Cédric Thédrez
Formation DL promotion 2015
Afpa Langueux

Projet fil rouge application JAVA et base de données

DAHOUET

Secteur Tertiaire Informatique
Filière étude – développement

Développeur Logiciel
Activité « Développer une application Client/Serveur »

Première Partie

PRESENTATION DE L'APPLICATION

L'application Daouet est une application de gestion de régates. Elle permet de gérer :

Les personnes impliquées dans le déroulement des courses.
Les navires participant aux épreuves ainsi que leurs propriétaires.
Les personnes participant aux épreuves.
Le déroulement des régates et des résultats de celle-ci.

CONCEVOIR ET METTRE EN PLACE UNE BASE DE DONNEES.

1. Construire le schéma entité/association.
1.1 A partir du cahier des charges, élaborer le dictionnaire des données.

-La base de données a été conçue sur le modèle d'un cahier des charges et de ses annexes (dossier de présentation) à partir des quels a été construit le dictionnaire de données (Annexe 1).

1.2 A partir du cahier des charges et des éléments précédemment répertoriés, réaliser le modèle entité/association en respectant le formalisme imposé. Le modèle doit être réalisé avec un outil de conception.

-Un Modèle Conceptuel de Données (MCD) au format Merise a été créé sur la base du dictionnaire de données et des contraintes du cahier des charges (Annexe 2).

1.3 Réaliser un document reprenant l'intégralité des éléments répertoriés avec l'outil de conception mis à votre disposition.

Ce document regroupe l'ensemble des éléments répertoriés et ceux à venir, en annexes, a l'exception des sources intégrales qui sont hébergées sur https://github.com/K4M1coder/DAHOUET
Les fichiers mis en annexes 1 et 2 ont été produits avec le logiciel powerAMC.

2. Construire le schéma physique

- **Construire le schéma physique de données optimisé à partir du model précédent.**

-Un Modèle Physique de Données (MPD) a été généré à partir du MCD et a ensuite été modifié afin de garantir la cohérence des données (Annexe 3).

3. Ecrire les scripts SQL de définition de données et des contraintes.

- **Créer et exécuter le script de génération de la base réalisé à partir du schéma physique obtenu à l'étape précédente.**
- **Intégrer la définition des contraintes sur les données.**

En cas d'anomalie, corriger la source de l'erreur au niveau adéquat.

Un script SQL est généré par PowerAMC, (base_Dahouet.sql) celui-ci est verifié, corrigé, et nommé Dahouet_base.SQL).

4. Administrer la base de données de test.

- **Créer un utilisateur de la base de données ayant uniquement des droits de lecture seule aux tables concernant les résultats des régates.**

L'utilisateur 'dahouet_readuser' a des droits (SELECT) uniquement sur les tables participation marin régates et voiliers de la base dahouet.

- **Décrire les procédures pour assurer les sauvegardes de la base Dahouet. Tester la restauration.**

Pour la sauvegarde et restauration de la base on peut utiliser l'outil Heidi SQL.
La procédure de sauvegarde est la suivante :
Dans le menu de droite, clic droit sur la base dahouet -> Exporter base de données en SQL
Pour sauvegarder la structure de la base seule : (Dahouet_base.sql)
Cocher supprimer et créer pour la base de données et les tables.
Selectionner No Data pour Données.
Pour sauvegarder les données de la base seules : (Dahouet_data.sql)
Décocher supprimer et créer pour la base de données et les tables.
Selectionner REPLACE existing data pour Données.
Pour sauvegarder la structure et les données de la base : (Dahouet_dumpdb.sql)
Cocher supprimer et créer pour la base de données et les tables.
Selectionner REPLACE existing data pour Données.
Dans tous les cas :
Choisir le fichier dans lequel sera enregistré l'export
Cliquer sur Export

Les fichiers générés ont étés stockés dans le dossier bdd MySQL du repo GITHUB

5. Manipuler les données avec SQL

Pour chacune des interrogations demandées, créer un script contenant la ou les requêtes nécessaires. Voir expression des besoins (§ 4.2).

Les requêtes sont stockées sur GITHUB et normalisées 4.2_rqX_NOM.sql
X vaut le numéro de la requête.
NOM vaux le nom de la requête.

Requête 1
On souhaite obtenir par challenge la moyenne des distances courues des différentes régates.
Requête 2
On souhaite obtenir la liste des participants (nom et numéro de licence) à une régate sélectionnée par son numéro. Préciser le libellé et la date de la régate ainsi que le nom des voiliers et les numéros des skippers associés.
Requête 3
On souhaite obtenir à la date du jour la liste des régates (libellé et date) non courues, avec pour chacune la liste des commissaires associés. Préciser le nom et le comité de ces commissaires.

Dans la base de données, des requêtes, ont étés crées suivant l'énoncé fournit. (Annexe 5)

6. Programmer dans le langage du SGBD

- **Créer des fonctions**

Créer une fonction qui renvoie un nouveau code régate (identifiant), composé du code challenge + le mois + un numéro séquentiel dans le challenge.

- **Programmer des déclencheurs (triggers)**

Mettre en place les déclencheurs de création, mise à jour et suppression proposés dans le cahier des charges. Expression des besoins (§ 4.3).

Les requêtes sont stockées sur GITHUB et normalisées 4.3_trX_NOM.sql
X vaut le numéro de la requête.
NOM vaux le nom de la requête.

Triggers de création
Table Régate
Vérifier que la date de la régate est comprise dans les dates du challenge concerné.
Triggers de mise à jour.
Table Participe (résultats des voiliers aux régates)
Vérifier que la place attribuée à un voilier à l'issue d'une régate, n'est pas supérieure au nombre de participants.
Triggers de suppression.
Table Régate
Ne pas supprimer une régate si le challenge auquel elle est associée n'est pas terminé.

Dans la base de données, des triggers ont étés crées suivant l'énoncé fournit. (Annexe 6).

- **Programmer des procédures stockées sur le SGBD**

Créer les procédures stockées proposées dans le cahier des charges. Expression des besoins (§ 4.4).

Les requêtes sont stockées sur GITHUB et normalisées 4.2_prX_NOM.sql
X vaut le numéro de la requête.
NOM vaux le nom de la requête.

Procédure 1 :
On souhaite obtenir la moyenne des distances des régates pour un challenge (hiver ou été) donné
Procédure 2 :
On souhaite obtenir la liste de l'équipage d'un voilier pour une régate.
Procédure 3 :
On veut lister les interventions des commissaires sur un challenge entre deux dates.
La procédure renvoie pour chaque intervention :
- Le nom du commissaire associé
- Le comité du commissaire
- La date de la régate concernée

Dans la base de données, des procédures stockées ont étés crées suivant l'énoncé fourni. (Annexe 7)

DEVELOPPER L'INTERFACE D'UNE APPLICATION INFORMATIQUE.

Le code source de l'application est hébergé sur https://github.com/K4M1coder/DAHOUET

1. **Ecrire un algorithme.**

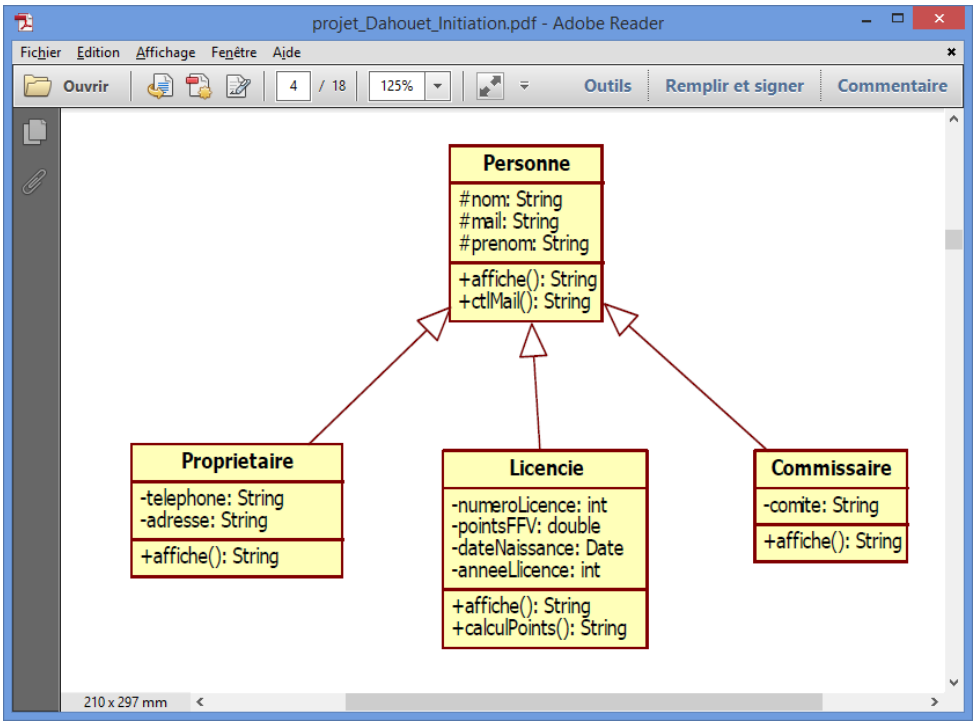
Coder un composant qui vérifie qu'une adresse mail fournie sous forme de chaîne de caractères est cohérente.

En entrée, ce composant reçoit l'adresse à vérifier.
On considère qu'une adresse email est correcte si elle contient @, et au moins un point dans la partie qui suit @. Le partie devant @ doit faire au moins deux caractères, entre @ et le point au moins deux caractères et après le point au moins deux caractères.
Il renvoie une chaîne de caractère à blanc si l'adresse est correcte, ou contenant le message de l'erreur détectée si l'adresse n'est pas cohérente.
Prévoir le jeu de test de cette fonction avec l'utilitaire de test unitaire.
Note : Vous pouvez utiliser les expressions régulières pour programmer les contrôles.

```
public static String ctMail(String email) {
        String result = "mail conforme";
        if (email.matches("^[a-zA-Z0-9._-]{2,}+@[a-zA-Z0-9._-]{2,}\\.[a-z]{2,4}$"));
        else
        { result="Adresse mail invalide"; }
        return result;
}
```

2. Développement objet.

Développer les composants métier et le jeu de test associé de la partie gérer les personnes du diagramme de classes suivant :



-Création des classes métiers à parti du modèle UML fourni et des classes de contrôle et de calcul demandé par l'énoncé. (Annexe 8)

- Pour le service « affiche() » des différentes classes, renvoyer sous forme d'une chaîne de caractères les informations concernant chaque objet. Pour le licencié calculer et renvoyer son âge avec les autres informations.

Voire les sources sur GITHUB

- Utiliser la fonction de contrôle du mail « ctlMail » développée précédemment pour l'introduire dans la classe Personne.

Voire les sources sur GITHUB

- Le service "calculPoints()" de la classe Licencié prend en entrée un nombre de points à additionner à la propriété pointsFFV, et une date. Pour mettre à jour le nombre de point il faut que la date en entrée corresponde à l'année de la date de la licence sinon la mise à jour est refusée avec en retour un message de non prise en compte

Voire les sources sur GITHUB

3. Développer l'interface graphique client/serveur.

- **Cas d'utilisation : Enregistrement d'un voilier (cf § 4.1) :**
  - **Concevoir la maquette de ce cas en mode stand-alone.**
  - **Développer le composant logiciel, ce composant est à réaliser après avoir créé la base de données du projet.**

L'application est construite en suivant une logique MVC

- Toute la logique applicative se trouve dans le contrôleur
- Toute la logique d'affichage se trouve dans les vues qui sont lancées par le contrôleur
- Toute le logique métier et les entités se trouvent dans les Model

- Toutes les interactions avec la base de données sont gérées par les classes DAO qui sont toujours appelées par un contrôleur

-Création des interfaces grâce à Java Swing (Annexe 9)

-Création des Classes de dialogue avec la base de données (DAO) permettant d'obtenir les informations nécessaires pour les formulaires et d'enregistrer de nouvelles données. (Annexe 10)

Le code source intégral de l'application est hébergé sur https://github.com/K4M1coder/DAHOUET

To-Do liste

Au-delà de la strique réponse au besoin formulé par le cahier des charges, l'application est sujette à quelques évolutions :

- Prise en charge complète des fonctions de modification et de suppression pour un voilier
- Implémentation de la fonction de création de nouvelles personnes
- Prise en charge complète des fonctions de modification suppression pour un propriétaire, incluant la même chose pour les personnes dont ils dépendent (au sens SQL).
- Création d'une boite de dialogue (Jdialog) A propos
- Prise en charge des raccourcis claviers pour la barre de menu haute
- Affichage d'information sur l'objet en cours de survol ou l'action encours dans la barre de statuts.
- Affichage de la progression des actions en cours dans une barre de progression.
- Vérification de la validité de toutes les informations saisies.
- Livrer l'application au format .jar

Pour que l'application soit complète, il faudrait qu'elle prenne en charges tous les aspects métiers non limités aux exemples suivants

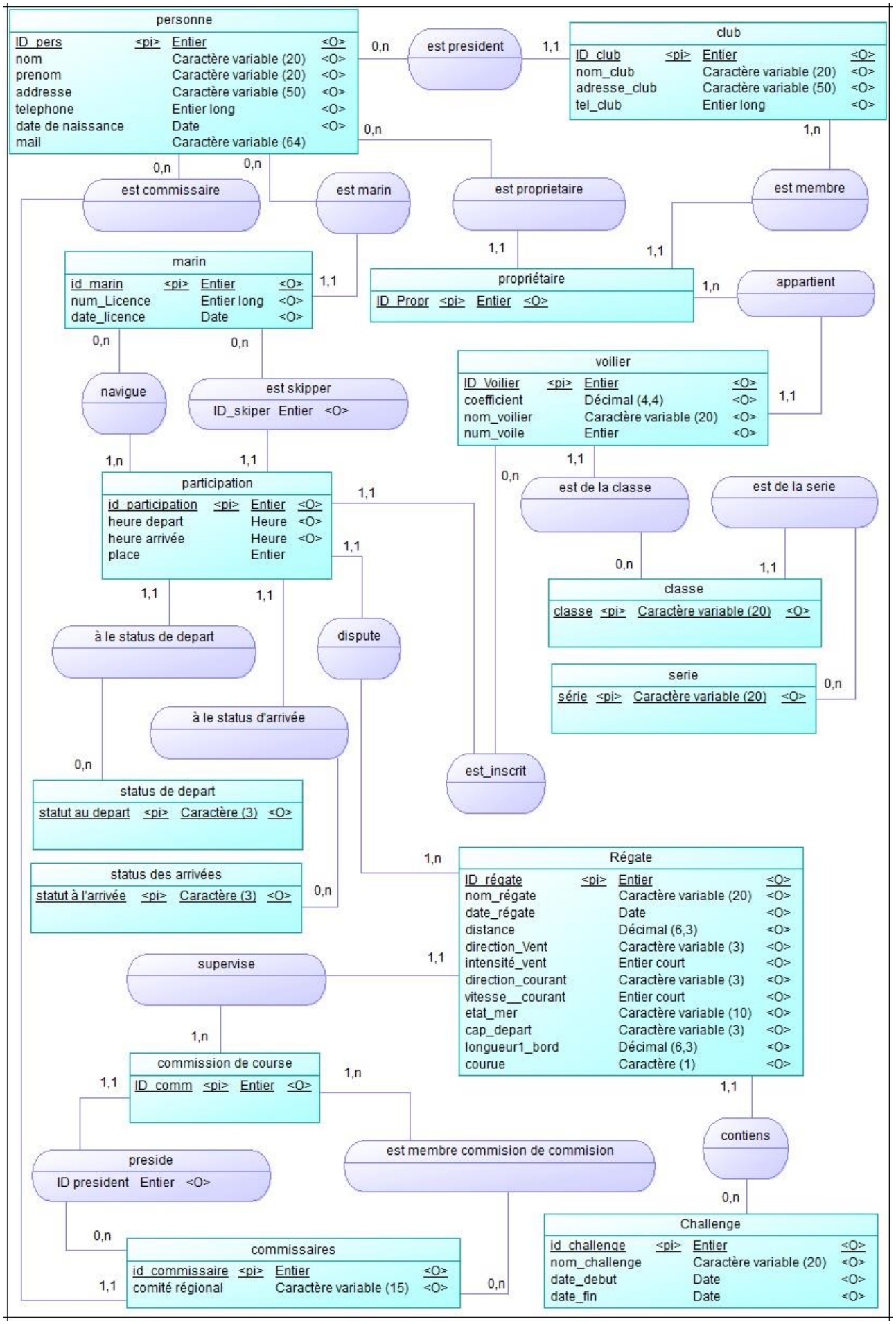
- Gestion des marins
- Gestion des clubs
- Gestion des commissions de supervision
- Inscription des marins
- Inscription des voiliers
- Gestion des commissaires de course
- Gestion de régates
- résultats
- … …

ANNEXES

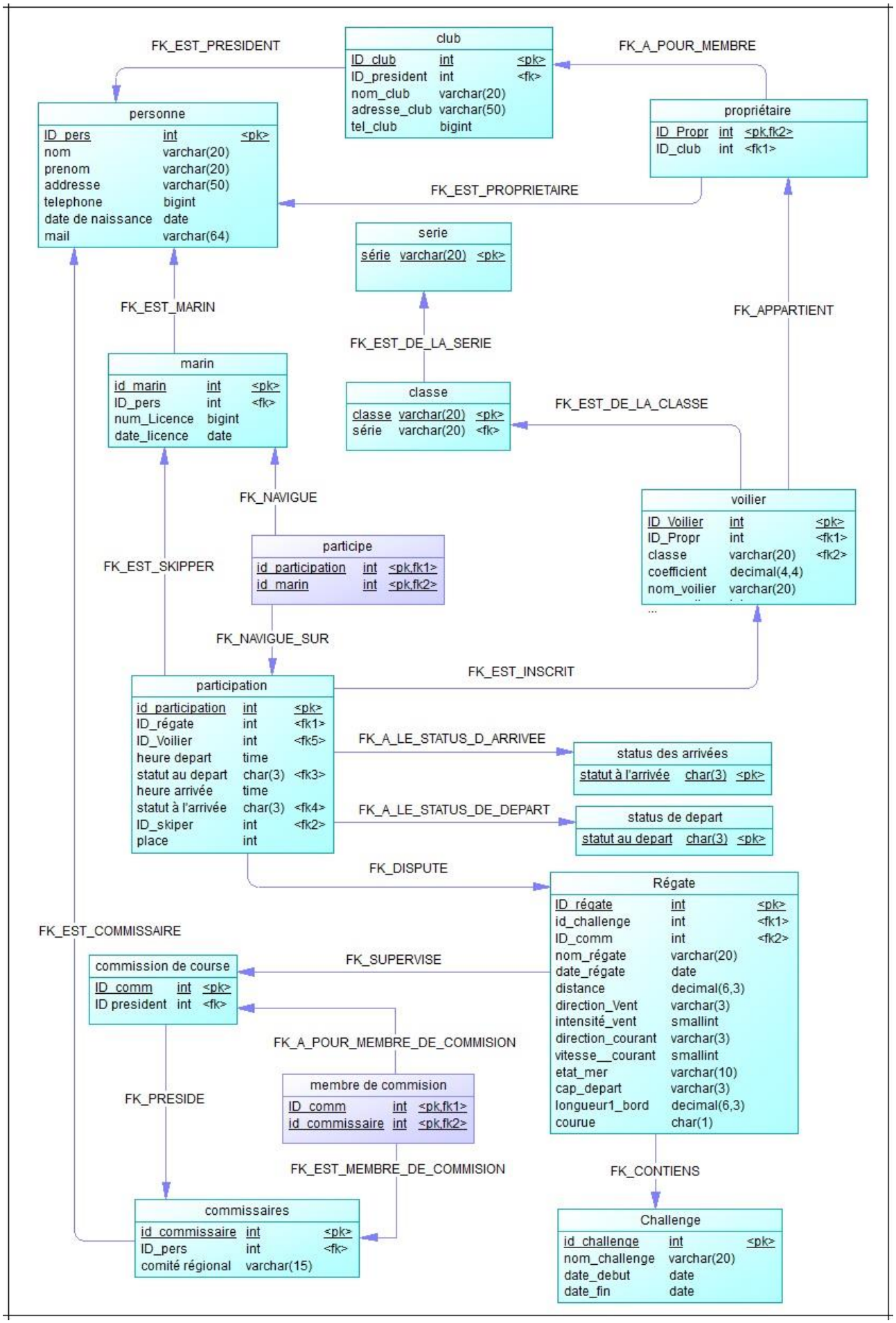
Annexe 1 : dictionnaire de données

Nom	Code	Table	Type de données	Longueur	Précision	Primaire	Clé étrangère	Obligatoire
addresse	ADDRESSE	personne	varchar(50)	50		FALSE	FALSE	TRUE
adresse_club	ADRESSE_CLUB	club	varchar(50)	50		FALSE	FALSE	TRUE
cap_depart	CAP	Régate	varchar(3)	3		FALSE	FALSE	TRUE
classe	CLASSE	classe	varchar(20)	20		TRUE	FALSE	TRUE
classe	CLASSE	voilier	varchar(20)	20		FALSE	TRUE	TRUE
coefficient	COEFF	voilier	decimal(4,4)	4	4	FALSE	FALSE	TRUE
comité régional	COM_REG	commissaires	varchar(15)	15		FALSE	FALSE	TRUE
courue	COURUE	Régate	char(1)	1		FALSE	FALSE	TRUE
date de naissance	DATE_N	personne	date			FALSE	FALSE	TRUE
date_debut	DATE_DEBUT	Challenge	date			FALSE	FALSE	TRUE
date_fin	DATE_FIN	Challenge	date			FALSE	FALSE	TRUE
date_licence	DATE_LICENCE	marin	date			FALSE	FALSE	TRUE
date_régate	DATE_REGATE	Régate	date			FALSE	FALSE	TRUE
direction_courant	D_COURANT	Régate	varchar(3)	3		FALSE	FALSE	TRUE
direction_Vent	D_VENT	Régate	varchar(3)	3		FALSE	FALSE	TRUE
distance	DISTANCE	Régate	decimal(6,3)	6	3	FALSE	FALSE	TRUE
etat_mer	ETAT_MER	Régate	varchar(10)	10		FALSE	FALSE	TRUE
heure arrivée	H_ARRIV	participation	time			FALSE	FALSE	TRUE
heure depart	H_DEP	participation	time			FALSE	FALSE	TRUE
ID president	ID_PRESIDENT	commission de course	int			FALSE	TRUE	TRUE
id_challenge	ID_CHALL	Challenge	int			TRUE	FALSE	TRUE
id_challenge	ID_CHALL	Régate	int			FALSE	TRUE	TRUE
ID_club	ID_CLUB	club	int			TRUE	FALSE	TRUE
ID_club	ID_CLUB	propriétaire	int			FALSE	TRUE	TRUE
ID_comm	ID_COM	Régate	int			FALSE	TRUE	TRUE
ID_comm	ID_COM	commission de course	int			TRUE	FALSE	TRUE
ID_comm	ID_COM	membre de commision	int			TRUE	TRUE	TRUE
id_commissaire	ID_COMMISSAIRE	commissaires	int			TRUE	FALSE	TRUE
id_commissaire	ID_COMMISSAIRE	membre de commision	int			TRUE	TRUE	TRUE
id_marin	ID_MARIN	marin	int			TRUE	FALSE	TRUE
id_marin	ID_MARIN	participe	int			TRUE	TRUE	TRUE
id_participation	ID_PART	participation	int			TRUE	FALSE	TRUE
id_participation	ID_PART	participe	int			TRUE	TRUE	TRUE
ID_pers	ID_PERS	commissaires	int			FALSE	TRUE	TRUE
ID_pers	ID_PERS	personne	int			TRUE	FALSE	TRUE
ID_pers	ID_PERS	marin	int			FALSE	TRUE	TRUE
ID_president	ID_PRESIDENT	club	int			FALSE	TRUE	TRUE
ID_Propr	ID_PROPR	propriétaire	int			TRUE	TRUE	TRUE
ID_Propr	ID_PROPR	voilier	int			FALSE	TRUE	TRUE
ID_régate	ID_REGATE	participation	int			FALSE	TRUE	TRUE
ID_régate	ID_REGATE	Régate	int			TRUE	FALSE	TRUE
ID_skiper	ID_SKIPER	participation	int			FALSE	TRUE	TRUE
ID_Voilier	ID_VOILIER	participation	int			FALSE	TRUE	TRUE
ID_Voilier	ID_VOILIER	voilier	int			TRUE	FALSE	TRUE
intensité_vent	I_VENT	Régate	smallint			FALSE	FALSE	TRUE
longueur1_bord	L_BORD	Régate	decimal(6,3)	6	3	FALSE	FALSE	TRUE
mail	MAIL	personne	varchar(64)	64		FALSE	FALSE	FALSE
nom	NOM	personne	varchar(20)	20		FALSE	FALSE	TRUE
nom_challenge	NOM_CHALL	Challenge	varchar(20)	20		FALSE	FALSE	TRUE
nom_club	NOM_CLUB	club	varchar(20)	20		FALSE	FALSE	TRUE
nom_régate	NOM_REG	Régate	varchar(20)	20		FALSE	FALSE	TRUE
nom_voilier	NOM_VOILIER	voilier	varchar(20)	20		FALSE	FALSE	TRUE
num_Licence	NUM_LICENCE	marin	bigint			FALSE	FALSE	TRUE
num_voile	NUM_VOILE	voilier	int			FALSE	FALSE	TRUE
place	PLACE	participation	int			FALSE	FALSE	FALSE
prenom	PRENOM	personne	varchar(20)	20		FALSE	FALSE	TRUE
statut au depart	STATUT_DEP	status de depart	char(3)	3		TRUE	FALSE	TRUE
statut au depart	STATUT_DEP	participation	char(3)	3		FALSE	TRUE	TRUE
statut à l'arrivée	STATUT_ARRIV	status des arrivées	char(3)	3		TRUE	FALSE	TRUE
statut à l'arrivée	STATUT_ARRIV	participation	char(3)	3		FALSE	TRUE	TRUE
série	SERIE	serie	varchar(20)	20		TRUE	FALSE	TRUE
série	SERIE	classe	varchar(20)	20		FALSE	TRUE	TRUE
tel_club	TEL_CLUB	club	bigint			FALSE	FALSE	TRUE
telephone	TELEPHONE	personne	bigint			FALSE	FALSE	TRUE
vitesse__courant	V_COURANT	Régate	smallint			FALSE	FALSE	TRUE

Annexe 2 : Modèle Conceptuel de Données



Annexe 3 : Modèle Physique de données



Annexe 4 : Script SQL généré

```
/*==============================================================*/
/* Nom de SGBD :  MySQL 5.0                                     */
/* Date de creation :  06/07/2015 11:51:20                      */
/*==============================================================*/


DROP TABLE IF EXISTS CHALLENGE;

DROP TABLE IF EXISTS CLASSE;

DROP TABLE IF EXISTS CLUB;

DROP TABLE IF EXISTS COMMISSAIRES;

DROP TABLE IF EXISTS COMMISSION_DE_COURSE;

DROP TABLE IF EXISTS MARIN;

DROP TABLE IF EXISTS MEMBRE_DE_COMMISION;

DROP TABLE IF EXISTS PARTICIPATION;

DROP TABLE IF EXISTS PARTICIPE;

DROP TABLE IF EXISTS PERSONNE;

DROP TABLE IF EXISTS PROPRIETAIRE;

DROP TABLE IF EXISTS REGATE;

DROP TABLE IF EXISTS SERIE;

DROP TABLE IF EXISTS STATUS_DES_ARRIVEES;

DROP TABLE IF EXISTS STATUS_DE_DEPART;

DROP TABLE IF EXISTS VOILIER;

/*==============================================================*/
/* Table : CHALLENGE                                            */
/*==============================================================*/
CREATE TABLE CHALLENGE
(
  ID_CHALL          INT NOT NULL AUTO_INCREMENT,
  NOM_CHALL         VARCHAR(20) NOT NULL,
  DATE_DEBUT        DATE NOT NULL,
  DATE_FIN          DATE NOT NULL,
  PRIMARY KEY (ID_CHALL)
);

/*==============================================================*/
/* Table : CLASSE                                               */
/*==============================================================*/
CREATE TABLE CLASSE
(
  CLASSE            VARCHAR(20) NOT NULL,
  SERIE             VARCHAR(20) NOT NULL,
  PRIMARY KEY (CLASSE)
);

/*==============================================================*/
/* Table : CLUB                                                 */
/*==============================================================*/
CREATE TABLE CLUB
(
  ID_CLUB           INT NOT NULL AUTO_INCREMENT,
  ID_PRESIDENT      INT NOT NULL,
  NOM_CLUB          VARCHAR(20) NOT NULL,
  ADRESSE_CLUB      VARCHAR(50) NOT NULL,
  TEL_CLUB          BIGINT NOT NULL,
  PRIMARY KEY (ID_CLUB)
```

```
);

/*==============================================================*/
/* Table : COMMISSAIRES                                         */
/*==============================================================*/
CREATE TABLE COMMISSAIRES
(
  ID_COMMISSAIRE     INT NOT NULL AUTO_INCREMENT,
  ID_PERS            INT NOT NULL,
  COM_REG            VARCHAR(15) NOT NULL,
  PRIMARY KEY (ID_COMMISSAIRE)
);

/*==============================================================*/
/* Table : COMMISSION_DE_COURSE                                 */
/*==============================================================*/
CREATE TABLE COMMISSION_DE_COURSE
(
  ID_COM             INT NOT NULL AUTO_INCREMENT,
  ID_PRESIDENT       INT NOT NULL,
  PRIMARY KEY (ID_COM)
);

/*==============================================================*/
/* Table : MARIN                                                */
/*==============================================================*/
CREATE TABLE MARIN
(
  ID_MARIN           INT NOT NULL AUTO_INCREMENT,
  ID_PERS            INT NOT NULL,
  NUM_LICENCE        BIGINT NOT NULL,
  DATE_LICENCE       DATE NOT NULL,
  PRIMARY KEY (ID_MARIN)
);

/*==============================================================*/
/* Table : MEMBRE_DE_COMMISION                                  */
/*==============================================================*/
CREATE TABLE MEMBRE_DE_COMMISION
(
  ID_COM             INT NOT NULL,
  ID_COMMISSAIRE     INT NOT NULL,
  PRIMARY KEY (ID_COM, ID_COMMISSAIRE)
);

/*==============================================================*/
/* Table : PARTICIPATION                                        */
/*==============================================================*/
CREATE TABLE PARTICIPATION
(
  ID_PART            INT NOT NULL,
  ID_REGATE          INT NOT NULL,
  ID_VOILIER         INT NOT NULL,
  H_DEP              TIME NOT NULL,
  STATUT_DEP         CHAR(3) NOT NULL,
  H_ARRIV            TIME NOT NULL,
  STATUT_ARRIV       CHAR(3) NOT NULL,
  ID_SKIPER          INT NOT NULL,
  PLACE              INT,
  PRIMARY KEY (ID_PART)
);

/*==============================================================*/
/* Table : PARTICIPE                                            */
/*==============================================================*/
CREATE TABLE PARTICIPE
(
  ID_PART            INT NOT NULL,
  ID_MARIN           INT NOT NULL,
  PRIMARY KEY (ID_PART, ID_MARIN)
);

/*==============================================================*/
/* Table : PERSONNE                                             */
```

```
/*==============================================================*/
CREATE TABLE PERSONNE
(
  ID_PERS          INT NOT NULL AUTO_INCREMENT,
  NOM              VARCHAR(20) NOT NULL,
  PRENOM           VARCHAR(20) NOT NULL,
  ADDRESSE         VARCHAR(50) NOT NULL,
  TELEPHONE        BIGINT NOT NULL,
  DATE_N           DATE NOT NULL,
  MAIL             VARCHAR(64),
  PRIMARY KEY (ID_PERS)
);

/*==============================================================*/
/* TABLE : PROPRIETAIRE                                         */
/*==============================================================*/
CREATE TABLE PROPRIETAIRE
(
  ID_PROPR         INT NOT NULL,
  ID_CLUB          INT NOT NULL,
  PRIMARY KEY (ID_PROPR)
);

/*==============================================================*/
/* TABLE : REGATE                                               */
/*==============================================================*/
CREATE TABLE REGATE
(
  ID_REGATE        INT NOT NULL AUTO_INCREMENT,
  ID_CHALL         INT NOT NULL,
  ID_COM           INT NOT NULL,
  NOM_REG          VARCHAR(20) NOT NULL,
  DATE_REGATE      DATE NOT NULL,
  DISTANCE         DECIMAL(6,3) NOT NULL,
  D_VENT           VARCHAR(3) NOT NULL,
  I_VENT           SMALLINT NOT NULL,
  D_COURANT        VARCHAR(3) NOT NULL,
  V_COURANT        SMALLINT NOT NULL,
  ETAT_MER         VARCHAR(10) NOT NULL,
  CAP              VARCHAR(3) NOT NULL,
  L_BORD           DECIMAL(6,3) NOT NULL,
  COURUE           CHAR(1) NOT NULL,
  PRIMARY KEY (ID_REGATE)
);

/*==============================================================*/
/* TABLE : SERIE                                                */
/*==============================================================*/
CREATE TABLE SERIE
(
  SERIE            VARCHAR(20) NOT NULL,
  PRIMARY KEY (SERIE)
);

/*==============================================================*/
/* TABLE : STATUS_DES_ARRIVEES                                  */
/*==============================================================*/
CREATE TABLE STATUS_DES_ARRIVEES
(
  STATUT_ARRIV     CHAR(3) NOT NULL,
  PRIMARY KEY (STATUT_ARRIV)
);

/*==============================================================*/
/* TABLE : STATUS_DE_DEPART                                     */
/*==============================================================*/
CREATE TABLE STATUS_DE_DEPART
(
  STATUT_DEP       CHAR(3) NOT NULL,
  PRIMARY KEY (STATUT_DEP)
);

/*==============================================================*/
/* TABLE : VOILIER                                              */
```

```
/*==============================================================*/
CREATE TABLE VOILIER
(
  ID_VOILIER        INT NOT NULL AUTO_INCREMENT,
  ID_PROPR          INT NOT NULL,
  CLASSE            VARCHAR(20) NOT NULL,
  COEFF             DECIMAL(4,4) NOT NULL,
  NOM_VOILIER       VARCHAR(20) NOT NULL,
  NUM_VOILE         INT NOT NULL,
  PRIMARY KEY (ID_VOILIER)
);

ALTER TABLE CLASSE ADD CONSTRAINT FK_EST_DE_LA_SERIE FOREIGN KEY (SERIE)
    REFERENCES SERIE (SERIE) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE CLUB ADD CONSTRAINT FK_EST_PRESIDENT FOREIGN KEY (ID_PRESIDENT)
    REFERENCES PERSONNE (ID_PERS) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE COMMISSAIRES ADD CONSTRAINT FK_EST_COMMISSAIRE FOREIGN KEY (ID_PERS)
    REFERENCES PERSONNE (ID_PERS) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE COMMISSION_DE_COURSE ADD CONSTRAINT FK_PRESIDE FOREIGN KEY (ID_PRESIDENT)
    REFERENCES COMMISSAIRES (ID_COMMISSAIRE) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE MARIN ADD CONSTRAINT FK_EST_MARIN FOREIGN KEY (ID_PERS)
    REFERENCES PERSONNE (ID_PERS) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE MEMBRE_DE_COMMISION ADD CONSTRAINT FK_A_POUR_MEMBRE_DE_COMMISION FOREIGN KEY
(ID_COM)
    REFERENCES COMMISSION_DE_COURSE (ID_COM) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE MEMBRE_DE_COMMISION ADD CONSTRAINT FK_EST_MEMBRE_DE_COMMISION FOREIGN KEY
(ID_COMMISSAIRE)
    REFERENCES COMMISSAIRES (ID_COMMISSAIRE) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPATION ADD CONSTRAINT FK_A_LE_STATUS_DE_DEPART FOREIGN KEY (STATUT_DEP)
    REFERENCES STATUS_DE_DEPART (STATUT_DEP) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPATION ADD CONSTRAINT FK_A_LE_STATUS_D_ARRIVEE FOREIGN KEY (STATUT_ARRIV)
    REFERENCES STATUS_DES_ARRIVEES (STATUT_ARRIV) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPATION ADD CONSTRAINT FK_DISPUTE FOREIGN KEY (ID_REGATE)
    REFERENCES REGATE (ID_REGATE) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPATION ADD CONSTRAINT FK_EST_INSCRIT FOREIGN KEY (ID_VOILIER)
    REFERENCES VOILIER (ID_VOILIER) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPATION ADD CONSTRAINT FK_EST_SKIPPER FOREIGN KEY (ID_SKIPER)
    REFERENCES MARIN (ID_MARIN) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPE ADD CONSTRAINT FK_NAVIGUE FOREIGN KEY (ID_MARIN)
    REFERENCES MARIN (ID_MARIN) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PARTICIPE ADD CONSTRAINT FK_NAVIGUE_SUR FOREIGN KEY (ID_PART)
    REFERENCES PARTICIPATION (ID_PART) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PROPRIETAIRE ADD CONSTRAINT FK_A_POUR_MEMBRE FOREIGN KEY (ID_CLUB)
    REFERENCES CLUB (ID_CLUB) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE PROPRIETAIRE ADD CONSTRAINT FK_EST_PROPRIETAIRE FOREIGN KEY (ID_PROPR)
    REFERENCES PERSONNE (ID_PERS) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE REGATE ADD CONSTRAINT FK_CONTIENS FOREIGN KEY (ID_CHALL)
    REFERENCES CHALLENGE (ID_CHALL) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE REGATE ADD CONSTRAINT FK_SUPERVISE FOREIGN KEY (ID_COM)
    REFERENCES COMMISSION_DE_COURSE (ID_COM) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE VOILIER ADD CONSTRAINT FK_APPARTIENT FOREIGN KEY (ID_PROPR)
    REFERENCES PROPRIETAIRE (ID_PROPR) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE VOILIER ADD CONSTRAINT FK_EST_DE_LA_CLASSE FOREIGN KEY (CLASSE)
    REFERENCES CLASSE (CLASSE) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

4.2_rq1_dist_moy.sql

```
select challenge.ID_CHALL,challenge.NOM_CHALL,
challenge.DATE_DEBUT,(sum(regate.DISTANCE)) /
      (count( regate.ID_CHALL)) as distance_moyenne
from challenge
inner join regate on challenge.ID_CHALL=regate.ID_CHALL
where regate.COURUE='y'
group by challenge.ID_CHALL;
```

4.2_rq1.CVS

ID_CHALL	NOM_CHALL	DATE_DEBUT	distance_moyenne
1	été	01/05/2014	11,35

4.2_rq2_particp_regate.sql

```
select personne.ID_PERS, personne.NOM,personne.PRENOM,
marin.NUM_LICENCE,regate.NOM_REG,regate.DATE_REGATE,voilier.NOM_VOILIER,particip
ation.ID_SKIPER
from marin inner join participe on marin.ID_MARIN=participe.ID_MARIN
inner join participation on participe.ID_PART=participation.ID_PART
inner join regate on participation.ID_REGATE = regate.ID_REGATE
inner join voilier on participation.ID_VOILIER = voilier.ID_VOILIER
inner join personne on personne.ID_PERS = marin.ID_PERS
where regate.ID_REGATE = 1
order by marin.ID_MARIN asc
```

4.2_rq2.csv

ID_PERS	NOM	PRENOM	NUM_LICENC	NOM_REG	DATE_REGAT	NOM_VOILIE	ID_SKIPER
4	pirrate	luffy	111111	ST BRIEUC	12/07/2014	SUNNY-GO	1
5	simon	phillipe	123456	ST BRIEUC	12/07/2014	SUNNY-GO	1
6	CHOPPER	sandra	234567	ST BRIEUC	12/07/2014	SUNNY-GO	1
7	blanche	barbe	345678	ST BRIEUC	12/07/2014	MOBY DICK	4
8	ACE	quentin	456789	ST BRIEUC	12/07/2014	MOBY DICK	4
9	MARCO	julio	222222	ST BRIEUC	12/07/2014	MOBY DICK	4
19	GATES	Bill	333333	ST BRIEUC	12/07/2014	MILLENIUM	7
11	XBOX	360	444444	ST BRIEUC	12/07/2014	MILLENIUM	7
12	KINNECT	sensor	555555	ST BRIEUC	12/07/2014	MILLENIUM	7
13	SHANKS	travis	666666	ST BRIEUC	12/07/2014	RED FORCE	10
14	YASSOP	lino	777777	ST BRIEUC	12/07/2014	RED FORCE	10
15	BECKMAN	Ben	888888	ST BRIEUC	12/07/2014	RED FORCE	10
16	JONES	Davy	999999	ST BRIEUC	12/07/2014	FLYING DUTC	13
17	VAN DER DE(	heirmack	987654	ST BRIEUC	12/07/2014	FLYING DUTC	13
18	ross	tim	654321	ST BRIEUC	12/07/2014	FLYING DUTC	13

4.2_rq3_regatesnc_lst_comsr.sql

```
select distinct regate.NOM_REG as regate, regate.DATE_REGATE as date,
personne.NOM as commissaire, commissaires.COM_REG as comité from
regate inner join commission_de_course on regate.ID_COM =
commission_de_course.ID_COM
inner join membre_de_commision on commission_de_course.ID_COM =
membre_de_commision.ID_COM
inner join commissaires on membre_de_commision.ID_COMMISSAIRE =
commissaires.ID_COMMISSAIRE
inner join personne on personne.ID_PERS = commissaires.ID_PERS
inner join participation on regate.ID_REGATE = participation.ID_REGATE
where regate.DATE_REGATE < curdate()
and regate.COURUE = "n"
order by regate.DATE_REGATE asc;
```

4.2_rq3.csv

regate	date	commissaire	comité
ST BRIEUC	14/11/2014	Auron	Spira
ST BRIEUC	14/11/2014	Cid	Avalanche
ST BRIEUC	15/12/2014	Auron	Spira
ST BRIEUC	15/12/2014	Cid	Avalanche

4.3_tr1_creation.sql

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `regate_before_insert` BEFORE INSERT ON `regate` FOR EACH ROW BEGIN
declare datedeb date;
declare datefin date;
declare error condition for sqlstate '45001';

select challenge.DATE_DEBUT into datedeb from challenge where
new.ID_CHALL = challenge.ID_CHALL;
select challenge.DATE_FIN into datefin from challenge where
new.ID_CHALL = challenge.ID_CHALL;
if( new.DATE_REGATE not BETWEEN datedeb and datefin)
then signal error
set message_text =' cette date est en dehors du challenge ... +?+(°o°)+?+ ', mysql_errno=9002;
end if;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

4.3_tr2_maj.sql

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- Export de la structure de déclancheur dahouet. participation_before_update
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `participation_before_update` BEFORE UPDATE ON `participation` FOR EACH ROW BEGIN
declare nbpart int;
declare error condition for sqlstate '45001';


select count(old.ID_VOILIER) from participation
        where new.ID_REGATE=ID_REGATE
into nbpart;
if(new.PLACE>nbpart)
        then signal error
        set message_text =' il n y à pas autant de voiliers', mysql_errno=9002;
end if;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

4.3_tr3_del.sql

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- Export de la structure de déclancheur dahouet. regate_before_delete
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `regate_before_delete` BEFORE DELETE ON `regate` FOR EACH ROW BEGIN
declare over date;
declare error condition for sqlstate '45001';
select challenge.DATE_FIN from challenge where old.ID_CHALL=challenge.ID_CHALL into over;
if (curdate()<=over) then
signal error
set message_text='Le challenge est pas terminé !',mysql_errno=9002;
end if;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

4.4_pr1_call_dist_moy_reg.sql

```
CALL `distmoyreg`('1');
```

4.4_pr1_dist_moy_reg.sql

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- Export de la structure de procedure dahouet. distmoyreg
DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `distmoyreg`(IN `numchallenge` INT)
BEGIN
select challenge.ID_CHALL,challenge.NOM_CHALL,
challenge.DATE_DEBUT,(sum(regate.DISTANCE)) /(count( regate.ID_CHALL)) as distance_moyenne
from challenge
inner join regate on challenge.ID_CHALL=regate.ID_CHALL
where regate.COURUE='y'
and challenge.ID_CHALL=numchallenge;
END//
DELIMITER ;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

4.4_pr2_call_lstequipvoilreg.sql

```
CALL `lstequipvoilreg`('2', 'MILLENIUM')
```

4.4_pr2_ lstequipvoilreg.sql

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- Export de la structure de procedure dahouet. lstequipvoilreg
DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `lstequipvoilreg`(IN `reg` INT, IN `voil` VARCHAR(20))
BEGIN
select personne.NOM, regate.ID_REGATE, regate.DATE_REGATE
from personne
inner join marin on marin.ID_PERS=personne.ID_PERS
inner join participe on marin.ID_MARIN=participe.ID_MARIN
inner join participation on participe.ID_PART=participation.ID_PART
inner join regate on participation.ID_REGATE = regate.ID_REGATE
inner join voilier on participation.ID_VOILIER = voilier.ID_VOILIER
where regate.ID_REGATE = reg
and voilier.NOM_VOILIER = voil
order by personne.NOM asc;
END//
DELIMITER ;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
```

```
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

4.4_pr3_call_lstintercomchal.sql

```
CALL `lstintercomchal`('2014-02-01', '2014-12-12', 'été');
```

4.4_pr3_ lstintercomchal.sql

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- Export de la structure de procedure dahouet. lstintercomchal
DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `lstintercomchal`(IN `datedebut` DATE, IN `datefin` DATE, IN
`chal` VARCHAR(20))
BEGIN
select  personne.NOM,
        commissaires.COM_REG,
        regate.DATE_REGATE,
        challenge.NOM_CHALL,
        voilier.NOM_VOILIER,
        participation.ID_PART,
        participation.STATUT_ARRIV
from personne
inner join commissaires on commissaires.ID_PERS=personne.ID_PERS
inner join membre_de_commision on
commissaires.ID_COMMISSAIRE=membre_de_commision.ID_COMMISSAIRE
inner join regate on regate.ID_COM=membre_de_commision.ID_COM
inner join challenge on challenge.ID_CHALL=regate.ID_CHALL
inner join participation on participation.ID_REGATE=regate.ID_REGATE
inner join voilier on voilier.ID_VOILIER=participation.ID_VOILIER
where challenge.NOM_CHALL=chal
and     regate.DATE_REGATE between datedebut and datefin;
END//
DELIMITER ;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

modèle Voilier :

```
package com.K4M1coder.dahouet.application.methodes.model;

public class Voilier {

	protected int idvoilier;
	protected int owner;
	protected double coef;
	protected String classe;
	protected String name;
	protected int num;

	/**
	 * @param idvoilier
	 * @param owner
	 * @param coef
	 * @param classe
	 * @param name
	 * @param num
	 */
	public Voilier(int idvoilier, int owner, double coef, String classe, String name, int num) {
		super();
		this.idvoilier = idvoilier;
		this.owner = owner;
		this.coef = coef;
		this.classe = classe;
		this.name = name;
		this.num = num;
	}

	public void setIdvoilier(int idvoilier) {
		this.idvoilier = idvoilier;
	}

	public int getIdvoilier() {
		return idvoilier;
	}

	public int getOwner() {
		return owner;
	}

	public void setOwner(int owner) {
		this.owner = owner;
	}

	public double getCoef() {
		return coef;
	}

	public void setCoef(double coef) {
		this.coef = coef;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
```

```
	public String getClasse() {
		return classe;
	}

	public void setClasse(String classe) {
		this.classe = classe;
	}

	public int getNum() {
		return num;
	}

	public void setNum(int num) {
		this.num = num;
	}

	@Override
	public String toString() {
		return  idvoilier+" : "+ name + ", classe " + classe + ", voile N°" + num;
	}


}
```

model Classe :

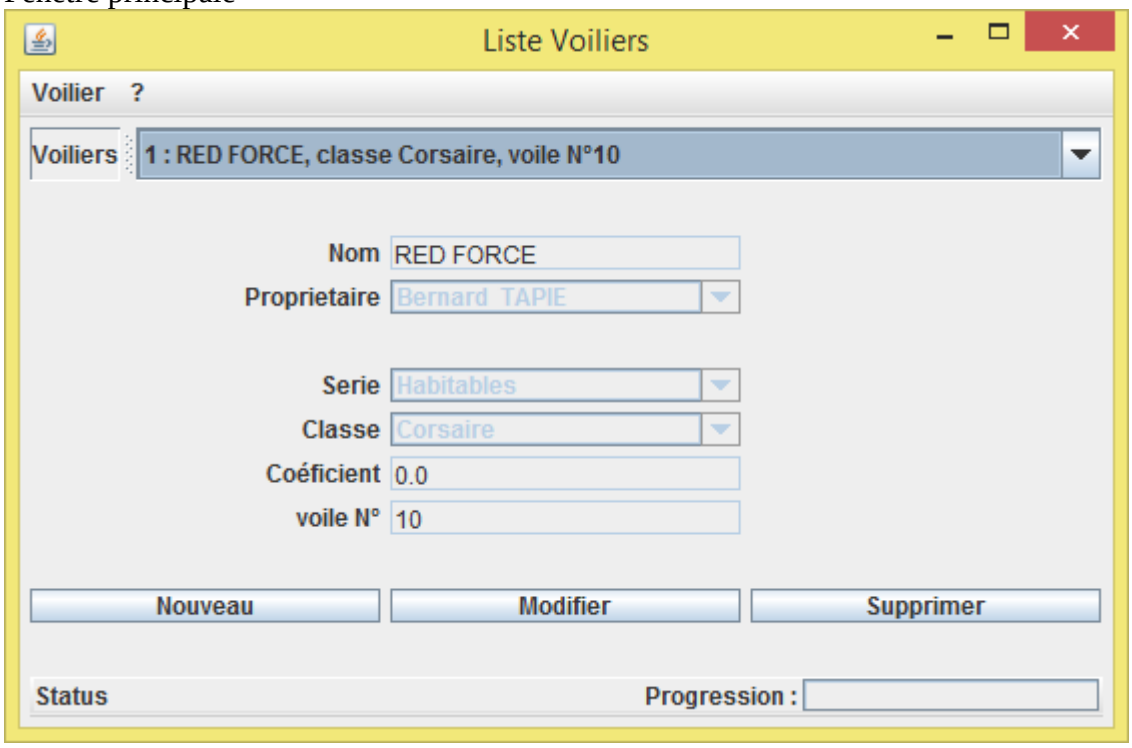
```
package com.K4M1coder.dahouet.application.methodes.model;

public class Serie {

	protected String nomSerie;

	/**
	 * @param nomSerie
	 */
	public Serie(String nomSerie) {
		super();
		this.nomSerie = nomSerie;
	}

	public String getNomSerie() {
		return nomSerie;
	}

	public void setNomSerie(String nomSerie) {
		this.nomSerie = nomSerie;
	}

	@Override
	public String toString() {
		return nomSerie;
	}

}
```

Extrait de la class Control :

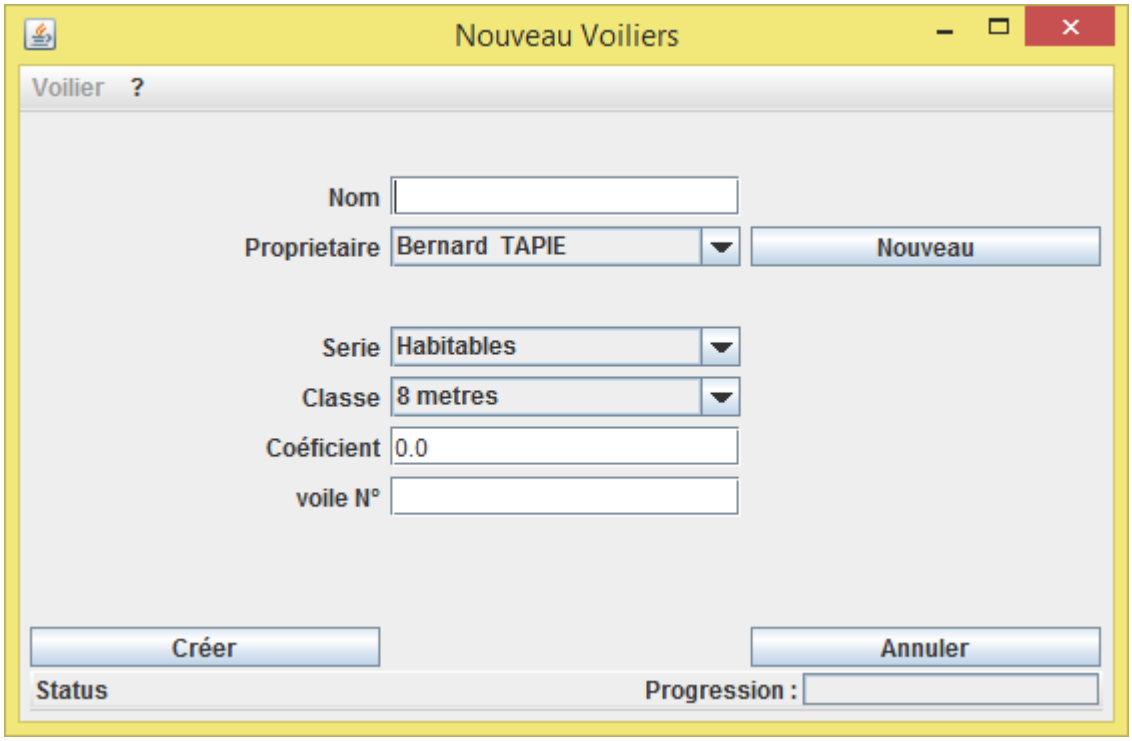
```
package com.K4M1coder.dahouet.application.control;

import java.util.ArrayList;

import com.K4M1coder.dahouet.application.dao.OwnerDAO;
import com.K4M1coder.dahouet.application.dao.PersDAO;
```

```
import com.K4M1coder.dahouet.application.dao.ShipDAO;
import com.K4M1coder.dahouet.application.methodes.model.Classe;
import com.K4M1coder.dahouet.application.methodes.model.Club;
import com.K4M1coder.dahouet.application.methodes.model.Personne;
import com.K4M1coder.dahouet.application.methodes.model.Proprietaire;
import com.K4M1coder.dahouet.application.methodes.model.Serie;
import com.K4M1coder.dahouet.application.methodes.model.Voilier;
import com.K4M1coder.dahouet.application.ui.UiOwner;
import com.K4M1coder.dahouet.application.ui.UiValidated;
import com.K4M1coder.dahouet.application.ui.UiVoilier;


public class Control {

	private UiVoilier frameShips;
	private UiOwner frameOwners;
	private UiValidated frameValidation;

	public static String ctMail(String email) {
		String result = "mail conforme";
		if (email.matches("^[a-zA-Z0-9._-]{2,}+@[a-zA-Z0-9._-]{2,}\\.[a-z]{2,4}$"));
		else
		{ result="Adresse mail invalide"; }
		return result;
	}

	public void initUIVoilier() {
		try {
			frameShips = new UiVoilier();
			frameShips.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public void initUIOwner() {
		try {
			frameOwners = new UiOwner();
			frameOwners.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public void initUIConfirm() {
		try {
			frameValidation = new UiValidated();
			frameValidation.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public ArrayList<Proprietaire> proprioInit() {
		ArrayList<Proprietaire> listOwner = new ArrayList<Proprietaire>();
		listOwner = OwnerDAO.getProprio();
		return listOwner;
	}

/// … …
```

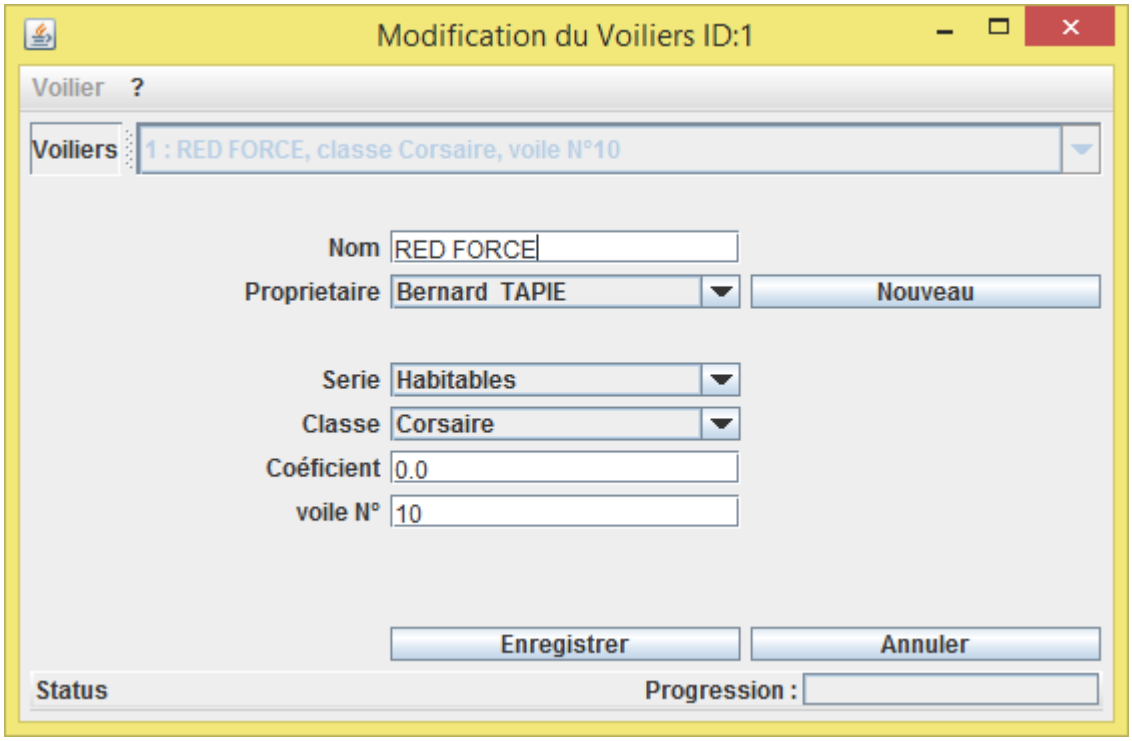
Fenêtre principale



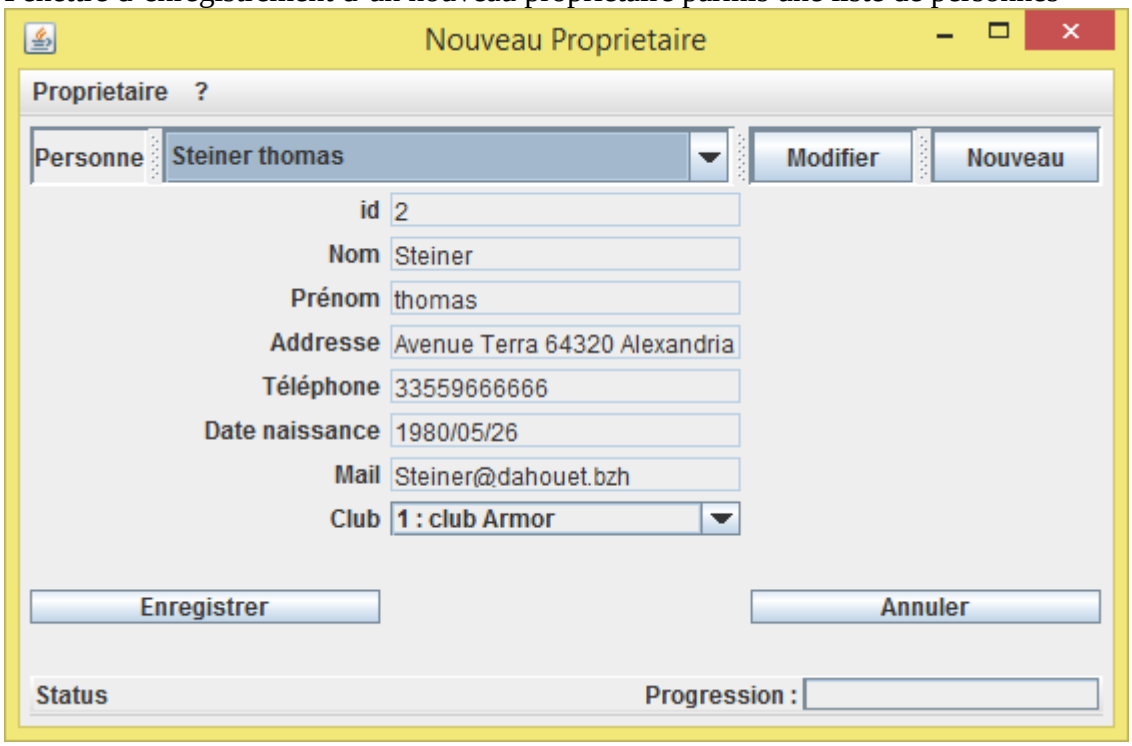
Fenêtre nouveau voilier



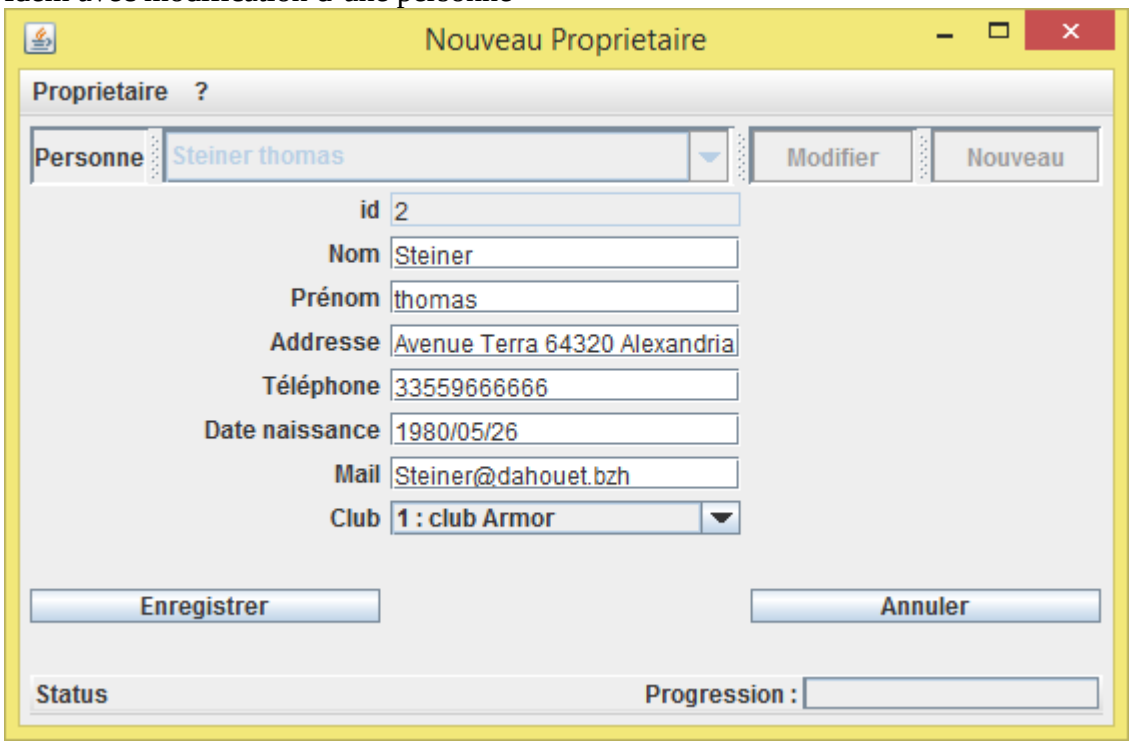
Fenettre de modification d'un voilier



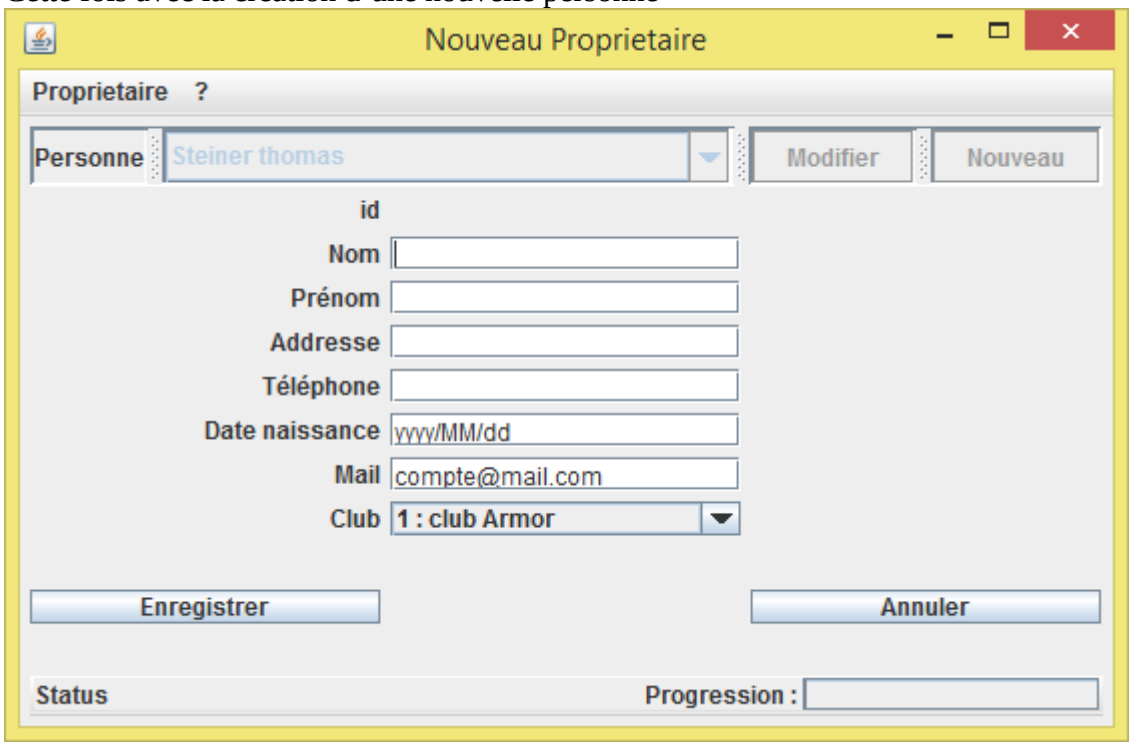
Fenettre d'enregistrement d'un nouveau proprietaire parmis une liste de personnes



Idem avec modification d’une personne



Cette fois avec la création d’une nouvelle personne



Code du connecteur DAO :

```
package com.K4M1coder.dahouet.application.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class connectDAO {
	static Connection con = null;

	public static Connection cConnect() {
		String url = "jdbc:mysql://localhost/dahouet";

		try {
			Class.forName("com.mysql.jdbc.Driver").newInstance();

			// On se connecte via la passerelle jdbc Oracle

			return con = DriverManager.getConnection(url, "dahouet_user",
					"password");
		} catch (SQLException sqlE) {
			System.out.println("Sql Erreur " + sqlE.getMessage());
			return null;
		} catch (Exception e) {
			e.printStackTrace();
			return null;
		}
	}
}
```

Code de la DAO PersDAO :

```
package com.K4M1coder.dahouet.application.dao;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Date;

import com.K4M1coder.dahouet.application.methodes.model.Personne;

public class PersDAO {

	private static Connection c;

	public static ArrayList<Personne> getPers(){

		c = connectDAO.cConnect();

		ArrayList<Personne> persList = new ArrayList<Personne>();

		Statement stm;

		try{
			stm = c.createStatement();
			String sql="select* from Personne";
			ResultSet rs_persList = stm.executeQuery(sql);
			while (rs_persList.next()){
			int idPersonne = rs_persList.getInt("ID_PERS");
			String nom = new String (rs_persList.getString("NOM"));
```

```
                String prenom = new String (rs_persList.getString("PRENOM"));
                String addresse = new String (rs_persList.getString("ADDRESSE"));
                Long telephone = rs_persList.getLong("TELEPHONE");
                Date dateN = rs_persList.getDate("DATE_N");
                String mail = new String(rs_persList.getString("MAIL"));
                Personne pers = new Personne(idPersonne, nom, prenom, addresse, telephone, dateN, mail);
                persList.add(pers);
                }
                rs_persList.close();
                stm.close();
                c.close();
            } catch (SQLException e){
                e.printStackTrace();
            }
            return persList;
        }
}
```