* QString *idVehicule*) [private]

Paramètres

longitude	
latitude	
idVehicule	

Références journaliser(), layerBus, mapadapter, mapLayer, mapPoint, mc, retard, et selectionnerBus().

Référencé par actualiserLocalisation().

```
{

    //Simulation
    //latitude = "43.563";
    //longitude = "1.42577";
    //idVehicule = "155";
    //Fin Simulation

    bool present = false;
    QList<Point*>* pointsBus;

    QString localisation = "<IHM::geocaliserVehicule()> idVehicule : " +
      idVehicule;
    localisation += " longitude : " + longitude + " latitude : " + latitude;
    journaliser(localisation);

    if (mapLayer.contains(idVehicule))
    {
        layerBus = mapLayer[idVehicule];
        pointsBus = &(mapPoint[idVehicule]);
        pointsBus->clear();
        present = true;
    }
    else
    {
```

```
        layerBus = new GeometryLayer("Parcours " + idVehicule, mapadapter);
        mapLayer[idVehicule] = layerBus;
        connect(mapLayer.value(idVehicule), SIGNAL(geometryClicked(Geometry*,
      QPoint)), this, SLOT(selectionnerBus(Geometry*, QPoint)));
        pointsBus = new QList<Point*>;
        mapPoint[idVehicule] = *pointsBus;
    }

    QPen* pen = new QPen(QColor(255, 0, 0, 100));
    pen->setWidth(5);
    if(retard == 0)
    {
        pointsBus->append(new ImagePoint(longitude.toDouble(), latitude.
      toDouble(), QCoreApplication::applicationDirPath() + "/images/bus.png", "",
      Point::Middle));
    }
    else
    {
        pointsBus->append(new ImagePoint(longitude.toDouble(), latitude.
      toDouble(), QCoreApplication::applicationDirPath() + "/images/bus-avance.png", "",
      Point::Middle));

    }
    layerBus->clearGeometries();
    LineString* ls = new LineString(*pointsBus, idVehicule, pen);
    layerBus->addGeometry(ls);
    if(present == false)
    {
        mc->addLayer(layerBus);
    }
    mc->update();
}
```

9.2.3.28 IHM : :journaliser (const QString & *message*)

Paramètres

message	

Références journal.

Référencé par afficherMessagesSIV(), decoderMessagesSIV(), dessinerCourse(), effacerVehiculeFlotte(), geocaliserVehicule(), IHM(), lireParametresCarte(), recupererArret(), recupererArrets(), recupererServices(), recupererSuivi(), recupererTraces(), et selectionnerVehiculeFlotte().

```
{
    journal->append(message);
}
```

9.2.3.29 IHM : :lireParametresCarte ( ) [private]

Références ParametresCarte : :hauteur, journaliser(), ParametresCarte : :largeur, ParametresCarte : :latitude, ParametresCarte-: :longitude, parametresCarte, et ParametresCarte : :zoom.

Référencé par IHM().

```
{
    QSettings parametres("sai.ini", QSettings::IniFormat);

    parametresCarte.latitude = parametres.value("carte/latitude", "1.445").
      toDouble();
    parametresCarte.longitude = parametres.value("carte/longitude", "43.605").
      toDouble();
    parametresCarte.zoom = parametres.value("carte/zoom", "12").toInt();
    parametresCarte.largeur = parametres.value("carte/largeur", "750").toInt();
    parametresCarte.hauteur = parametres.value("carte/hauteur", "500").toInt();

    QString message = "latitude : " + QString::number(parametresCarte.latitude)
       + " longitude : " + QString::number(parametresCarte.longitude);
    journaliser(message);
}
```

9.2.3.30 IHM : :recupererArret (QString *nomLieu*, QStringList & *arret*) [private]

Paramètres

nomLieu	
arret	

Renvoie

bool

Références bdd, journaliser(), et BaseDeDonnees : :recuperer().

Référencé par selectionnerBus(), et selectionnerVehiculeFlotte().

```
{
    QString message;
    bool retour = false;
    bool valide = false;
    QString requeteArret = "SELECT lieu.latitude, lieu.longitude FROM lieu
       where nomLieu = '" + nomLieu + "'" ;

    retour = bdd->recuperer(requeteArret, arret);
    if(retour != false)
    {
        if(!arret.isEmpty())
        {
            valide = true;
            //qDebug() << Q_FUNC_INFO << arret;
        }
        else
        {
            message = "<IHM::recupererArret()> erreur aucun arret !" ;
            journaliser(message);
            valide = false;
        }
    }
    else
    {
        message = "<IHM::recupererArret()> erreur requete " + requeteArret + "
       !";
        journaliser(message);
        valide = false;
    }
    return valide;
}
```

9.2.3.31 IHM : :recupererArrets (QString *idItineraire,* QVector< QStringList > & *arrets*) [private]

Paramètres

idItineraire	
arrets	

Renvoie

bool

Références bdd, journaliser(), et BaseDeDonnees : :recuperer().

Référencé par dessinerCourse().

```
{
    QString message;
    bool retour = false;
    bool valide = false;
    QString requeteArrets = "SELECT arret.idArret, nomLieu, heureArrivee,
       heureDepart, latitude, longitude, numeroSequence FROM itineraire INNER JOIN arret ON
       arret.idItineraire=itineraire.idItineraire INNER JOIN lieu ON
       lieu.idArret=arret.idArret WHERE itineraire.idItineraire='" + idItineraire + "' ORDER BY
       numeroSequence ASC";

    retour = bdd->recuperer(requeteArrets, arrets);
    if(retour != false)
    {
        if(!arrets.isEmpty())
        {
            valide = true;
            //qDebug() << Q_FUNC_INFO << arrets;
        }
        else
        {
            message = "<IHM::recupererArrets()> erreur aucun arret !" ;
            journaliser(message);
            valide = false;
        }
    }
    else
    {
        message = "<IHM::recupererArrets()> erreur requete " + requeteArrets +
      " !";
```

```
        journaliser(message);
        valide = false;
    }

    return valide;
}
```

9.2.3.32 IHM : :recupererServices ( ) [private]

Renvoie

QVector<QStringList>

Références bdd, journaliser(), BaseDeDonnees : :recuperer(), et services.

Référencé par demarrer().

```
{
    QString dateDebut;
    QString heureDebut;
    //Simulation
    dateDebut = "2015-07-11";
    heureDebut = "07:00:00";
    //Fin Simulation
    QString message;
    bool retour = false;
    QVector<QStringList> services;
    //QString requeteServices = "SELECT service.idService,
       service.codeConducteur, service.idVehicule, conducteur.nomconducteur, service.dateDebut,
       service.heureDebut, service.effectue FROM service INNER JOIN conducteur ON
       conducteur.codeconducteur = service.codeConducteur WHERE service.dateDebut = \"" + dateDebut +
       "\" AND service.heureDebut = \"" + heureDebut + "\" AND service.effectue = 0  ";
    QString requeteServices = "SELECT service.idService,
       service.codeConducteur, service.idVehicule, conducteur.nomconducteur, service.dateDebut,
       service.heureDebut, service.effectue FROM service INNER JOIN conducteur ON
       conducteur.codeconducteur = service.codeConducteur WHERE service.effectue = 0  ";
    //message = requeteServices;
    //journaliser(message);

    retour = bdd->recuperer(requeteServices, services);
    if(retour != false)
    {
        if(!services.isEmpty())
        {
            //qDebug() << Q_FUNC_INFO << services;
        }
        else
        {
            message = "<IHM::recupererServices()> erreur aucun service !" ;
            journaliser(message);
        }
    }
    else
    {
        message = "<IHM::recupererServices()> erreur requete !" ;
        journaliser(message);
    }
    return services;
}
```

9.2.3.33 IHM : :recupererSuivi (QString *idItineraire*) [private]

Paramètres

idItineraire	

Renvoie

QVector<QStringList>

Références bdd, journaliser(), et BaseDeDonnees : :recuperer().

Référencé par afficherServicesArret(), selectionnerBus(), et selectionnerVehiculeFlotte().

```
{
    QString message;
    bool retour = false;
    QVector<QStringList> suivi;
    QString requeteSuivi = "SELECT  numeroSequence, nomLieu, heureArrivee,
       heureDepart FROM itineraire INNER JOIN arret ON
       arret.idItineraire=itineraire.idItineraire INNER JOIN lieu ON lieu.idArret=arret.idArret WHERE
       itineraire.idItineraire='" + idItineraire + "' ORDER BY numeroSequence ASC";
```

```
    retour = bdd->recuperer(requeteSuivi, suivi);
    if(retour != false)
    {
        if(!suivi.isEmpty())
        {
            //qDebug() << Q_FUNC_INFO << suivi;
        }
        else
        {
            message = "<IHM::recupererSuivi()> erreur aucun suivi !" ;
            journaliser(message);
        }
    }
    else
    {
        message = "<IHM::recupererSuivi()> erreur requete !" ;
        journaliser(message);
    }
    return suivi;
}
```

9.2.3.34 IHM : :recupererTraces (QString *idItineraire,* QVector< QStringList > & *trace*) [private]

Paramètres

idItineraire	
trace	

Références bdd, journaliser(), et BaseDeDonnees : :recuperer().

Référencé par dessinerCourse().

```
{
    QString message;
    bool retour = false;
    QString requeteTraces = "SELECT trace.pt_lat, trace.pt_lon  FROM trace
       INNER JOIN traces ON traces.idTrace=trace.idTrace WHERE traces.idItineraire='" +
      idItineraire + "'ORDER BY trace.pt_sequence ASC";
    retour = bdd->recuperer(requeteTraces, trace);
    if(retour != false)
    {
        if(!trace.isEmpty())
        {
            //qDebug() << Q_FUNC_INFO << trace;
        }
        else
        {
            message = "<IHM::recupererTraces()> erreur aucune trace !" ;
            journaliser(message);
        }
    }
    else
    {
        message = "<IHM::recupererTraces()> erreur requete " + requeteTraces +
      " !";
        journaliser(message);
    }
}
```

9.2.3.35 IHM : :selectionnerBus (Geometry * *layer,* QPoint *point*) [slot]

Paramètres

layer	l'id de bus
point	non utilisé

Références afficherSuivi(), arretclic, COLONNE_NUMERO_BUS, informationsFlotte, itinerairesBus, layerarret, mapadapter, mc, nomClicCarte, recupererArret(), recupererSuivi(), selectionnerVehiculeFlotte(), SERVICE_ID_VEHICULE, services, et timerArret.

Référencé par dessinerCourse(), et geocaliserVehicule().

```
{
    //qDebug() << Q_FUNC_INFO << "point" << point;
    QString nom = layer->name();
    qDebug() << Q_FUNC_INFO << __LINE__ << "nom" << nom;
    nomClicCarte->setText(nom);

    QString idVehicule = "";
    for(int i = 0; i < services.size(); i++)
    {
```

```
        // idVehicule en service ?
        if(services.at(i).at(SERVICE_ID_VEHICULE) == nom)
        {
            idVehicule = nom;
            break;
        }
    }

    // idVehicule trouve ?
    if(idVehicule != "")
    {
        QString idItineraire;
        if (itinerairesBus.contains(idVehicule))
        {
            idItineraire = itinerairesBus[idVehicule];
            QVector<QStringList> suivi;
            suivi = recupererSuivi(idItineraire);
            afficherSuivi(idVehicule, suivi);
        }
        int nb = informationsFlotte->rowCount();
        for(int i = 0; i < nb; i++)
        {
            QTableWidgetItem *item = informationsFlotte->item(i,
      COLONNE_NUMERO_BUS);
            if(item->text() == idVehicule)
            {
                selectionnerVehiculeFlotte(item);
                break;
            }
        }
    }

    if (arretclic == true)
    {
        layerarret->clearGeometries();
        arretclic = false;
    }
    if(idVehicule!=nom)
    {
        //On recupere les coordonnees de l'arret
        QStringList coordonnes;;
        bool etat = recupererArret(nom,coordonnes);
        if(etat == true)
        {
            QString latitude = coordonnes.at(0);
            QString longitude = coordonnes.at(1);
            QList<Point*> points;
            QPen* pen = new QPen(QColor(QColor(255, 0, 0, 150))); // on trace
       avec la couleur indiquée par le référentiel
            pen->setWidth(5);
            points.append(new Point(longitude.toDouble(), latitude.toDouble(),
      "", Point::Middle));

            layerarret = new GeometryLayer("Arret " + nom, mapadapter);

            points.clear();
            points.append(new ImagePoint(longitude.toDouble(), latitude.
      toDouble(), QCoreApplication::applicationDirPath() + "/images/bus_stop.png", nom,
      Point::Middle));
            layerarret->addGeometry(new LineString(points, "arret", pen));
            if (arretclic == false)
            {
                mc->addLayer(layerarret);
                arretclic =true;
                timerArret->start();
            }
            mc->update();
        }
    }

}
```

9.2.3.36 IHM : :selectionnerVehiculeFlotte (QTableWidgetItem * *item*) [slot]

Paramètres

item	

Références afficherSuivi(), COLONNE_AVANCE_RETARD, COLONNE_ETAT, COLONNE_HORODATAGE, COLONNE_-NOM_ARRET, COLONNE_NUMERO_BUS, COLONNE_NUMERO_SERVICE, coloriserFondLigne(), idVehiculeSelectionne, informationsFlotte, itinerairesBus, journaliser(), mc, parametresCarte, recupererArret(), recupererSuivi(), et ParametresCarte-: :zoom.

Référencé par IHM(), et selectionnerBus().

```
{
```

```
    int ligne = item->row();
    QString etat = informationsFlotte->item(ligne, COLONNE_ETAT)->text();
    QString idVehicule = informationsFlotte->item(ligne, COLONNE_NUMERO_BUS)->
      text();
    QString nomArret = informationsFlotte->item(ligne, COLONNE_NOM_ARRET)->text
      ();
    QString heureHorodatage = informationsFlotte->item(ligne, COLONNE_HORODATAGE
      )->text();
    QString idService = informationsFlotte->item(ligne, COLONNE_NUMERO_SERVICE)
      ->text();
    QString avanceRetard = informationsFlotte->item(ligne, COLONNE_AVANCE_RETARD
      )->text();
    QString idItineraire;
    QVector<QStringList> suivi;

    if (itinerairesBus.contains(idVehicule))
    {
        idItineraire = itinerairesBus[idVehicule];
        #ifdef DEBUG_IHM
        qDebug() << "<IHM::selectionnerVehiculeFlotte()> idVehicule : " <<
      idVehicule << " idService : " << idService << " idItineraire : " << idItineraire;
        QString message = "<IHM::selectionnerVehiculeFlotte()> idVehicule : " +
       idVehicule + " idService : " + idService + " idItineraire : " + idItineraire;
        journaliser(message);
        #endif
    }
    else
    {
        #ifdef DEBUG_IHM
        qDebug() << "<IHM::selectionnerVehiculeFlotte()> idVehicule : " <<
      idVehicule << " idService : " << idService << " pas d'itineraire !";
        QString message = "<IHM::selectionnerVehiculeFlotte()> idVehicule : " +
       idVehicule + " idService : " + idService + " pas d'itineraire !";
        journaliser(message);
        #endif
        return;
    }

    #ifdef DEBUG_IHM
    qDebug() << "<IHM::selectionnerVehiculeFlotte()> idVehicule : " <<
      idVehicule << " idService : " << idService << " nomArret " <<nomArret ;
    QString message = "<IHM::selectionnerVehiculeFlotte()> idVehicule : " +
      idVehicule + " idService : " + idService +" nomArret " + nomArret;
    journaliser(message);
    #endif

    idVehiculeSelectionne = idVehicule;
    // Sélectionner visuellement la ligne
    int nb = informationsFlotte->rowCount();
    for(int i = 0; i < nb; i++)
    {
        if(i == ligne)
            coloriserFondLigne(informationsFlotte, i, "#ff0000");
        else
            coloriserFondLigne(informationsFlotte, i, "#ffffff");
    }

    //coloriserTexteLigne(informationsFlotte, ligne, "#00ff00");
    //coloriserTexteCellule(informationsFlotte, ligne, 0, QString("#0000ff"));

    QStringList coordonnee;
    suivi = recupererSuivi(idItineraire);
    afficherSuivi(idVehicule,suivi);

    if(!nomArret.isEmpty())
    {
        if(recupererArret(nomArret, coordonnee))
        {
            //geocaliserVehicule(coordonnee.at(1),coordonnee.at(0),idVehicule);
            mc->setZoom(parametresCarte.zoom+1);
            mc->setView(QPointF(coordonnee.at(1).toDouble(), coordonnee.at(0).
      toDouble())); // se recentre sur l'arrêt
        }
    }
    //dessinerCourse(idVehicule, idItineraire);
}
```

9.2.4 Documentation des données membres

9.2.4.1 bool IHM : :arretclic [private]

Référencé par effacerArret(), IHM(), et selectionnerBus().

9.2.4.2 QMap<QString,QVector<QStringList> > IHM : :arrets [private]

Pour stocker les arrets

Référencé par afficherSuivi(), afficherSuiviHeure(), et dessinerCourse().

9.2.4.3 QPushButton* IHM : :bArreter [private]

relation avec le bouton Arreter

Référencé par arreter(), demarrer(), et IHM().

9.2.4.4 BaseDeDonnees* IHM : :bdd [private]

la relation avec la classe BaseDeDonnees

Référencé par afficherServicesVitesse(), demarrer(), estEnCourse(), recupererArret(), recupererArrets(), recupererServices(), recupererSuivi(), et recupererTraces().

9.2.4.5 QPushButton* IHM : :bDemarrer [private]

relation avec le bouton demarrer

Référencé par arreter(), demarrer(), et IHM().

9.2.4.6 QString IHM : :idVehiculeSelectionne [private]

Pour stocker le vehicule selectionné

Référencé par afficherSuiviHeure(), IHM(), et selectionnerVehiculeFlotte().

9.2.4.7 QTableWidget* IHM : :informationsFlotte [private]

Pour le tableau de Service

Référencé par actualiserLocalisation(), afficherServices(), afficherServicesArret(), afficherServicesVitesse(), effacerVehiculeFlotte(), IHM(), selectionnerBus(), et selectionnerVehiculeFlotte().

9.2.4.8 QMap<QString,QString> IHM : :itinerairesBus [private]

Conteneur pour stocker les itineraires pour les SIV

Référencé par decoderMessagesSIV(), selectionnerBus(), et selectionnerVehiculeFlotte().

9.2.4.9 QTextEdit* IHM : :journal [private]

Pour la journalisation

Référencé par IHM(), et journaliser().

9.2.4.10 Layer* IHM : :l [private]

Calque

Référencé par IHM().

9.2.4.11 QLabel* IHM : :labelInformationsFlotte [private]

Pour contenir le nom du tableau Service

Référencé par IHM().

9.2.4.12 QLabel* IHM : :labelsuivisCourse [private]

Pour contenir le nom du tableau Suivis

Référencé par IHM().

9.2.4.13 Layer* IHM : :layerarret [private]

Référencé par effacerArret(), et selectionnerBus().

9.2.4.14 Layer* IHM : :layerBus [private]

Calque pour le bus

Référencé par geocaliserVehicule().

9.2.4.15 Layer* IHM : :layerParcours [private]

Les calques pour les itineraires

Référencé par dessinerCourse().

9.2.4.16 TileMapAdapter* IHM : :mapadapter [private]

Tuile pour la carte qu'on va utiliser

Référencé par dessinerCourse(), geocaliserVehicule(), IHM(), et selectionnerBus().

9.2.4.17 QMap<QString, Layer*> IHM : :mapLayer [private]

Contient les calques pour les arrets

Référencé par geocaliserVehicule().

9.2.4.18 QMap<QString, Layer*> IHM : :mapParcours [private]

Conteneur pour stocker les Calques pour chaques idVehicules

Référencé par dessinerCourse().

9.2.4.19 QMap<QString, QList<Point*> > IHM : :mapPoint [private]

Pour contenir les idVehicules

Référencé par geocaliserVehicule().

9.2.4.20 MapControl* IHM : :mc [private]

La carte

Référencé par dessinerCourse(), effacerArret(), geocaliserVehicule(), IHM(), selectionnerBus(), et selectionnerVehiculeFlotte().

9.2.4.21 QLabel* IHM : :nomClicCarte [private]

le nom de l'élement sur lequel on a cliqué sur la carte

Référencé par IHM(), et selectionnerBus().

9.2.4.22 QString IHM : :nomReferentiel [private]

Pour stocker le nom du referentiel qu'on utilise

Référencé par demarrer(), et IHM().

9.2.4.23 ParametresCarte IHM : :parametresCarte [private]

Pour les parametres prédefini de la carte

Référencé par IHM(), lireParametresCarte(), et selectionnerVehiculeFlotte().

9.2.4.24 double IHM : :retard [private]

Avance retard

Référencé par afficherServicesArret(), afficherSuivi(), afficherSuiviHeure(), demarrer(), et geocaliserVehicule().

9.2.4.25 Serveur* IHM : :serveur [private]

la relation avec la classe Serveur

Référencé par arreter(), demarrer(), et IHM().

9.2.4.26 QVector<QStringList> IHM : :services [private]

Conteneur pour stocker la liste des services

Référencé par demarrer(), estEnService(), recupererServices(), et selectionnerBus().

9.2.4.27 QVector<QStringList> IHM : :sivMessages [private]

Conteneur pour stocker les messages pour chaque SIV

9.2.4.28 QTableWidget* IHM : :suivisCourse [private]

Pour le tableau pour le suivie d'une course

Référencé par afficherSuivi(), afficherSuiviHeure(), et IHM().

9.2.4.29 QTimer* IHM : :timerArret [private]

Référencé par effacerArret(), IHM(), et selectionnerBus().

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

9.3 Référence de la structure ParametresCarte

Pour les configurations de la carte.

`#include <ihm.h>`

Attributs publics

– double latitude
– double longitude
– int zoom
– int largeur
– int hauteur

9.3.1 Documentation des données membres

9.3.1.1 int ParametresCarte : :hauteur

hauteur du widget pour la carte

Référencé par IHM : :IHM(), et IHM : :lireParametresCarte().

9.3.1.2 int ParametresCarte : :largeur

largeur du widget pour la carte

Référencé par IHM : :IHM(), et IHM : :lireParametresCarte().

9.3.1.3 double ParametresCarte : :latitude

longitude du centre de la ville

Référencé par IHM : :IHM(), et IHM : :lireParametresCarte().

9.3.1.4 double ParametresCarte : :longitude

latitude du centre de dans la ville

Référencé par IHM : :IHM(), et IHM : :lireParametresCarte().

9.3.1.5 int ParametresCarte : :zoom

zoom prédefini pour voir la carte

Référencé par IHM : :IHM(), IHM : :lireParametresCarte(), et IHM : :selectionnerVehiculeFlotte().

La documentation de cette structure a été générée à partir du fichier suivant :
– ihm.h

9.4 Référence de la classe Serveur

La classe qui gere la connexion entre SIV et SAI.

`#include <serveur.h>`

Signaux

– void donneesRecues (QString messageSIV)

Fonctions membres publiques

– Serveur (QObject ∗parent=0)
Constructeur.
– ∼Serveur ()
Destructeur.
– void demarrer ()
Pour démarrage le serveur en écoute sur toutes les IP disponibles.

Connecteurs privés

– void connecterClient ()
Pour connecter le client au serveur.
– void deconnecterClient ()
Pour deconnecter les clients du serveur.
– void recevoir ()
Pour recevoir les messages du clients.
– void gererErreur (QAbstractSocket : :SocketError erreur)

Fonctions membres privées

– void envoyer (QTcpSocket ∗client, QString message)
Pour envoyer les trames.

Attributs privés

– QTcpServer ∗ serveur
– QList< QTcpSocket ∗ > clients

9.4.1 Documentation des constructeurs et destructeur

9.4.1.1 Serveur : :Serveur (QObject ∗ *parent* = 0)

Paramètres

parent	

Références serveur.

```
                                : QObject( parent )
{
   //qDebug() << QString::fromUtf8("Instancie un objet QTcpServer");
   serveur = new QTcpServer(this);
}
```

9.4.1.2 Serveur : :∼Serveur ( )

```
{
   //qDebug() << QString::fromUtf8("Ferme le serveur");
}
```

9.4.2 Documentation des fonctions membres

9.4.2.1 Serveur : :connecterClient ( ) [private, slot]

Références clients, deconnecterClient(), gererErreur(), recevoir(), et serveur.

Référencé par demarrer().

```
{
    QTcpSocket *nouveauClient = serveur->nextPendingConnection();

    qDebug() << QString::fromUtf8("Un SIV s'est connecté : ") << nouveauClient
```

```
      ->peerAddress().toString() << ":" << QString::number(nouveauClient->peerPort());
    clients << nouveauClient;

    connect(nouveauClient, SIGNAL(readyRead()), this, SLOT(recevoir()));
    connect(nouveauClient, SIGNAL(disconnected()), this, SLOT(deconnecterClient
      ()));
    connect(nouveauClient, SIGNAL(error(QAbstractSocket::SocketError)), this,
      SLOT(gererErreur(QAbstractSocket::SocketError)));
}
```

9.4.2.2 Serveur : :deconnecterClient ( ) [private, slot]

Références clients.

Référencé par connecterClient().

```
{
    // On détermine quel client se déconnecte
    QTcpSocket *client = qobject_cast<QTcpSocket *>(sender());

    if (client == 0) // Si par hasard on n'a pas trouvé le client à l'origine
       du signal, on arrête la méthode
        return;

    qDebug() << QString::fromUtf8("Un SIV s'est déconnecté : ") << client->
      peerAddress().toString() << ":" << QString::number(client->peerPort());
    clients.removeOne(client);

    client->deleteLater();
}
```

9.4.2.3 Serveur : :demarrer ( )

Références connecterClient(), PORT_SERVEUR, et serveur.

Référencé par IHM : :demarrer().

```
{
    // Démarrage du serveur en écoute sur toutes les IP disponibles (= toutes
       les interfaces locales du serveur) et sur le port 5000
    if (!serveur->listen(QHostAddress::Any, PORT_SERVEUR))
    {
        // Si le serveur n'a pas été démarré correctement
        qDebug() << QString::fromUtf8("Le serveur n'a pas pu être démarré.") <<
       serveur->errorString();
    }
    else
    {
        // Si le serveur a été démarré correctement
        qDebug() << QString::fromUtf8("Le serveur a été démarré sur le port ")
      << QString::number(serveur->serverPort());
        qDebug() << QString::fromUtf8("Adresse IP du Serveur : ") + serveur->
      serverAddress().toString();
        qDebug() << QString::fromUtf8("Numéro de port d'écoute du Serveur : ")
      + QString::number(serveur->serverPort());
        connect(serveur, SIGNAL(newConnection()), this, SLOT(connecterClient())
      );
    }
}
```

9.4.2.4 Serveur : :donneesRecues (QString *messageSIV*) [signal]

Paramètres

messageSIV	

Référencé par recevoir().

9.4.2.5 Serveur : :envoyer (QTcpSocket * *client*, QString *message*) [private]

Paramètres

client	QTcpSocket *
message	QString

```
{
   qDebug() << QString::fromUtf8("Adresse IP du SIV : ") + client->peerAddress(
      ).toString();
   qDebug() << QString::fromUtf8("Numéro de port du SIV : ") + QString::number(
```

```
      client->peerPort());

   QByteArray trame; // représente le message de la couche Application
   trame.append(message);
   qint64 ecrits = -1;

   // Envoie du message
   ecrits = client->write(trame);
   switch(ecrits)
   {
      case -1 : // une erreur !
         qDebug() << QString::fromUtf8("Erreur lors de l'envoi !"); break;
      default: // envoi de n octets
         qDebug() << QString::fromUtf8("Message envoyé au SIV : ") << trame;
         //qDebug() << QString::fromUtf8("Octets envoyés : ") << ecrits;
         //qDebug() << QString::fromUtf8("Message envoyé avec succès !");
   }
}
```

9.4.2.6 Serveur : :gererErreur (QAbstractSocket : :SocketError *erreur*) [private, slot]

Paramètres

erreur	

Référencé par connecterClient().

```
{
   //qDebug() << QString::fromUtf8("Une erreur s'est produite sur la socket :
       ") << tcpSocket->errorString();
   qDebug() << QString::fromUtf8("Erreur : ") << erreur;

}
```

9.4.2.7 Serveur : :recevoir ( ) [private, slot]

Références donneesRecues().

Référencé par connecterClient().

```
{
   QByteArray message;
   // On détermine quel client envoie le message (recherche du QTcpSocket du
       client)
   QTcpSocket *client = qobject_cast<QTcpSocket *>(sender());

   if (client == 0) // Si par hasard on n'a pas trouvé le client à l'origine du
       signal, on arrête la méthode
           return;

   if (client->bytesAvailable() < 0)
      return;

   message = client->readAll();

   // TODO : extraire le numéro de SIV du message pour conserver la socket dans
       listeSIV
   //QString numSIV;
   //listeSIV[numSIV] = client;

   //qDebug() << QString::fromUtf8("Des données ont été reçues en provenance du
       SIV");
   //qDebug() << QString::fromUtf8("Octets reçus : ") << message.size();
   qDebug() << QString::fromUtf8("Message reçu du SIV : ") << message;

   // signaler à l'IHM que des données ont été reçues et les lui donner
   QString messageSIV(message); // QByteArray -> QString
   emit donneesRecues(messageSIV);

   //envoyer(client); // un acquittement
}
```

9.4.3 Documentation des données membres

9.4.3.1 QList<QTcpSocket *> Serveur : :clients [private]

Liste des objets QTcpServer

Référencé par connecterClient(), et deconnecterClient().

9.4.3.2 QTcpServer* Serveur : :serveur [private]

objet serveur de type QTcpServer

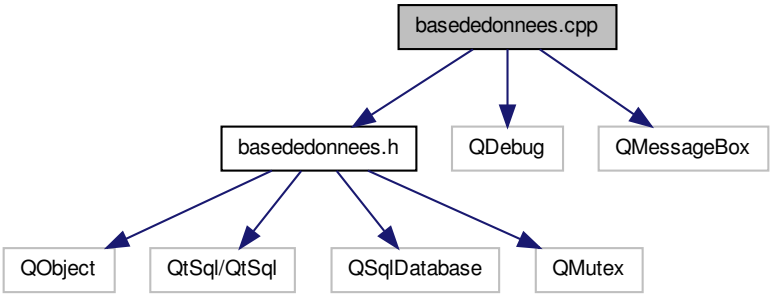
Référencé par connecterClient(), demarrer(), et Serveur().

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

10 Documentation des fichiers

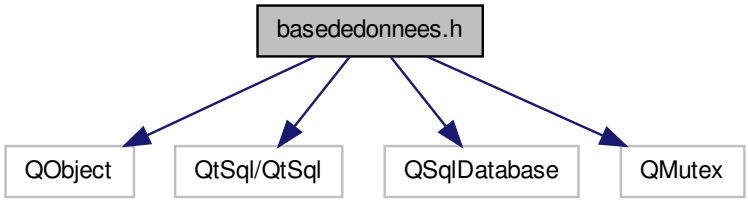
10.1 Référence du fichier basededonnees.cpp

`#include "basededonnees.h"` `#include <QDebug>` `#include <QMessageBox>` Graphe des dépendances par inclusion de basededonnees.cpp :

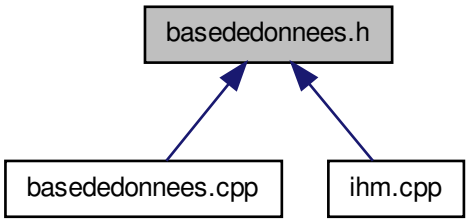


10.2 Référence du fichier basededonnees.h

`#include <QObject>` `#include <QtSql/QtSql>` `#include <QSqlDatabase>` `#include <QMutex>` × Graphe des dépendances par inclusion de basededonnees.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

– class BaseDeDonnees
Classe singleton + ressource critique protégée par un mutex.

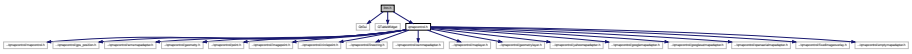
10.3 Référence du fichier Changelog.dox

10.4 Référence du fichier ihm.cpp

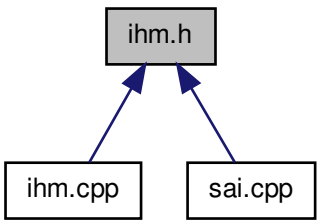
`#include "ihm.h" #include "basededonnees.h" #include "serveur.h" #include <unistd.h>` × `#include <time.h>` Graphe des dépendances par inclusion de ihm.cpp :

10.5 Référence du fichier ihm.h

`#include <QtGui> #include <QTableWidget> #include "qmapcontrol.h"` Graphe des dépendances par inclusion de ihm.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- struct ParametresCarte
  Pour les configurations de la carte.
- class IHM
  Class.

Macros

- #define DEBUG_IHM

Énumérations

- enum ColonneInformationsFlotte { COLONNE_ETAT = 0, COLONNE_NUMERO_BUS, COLONNE_NUMERO_LIGNE, COLONNE_NOM_CONDUCTEUR, COLONNE_DESTINATION, COLONNE_NOM_ARRET, COLONNE_PROCHAIN_ARRET, COLONNE_AVANCE_RETARD, COLONNE_VITESSE, COLONNE_HORODATAGE, COLONNE_NUMERO_SERVICE, NB_COLONNES }
  pour le QTableWidget informationsFlotte
- enum ChampService { SERVICE_ID_SERVICE, SERVICE_CODE_CONDUCTEUR, SERVICE_ID_VEHICULE, SERVICE_NOM_CONDUCTEUR, SERVICE_DATE_DEBUT, SERVICE_HEURE_DEBUT, SERVICE_DATE_FIN, SERVICE_HEURE_FIN, SERVICE_EFFECTUE }
  pour le QVector<QStringList> services
- enum Localisation { DATE_DEBUT, HEURE_DEBUT, IDBUS, ID_SERVICE, ID_LIGNE, ID_ITINERAIRE, LATITUDE, LAT_ORI, LONGITUDE, LONG_ORI, DIRECTION, VITESSE }
  pour les trames
- enum ColonneSuivis { COLONNE_SEQUENCE = 0, COLONNE_SERVICE_ARRET, COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE, COLONNE_SERVICE_DATE_DEBUT_REEL, COLONNE_AVANCE, COLONNE_SERVICE_HEURE_ARRIVE_THEORIQUE, COLONNE_SERVICE_HEURE_ARRIVE_REEL, COLONNE_TEMPS_ARRIVE }
  pour le QTableWidget suivisCourse
- enum ChampSuivi { SEQUENCE, SERVICE_ARRET, SERVICE_HEURE_DEBUT_THEORIQUE, SERVICE_HEURE_ARRIVE_THEORIQUE }
  pour le QVector<QStringList> suivi

10.5.1 Documentation des macros

10.5.1.1 #define DEBUG_IHM

10.5.2 Documentation du type de l'énumération

10.5.2.1 enum ChampService

Valeurs énumérées :

SERVICE_ID_SERVICE

SERVICE_CODE_CONDUCTEUR

SERVICE_ID_VEHICULE

SERVICE_NOM_CONDUCTEUR

SERVICE_DATE_DEBUT

SERVICE_HEURE_DEBUT

SERVICE_DATE_FIN

SERVICE_HEURE_FIN

SERVICE_EFFECTUE

```
{ SERVICE_ID_SERVICE, SERVICE_CODE_CONDUCTEUR, SERVICE_ID_VEHICULE,
      SERVICE_NOM_CONDUCTEUR , SERVICE_DATE_DEBUT, SERVICE_HEURE_DEBUT,
      SERVICE_DATE_FIN, SERVICE_HEURE_FIN, SERVICE_EFFECTUE} ChampService;
```

10.5.2.2 enum ChampSuivi

Valeurs énumérées :

SEQUENCE

SERVICE_ARRET

SERVICE_HEURE_DEBUT_THEORIQUE

SERVICE_HEURE_ARRIVE_THEORIQUE

```
{ SEQUENCE, SERVICE_ARRET, SERVICE_HEURE_DEBUT_THEORIQUE,
      SERVICE_HEURE_ARRIVE_THEORIQUE } ChampSuivi;
```

10.5.2.3 enum ColonneInformationsFlotte

Valeurs énumérées :

COLONNE_ETAT

COLONNE_NUMERO_BUS

COLONNE_NUMERO_LIGNE

COLONNE_NOM_CONDUCTEUR

COLONNE_DESTINATION

COLONNE_NOM_ARRET

COLONNE_PROCHAIN_ARRET

COLONNE_AVANCE_RETARD

COLONNE_VITESSE

COLONNE_HORODATAGE

COLONNE_NUMERO_SERVICE

NB_COLONNES

```
{ COLONNE_ETAT = 0, COLONNE_NUMERO_BUS, COLONNE_NUMERO_LIGNE,
      COLONNE_NOM_CONDUCTEUR, COLONNE_DESTINATION, COLONNE_NOM_ARRET,
      COLONNE_PROCHAIN_ARRET, COLONNE_AVANCE_RETARD, COLONNE_VITESSE,
      COLONNE_HORODATAGE, COLONNE_NUMERO_SERVICE, NB_COLONNES }
      ColonneInformationsFlotte;
```

10.5.2.4 enum ColonneSuivis

Valeurs énumérées :

COLONNE_SEQUENCE

COLONNE_SERVICE_ARRET

COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE

COLONNE_SERVICE_DATE_DEBUT_REEL

COLONNE_AVANCE

COLONNE_SERVICE_HEURE_ARRIVE_THEORIQUE

COLONNE_SERVICE_HEURE_ARRIVE_REEL

COLONNE_TEMPS_ARRIVE

```
{ COLONNE_SEQUENCE = 0, COLONNE_SERVICE_ARRET,
      COLONNE_SERVICE_HEURE_DEBUT_THEORIQUE, COLONNE_SERVICE_DATE_DEBUT_REEL,
      COLONNE_AVANCE, COLONNE_SERVICE_HEURE_ARRIVE_THEORIQUE,
      COLONNE_SERVICE_HEURE_ARRIVE_REEL,COLONNE_TEMPS_ARRIVE} ColonneSuivis;
```

10.5.2.5 enum Localisation

Valeurs énumérées :

DATE_DEBUT

HEURE_DEBUT

IDBUS

ID_SERVICE

ID_LIGNE

ID_ITINERAIRE

LATITUDE

LAT_ORI

LONGITUDE

LONG_ORI

DIRECTION

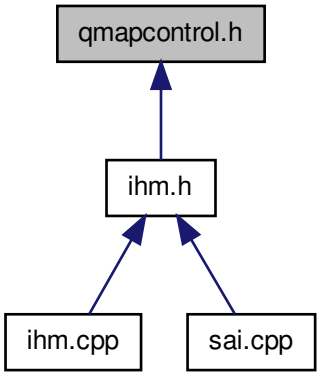
VITESSE

```
{DATE_DEBUT, HEURE_DEBUT, IDBUS, ID_SERVICE, ID_LIGNE, ID_ITINERAIRE, LATITUDE,
       LAT_ORI, LONGITUDE, LONG_ORI, DIRECTION, VITESSE} Localisation;
```

10.6 Référence du fichier qmapcontrol.h

```
#include "../qmapcontrol/mapcontrol.h" #include "../qmapcontrol/gps_position.h" #include
"../qmapcontrol/wmsmapadapter.h" #include "../qmapcontrol/geometry.h" #include "../qmapcontrol
h"   #include "../qmapcontrol/imagepoint.h"   #include "../qmapcontrol/circlepoint.h" ×
#include "../qmapcontrol/linestring.h"     #include "../qmapcontrol/osmmapadapter.h" ×
#include "../qmapcontrol/maplayer.h" #include "../qmapcontrol/geometrylayer.h" #include
"../qmapcontrol/yahoomapadapter.h" #include "../qmapcontrol/googlemapadapter.h" #include
"../qmapcontrol/googlesatmapadapter.h" #include "../qmapcontrol/openaerialmapadapter.-
h" #include "../qmapcontrol/fixedimageoverlay.h" #include "../qmapcontrol/emptymapadapter.-
h" Graphe des dépendances par inclusion de qmapcontrol.h :
```

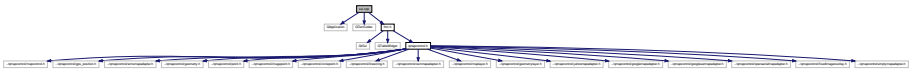
Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



10.7 Référence du fichier README.dox

10.8 Référence du fichier sai.cpp

`#include <QApplication>` `#include <QTextCodec>` `#include "ihm.h"` Graphe des dépendances par inclusion de sai.cpp :



Fonctions

– int main (int argc, char **argv)

10.8.1 Documentation des fonctions

10.8.1.1 main (int *argc*, char ** *argv*)

Paramètres

argc	
argv	

Renvoie

int

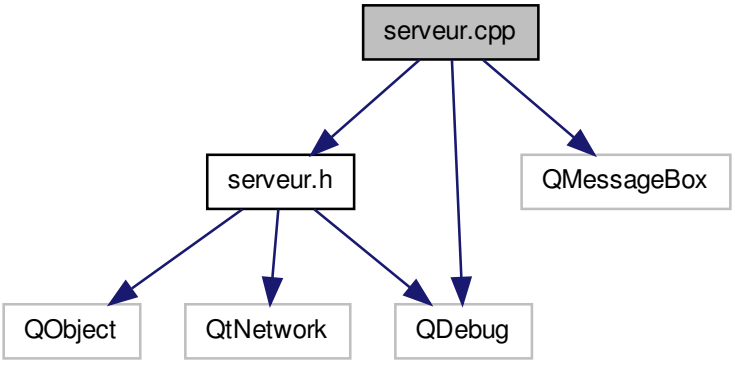
```
{
    QApplication a( argc, argv );
    QTextCodec::setCodecForCStrings(QTextCodec::codecForName("UTF-8"));

    IHM ihm;
    //ihm.adjustSize();
    ihm.show();

    return a.exec();
}
```

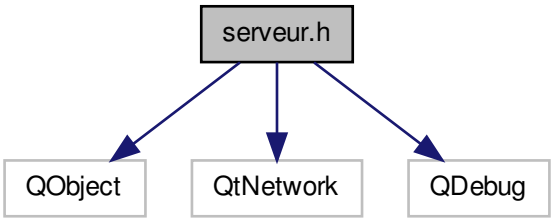
10.9 Référence du fichier serveur.cpp

`#include "serveur.h"` `#include <QDebug>` `#include <QMessageBox>` Graphe des dépendances par inclusion de serveur.cpp :

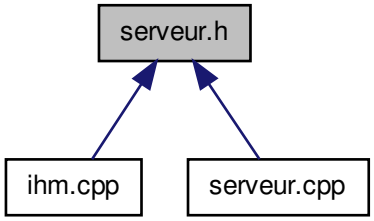


10.10 Référence du fichier serveur.h

`#include <QObject>` `#include <QtNetwork>` `#include <QDebug>` Graphe des dépendances par inclusion de serveur.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

– class Serveur
La classe qui gere la connexion entre SIV et SAI.

Macros

– #define PORT_SERVEUR 5000

10.10.1 Documentation des macros

10.10.1.1 #define PORT_SERVEUR 5000

Référencé par Serveur : :demarrer().

Index