Projet Dahouet

Compte rendu et résultats

Table des matières

I- Module 2 – Concevoir et créer un BDD

I-1. Construction schéma entité/association

I-1.1. dictionnaire de données et règles de gestion

- Dictionnaire de données : **cf annexe 1**

- Règles de gestion :

1. Un propriétaire/commissaire/participant ne peut-être qu'une et une personne
2. Un propriétaire est membre d'un et un seul club
3. Un commissaire appartient à un et un seul comité
4. Une participant possède forcément un numéro de licence
5. Un voilier possède un et un seul propriétaire
6. Un voilier appartient à une et une seule classe
7. Chaque classe possède un coefficient de compensation propre permettant de calculer le temps final à une régate
8. Une classe appartient à une et une seule série
9. Un participant peut-être un équipier ou un skipper
10. Un équipier ne peut-être qu'un et un seul participant
11. Un équipier ne peut participer qu'à une et une seule participation d'un voilier
12. Une participation d'un voilier ne peut avoir qu'un et un seul skipper/voilier/régate
13. Un résultat ne correspondre qu'à une et une seule participation d'un voiliers
14. Un résultat ne peut avoir qu'un et un seul temps/nombre de points
15. Une régate ne peut avoir qu'un seul libellé/date/distance
16. Une régate appartient à un et un seul challenge
17. Un challenge ne peut avoir qu'un unique nom/date début/date fin
18. Un challenge est constitué de 12 régates
19. Un comité de course correspond à l'association d'un commissaire à une régate
20. Une régate doit être comprise entre la date de début et la date de fin du challenge auquel elle est associé
21. La place d'un voilier à l'issue d'une régate ne peut être supérieur au nombre de voilier participant à la régate

22. Une régate ne peut être supprimée tant que le challenge associé n’est pas terminé

I-1.2. Modèle entité/association

Cf annexe 2 ou « Dahouet_exo.asi » (projet Github ici)

I-2. Construction de modèle physique de données

Cf annexe 3 ou « Dahouet_exo.asi » (projet Github ici)

I-3. Scripts SQL de définition et contrainte

```
DROP TABLE IF EXISTS voilier ;
CREATE TABLE voilier (voi_id INT  AUTO_INCREMENT NOT NULL,
voi_num_voile INT,
voi_nom VARCHAR(120),
cla_id INT,
pro_id INT,
PRIMARY KEY (voi_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS serie ;
CREATE TABLE serie (ser_id INT  AUTO_INCREMENT NOT NULL,
ser_nom VARCHAR(25),
PRIMARY KEY (ser_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS classe ;
CREATE TABLE classe (cla_id INT  AUTO_INCREMENT NOT NULL,
cla_coefficient DECIMAL,
cla_nom VARCHAR(25),
ser_id INT NOT NULL,
PRIMARY KEY (cla_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS challenge ;
CREATE TABLE challenge (cha_id INT  AUTO_INCREMENT NOT NULL,
cha_nom VARCHAR(25),
cha_date_debut DATE,
cha_date_fin DATE,
PRIMARY KEY (cha_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS regate ;
CREATE TABLE regate (reg_id INT  AUTO_INCREMENT NOT NULL,
reg_date DATE,
reg_libellé VARCHAR(200),
red_distance DECIMAL,
cha_id INT,
PRIMARY KEY (reg_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS personne ;
CREATE TABLE personne (per_id INT  AUTO_INCREMENT NOT NULL,
per_nom VARCHAR(50),
per_prenom VARCHAR(50),
per_date_naissance DATE,
PRIMARY KEY (per_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS proprietaire ;
CREATE TABLE proprietaire (pro_id INT  AUTO_INCREMENT NOT NULL,
per_id INT NOT NULL,
clu_id INT NOT NULL,
PRIMARY KEY (pro_id) ) ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS commissaire ;
CREATE TABLE commissaire (com_id INT  AUTO_INCREMENT NOT NULL,
per_id INT NOT NULL,
cmt_com_id INT NOT NULL,
PRIMARY KEY (com_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS participant ;
CREATE TABLE participant (par_id INT  AUTO_INCREMENT NOT NULL,
par_numero_ffv INT,
per_id INT,
PRIMARY KEY (par_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS comite_commissaire ;
CREATE TABLE comite_commissaire (cmt_com_id INT  AUTO_INCREMENT NOT NULL,
cmt_com_nom VARCHAR(100),
PRIMARY KEY (cmt_com_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS club_nautique ;
CREATE TABLE club_nautique (clu_id INT  AUTO_INCREMENT NOT NULL,
clu_nom VARCHAR(100),
PRIMARY KEY (clu_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS participation_voilier ;
CREATE TABLE participation_voilier (par_voi_id INT  AUTO_INCREMENT NOT NULL,
par_id INT NOT NULL,
voi_id INT NOT NULL,
reg_id INT NOT NULL,
resultat_res_temps INT NOT NULL,
PRIMARY KEY (par_voi_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS resultat ;
CREATE TABLE resultat (res_temps TIME  AUTO_INCREMENT NOT NULL,
res_points INT,
res_id INT,
participation_voilier_par_voi_id TIME NOT NULL,
PRIMARY KEY (res_temps) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS ETRE_EQUIPIER ;
CREATE TABLE ETRE_EQUIPIER (par_id INT  AUTO_INCREMENT NOT NULL,
par_voi_id INT NOT NULL,
PRIMARY KEY (par_id,
 par_voi_id) ) ENGINE=InnoDB;
DROP TABLE IF EXISTS EST_ASSIGNE ;
CREATE TABLE EST_ASSIGNE (com_id INT  AUTO_INCREMENT NOT NULL,
reg_id INT NOT NULL,
PRIMARY KEY (com_id,
 reg_id) ) ENGINE=InnoDB;
ALTER TABLE voilier ADD CONSTRAINT FK_voilier_cla_id FOREIGN KEY (cla_id)
REFERENCES classe (cla_id);
ALTER TABLE voilier ADD CONSTRAINT FK_voilier_pro_id FOREIGN KEY (pro_id)
REFERENCES proprietaire (pro_id);
ALTER TABLE classe ADD CONSTRAINT FK_classe_ser_id FOREIGN KEY (ser_id)
REFERENCES serie (ser_id);
ALTER TABLE regate ADD CONSTRAINT FK_regate_cha_id FOREIGN KEY (cha_id)
REFERENCES challenge (cha_id);
ALTER TABLE proprietaire ADD CONSTRAINT FK_proprietaire_per_id FOREIGN KEY
(per_id) REFERENCES personne (per_id);
ALTER TABLE proprietaire ADD CONSTRAINT FK_proprietaire_clu_id FOREIGN KEY
(clu_id) REFERENCES club_nautique (clu_id);
ALTER TABLE commissaire ADD CONSTRAINT FK_commissaire_per_id FOREIGN KEY
(per_id) REFERENCES personne (per_id);
ALTER TABLE commissaire ADD CONSTRAINT FK_commissaire_cmt_com_id FOREIGN KEY
(cmt_com_id) REFERENCES comite_commissaire (cmt_com_id);
ALTER TABLE participant ADD CONSTRAINT FK_participant_per_id FOREIGN KEY
(per_id) REFERENCES personne (per_id);
ALTER TABLE participation_voilier ADD CONSTRAINT FK_participation_voilier_par_id
FOREIGN KEY (par_id) REFERENCES participant (par_id);
ALTER TABLE participation_voilier ADD CONSTRAINT FK_participation_voilier_voi_id
FOREIGN KEY (voi_id) REFERENCES voilier (voi_id);
```

```
ALTER TABLE participation_voilier ADD CONSTRAINT FK_participation_voilier_reg_id
FOREIGN KEY (reg_id) REFERENCES regate (reg_id);
ALTER TABLE participation_voilier ADD CONSTRAINT
FK_participation_voilier_resultat_res_temps FOREIGN KEY (resultat_res_temps)
REFERENCES resultat (res_temps);
ALTER TABLE resultat ADD CONSTRAINT FK_resultat_participation_voilier_par_voi_id
FOREIGN KEY (participation_voilier_par_voi_id) REFERENCES participation_voilier
(par_voi_id);
ALTER TABLE ETRE_EQUIPIER ADD CONSTRAINT FK_ETRE_EQUIPIER_par_id FOREIGN KEY
(par_id) REFERENCES participant (par_id);
ALTER TABLE ETRE_EQUIPIER ADD CONSTRAINT FK_ETRE_EQUIPIER_par_voi_id FOREIGN KEY
(par_voi_id) REFERENCES participation_voilier (par_voi_id);
ALTER TABLE EST_ASSIGNE ADD CONSTRAINT FK_EST_ASSIGNE_com_id FOREIGN KEY
(com_id) REFERENCES commissaire (com_id);
ALTER TABLE EST_ASSIGNE ADD CONSTRAINT FK_EST_ASSIGNE_reg_id FOREIGN KEY
(reg_id) REFERENCES regate (reg_id);
```

I-4. Administrer la BDD

- Création d'un utilisateur aux droits restreints : utilisation du menu « Privilèges » de phpmyadmin
- Procédures de sauvegarde et restauration : Utiliser l'option « Exporter » de phpmyadmin pour sauvegarder le fichier sql de la bdd ainsi que le jeu test de données (cf « dahouetReg.sql ». Pour la restauration, importer le fichier sql sauvegardé.

I-5. Manipulation des données SQL

- **Requête 1 :**

SELECT c.cha_nom, AVG(r.reg_distance)

from challenge c

INNER JOIN regate r ON c.cha_id = r.cha_id

group BY c.cha_nom

Resultat :

cha_nom	AVG(r.reg_distance)
été	6.539167
hiver	6.763333

- **Requête 2 :**

```
select p.per_nom, p.per_prenom, par.par_numero_ffv, r.reg_date, r.reg_libelle, v.voi_nom,
par_voi.voi_skipper_id
from personne p
inner join participant par on p.per_id = par.per_id
inner join equipier e on par.par_id = e.par_id
inner join participation_voilier par_voi on e.par_voi_id = par_voi.par_voi_id
inner join regate r on par_voi.reg_id = r.reg_id
inner join voilier v on par_voi.voi_id = v.voi_id
where r.reg_id = "3"
```

Resultat :

per_nom	per_prenom	par_numero_ffv	reg_date	reg_libellé	voi_nom	voi_skipper_id
Sonia	Hervochon	68463	2017-11-05	Regate d'été de la Baie 1	L'écume des jours	1

- **Requête 3 :**

```
select r.reg_libelle, r.reg_date, p.per_nom, p.per_prenom, com_com.cmt_com_nom
from regate r
inner join comite_course com_cou on com_cou.reg_id = r.reg_id
inner join commissaire c on c.com_id = com_cou.com_id
inner join comite_commissaire com_com on com_com.cmt_com_id = c.cmt_com_id
inner join personne p on c.per_id = p.per_id
where r.reg_date > curdate()
```

Resultat :

reg_libellé	reg_date	per_nom	per_prenom	cmt_com_nom
Regate d'été de la Baie 1	2017-11-05	Jezecquel	Margaux	Comité de Bretagne
Regate d'été de la Baie 5	2017-12-03	Lelu	Florent	Comité d'Ile de France
Regate d'été de la Baie 2	2017-11-12	Ursache	Ovidiu	Comité du Poitou-Charente

I-6. Programmation SQL

- **Fonction**

```
CREATE DEFINER=`root`@`localhost` FUNCTION `getRegCode`(reg_id int(4)) RETURNS varchar(10) CHARSET latin1
BEGIN
DECLARE regCode varchar(10);
DECLARE chaCode int(1);
DECLARE regMois int(2);
DECLARE numSeq int(2);

SELECT r.cha_id into chaCode from regate r where r.reg_id = reg_id;
SELECT MONTH(r.reg_date) into regMois from regate r where r.reg_id = reg_id;
SELECT COUNT(DISTINCT(r.reg_id)) into numSeq FROM regate r WHERE r.cha_id = chaCode;

SET regCode = CONCAT(chaCode, regMois, numSeq);
RETURN regCode;
END
```

- **Triggers**

1. Création

```
CREATE DEFINER=`root`@`localhost` TRIGGER `dahouetReg`.`regate_BEFORE_INSERT` BEFORE INSERT ON `regate` FOR EACH ROW
BEGIN
DECLARE testingDate DATE;
DECLARE startingDate DATE;
DECLARE endingDate DATE;
DECLARE msg varchar(100);

SELECT cha_date_debut INTO startingDate FROM challenge WHERE cha_id = new.cha_id;
SELECT cha_date_fin INTO endingDate FROM challenge WHERE cha_id = new.cha_id;

IF startingDate > new.reg_date OR endingDate < new.reg_date THEN
SET msg = 'La date ne correspond pas';
signal sqlstate '45000' set message_text = msg, MYSQL_ERRNO = 45000;
   END IF;
   END
```

2. MAJ

```
CREATE DEFINER=`root`@`localhost` TRIGGER `dahouetReg`.`resultat_BEFORE_UPDATE`
BEFORE UPDATE ON `resultat` FOR EACH ROW

BEGIN

DECLARE nbPartVoi int(3);

DECLARE tempRegId int(3);

DECLARE msg varchar(120);


SELECT p.reg_id INTO tempRegId FROM resultat r INNER JOIN participation_voilier p on
p.par_voi_id = r.par_voi_id WHERE r.par_voi_id = new.par_voi_id GROUP BY p.reg_id;

SELECT COUNT(DISTINCT(voi_id)) INTO nbPartVoi FROM participation_voilier WHERE
reg_id = tempRegId;


IF new.res_points > nbPartVoi THEN

SET msg = 'Il y a trop de resultats par rapport au nombre de participants';

signal sqlstate '45001' set message_text = msg;

END IF;

END
```

3. Suppression

```
CREATE DEFINER=`root`@`localhost` TRIGGER `dahouetReg`.`regate_BEFORE_DELETE`
BEFORE DELETE ON `regate` FOR EACH ROW

BEGIN

DECLARE currentDate DATE;

DECLARE endingDateCha Date;

DECLARE msg varchar(120);


SET currentDate = CURDATE();

SELECT cha_date_fin into endingDateCha FROM challenge WHERE cha_id = old.cha_id;


IF currentDate < endingDateCha THEN

SET msg = 'Impossible de supprimer cette régate, le challenge est toujours en cours ou n\'est pas
commencé';

signal sqlstate '45001' set message_text = msg;

END IF;

END
```

- **Procédures stockées**

Procédure 1 : moyenne des distances des régates par challenge

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getRegDistMoy`(IN cha_id int(1))
BEGIN
SELECT c.cha_nom, AVG(r.reg_distance) AS moy_dist_reg
FROM challenge c
INNER JOIN regate r ON c.cha_id = r.cha_id
WHERE c.cha_id = cha_id
GROUP BY c.cha_nom;
END
```

Procédure 2 : récupérer l'équipage d'un voilier

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getEquipageVoilier`(in voi_id int(3), in reg_id int(3))
BEGIN
SELECT per.per_nom, per.per_prenom, p.voi_skipper_id
FROM participation_voilier p
INNER JOIN equipier e ON p.par_voi_id = e.par_voi_id
INNER JOIN  participant par ON e.par_id = par.par_id
INNER JOIN personne per ON par.per_id = per.per_id
WHERE p.voi_id = voi_id AND p.reg_id = reg_id;
END
```

Procédure 3 : intervention commissaire par challenge entre deux dates

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getCommissaires`(in startDate DATE, in endDate DATE)
BEGIN
SELECT p.per_nom, p.per_prenom, com_com.cmt_com_nom
FROM personne p
INNER JOIN commissaire c ON p.per_id = c.per_id
INNER JOIN comite_commissaire com_com ON c.cmt_com_id = com_com.cmt_com_id
INNER JOIN comite_course com_cou ON c.com_id = com_cou.com_id
INNER JOIN regate r ON com_cou.reg_id = r.reg_id
INNER JOIN challenge cha ON r.cha_id = cha.cha_id
WHERE cha.cha_date_debut = startDate AND cha.cha_date_fin = endDate;
END
```

II. Module 1 – Développer logiciel informatique

II-1. Algorithme

Cf programme « java/emailCheck »

ou

https://github.com/lfourteau/AFPA_ValidationTests/tree/master/bdd-Java-Android-Modules_dahouetReg/java/emailCheck

II-2. Développement objet

Cf programme « java/persGestValidation »

ou

https://github.com/lfourteau/AFPA_ValidationTests/tree/master/bdd-Java-Android-Modules_dahouetReg/java/persGestValidation

Classe Personne : les trois autres classes étendent celle-ci

```java
package model;

import java.util.Calendar;

public class Personne {

    String nom;
    String prenom;
    String email;
    int anneeNaissance;

    public Personne(String nom, String prenom, String email, int anneeNaissance) {
        this.nom = nom;
        this.prenom = prenom;
        this.email = email;
        this.anneeNaissance = anneeNaissance;
    }

    private int calculAge() {
        int actualYear = Calendar.getInstance().get(Calendar.YEAR);
        int age = actualYear-anneeNaissance;
        return age;
    }

    public int getAnneeNaissance() {
        return anneeNaissance;
    }

    @Override
    public String toString() {
        return prenom + " " + nom + " qui a " + calculAge() + " ans";
    }
}
```

Classe Commissaire (exemple de toString):

```java
@Override
public String toString() {
    return  super.toString() + " est commissaire du comité de " + commite;
}
```

Classe Licencié (service calculPoints) :

```
30    public double calculPoints(double points, Calendar annee) {
31
32        if (anneeLicence == annee.get(Calendar.YEAR)) {
33            nbPoints = pointsFFV + points;
34            System.out.println("Après ajout des derniers resultats, " + getPrenom() + " " + getNom() + " possède : " + nbPoints);
35        } else {
36            System.out.print("La date ne corresponde à l'année de la licence pour " + getPrenom() + " " + getNom());
37        }
38        return nbPoints;
39    }
```

II-3. Développement de l'interface graphique client/serveur

- Maquette : **cf annexe 4**
- composant logiciel : **cf programme « java/shipRegister** »

ou

https://github.com/lfourteau/AFPA_ValidationTests/tree/master/bdd-Java-Android-Modules_dahouetReg/java/shipRegister

II-4. Outil de génération d'état

Cf « Liste_depart.jrxml » (projet Github ici)

III. Module 5 – Développer une application mobile

III.1. Maquette

Cf annexe 5

III.2. Développement de l'appli

Cf programme « androidDahouetRegResults/ »

ou

https://github.com/lfourteau/AFPA_ValidationTests/tree/master/bdd-Java-Android-Modules_dahouetReg/android/dahouetRegResults

N.B : service API permettant d'accéder aux données :

Cf programme « androidDahouetRegResults/dahouetSlim »

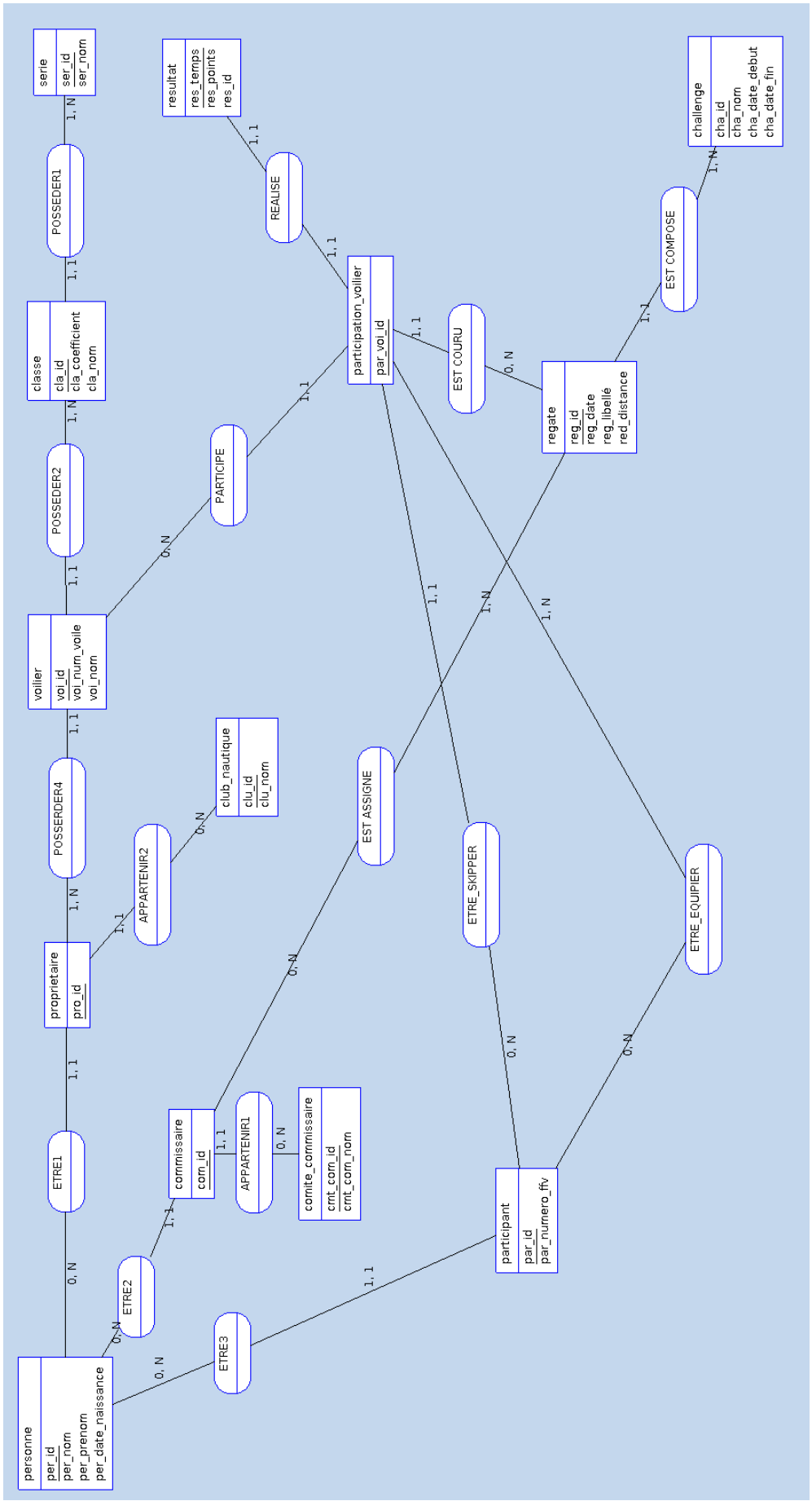
ou

https://github.com/lfourteau/AFPA_ValidationTests/tree/master/bdd-Java-Android-Modules_dahouetReg/android/dahouetRegResults/dahouetSlim

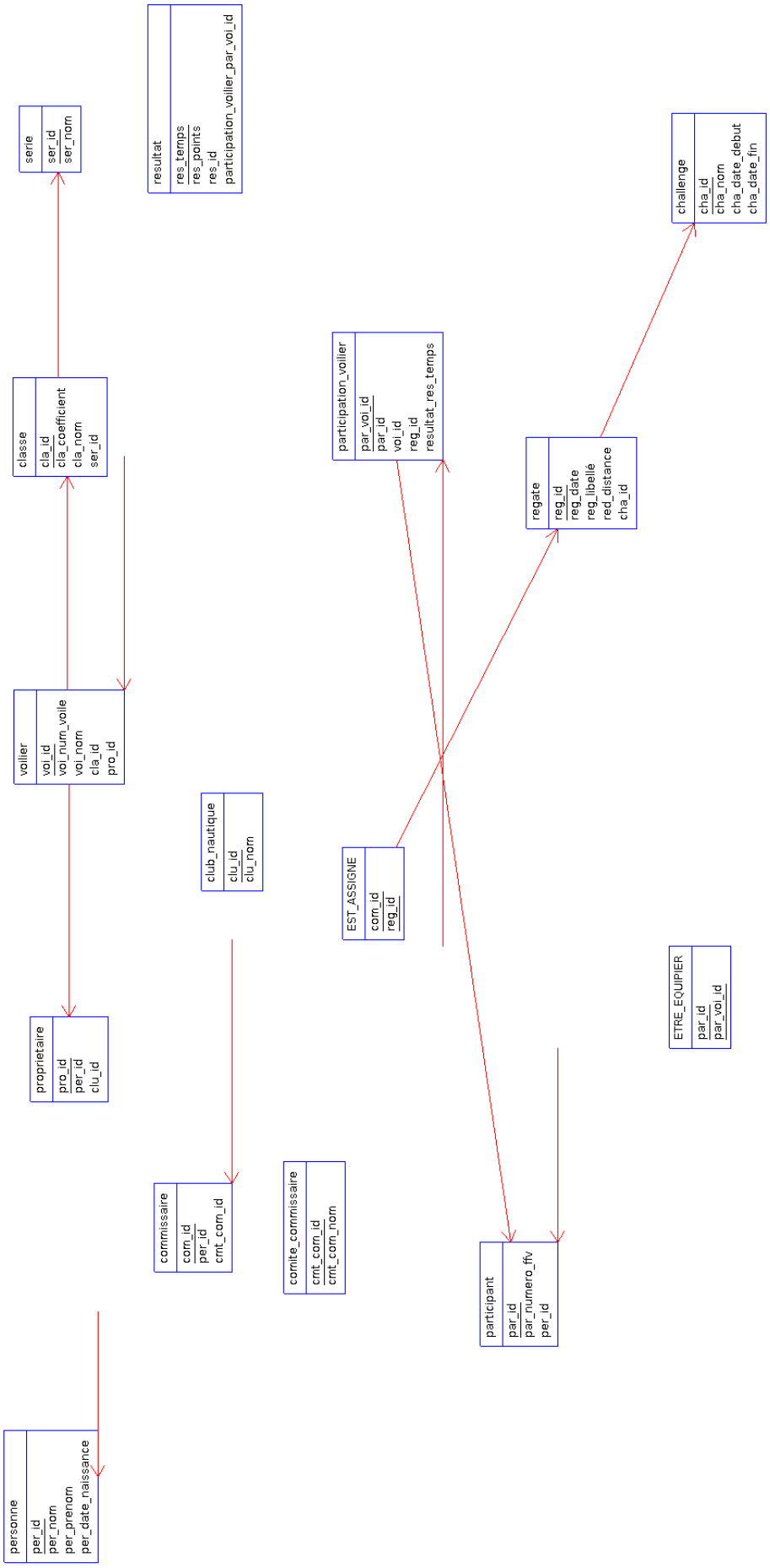
Annexe 1 – Dictionnaire de données

Code	Libellé	Type		Table
cha_date_debut	Date de début de challenge	DATE	0	challenge
cha_date_fin	Date de fin de challenge	DATE	0	challenge
cha_id	Identifiant du challenge	INT_AUTO_INCREMENT	0	challenge
cha_nom	Nom du challenge	VARCHAR	25	challenge
cla_coefficient	Coefficient de compensation de la classe	DECIMAL	0	classe
cla_id	Identifiant de la classe	INT_AUTO_INCREMENT	0	classe
cla_nom	Nom de la classe	VARCHAR	25	classe
clu_id	Identifiant du club	INT_AUTO_INCREMENT	0	club_nautique
clu_nom	Nom du club	VARCHAR	100	club_nautique
cmt_com_id	Identifiant du comité des commissaires	INT_AUTO_INCREMENT	0	comite_commissaire
cmt_com_nom	Nom du comité des commissaires	VARCHAR	100	comite_commissaire
com_id	Identifiant du commissaire	INT_AUTO_INCREMENT	0	commissaire
par_id	Identifiant du participant	INT_AUTO_INCREMENT	0	participant
par_numero_ffv	Numéro de licence FFV du participant	INT	0	participant
par_voi_id	Identifiant de la participation voilier	INT_AUTO_INCREMENT	0	participation_voilier
per_date_naissance	Date naissance personne	DATE	0	personne
per_id	Identifiant personne	INT_AUTO_INCREMENT	0	personne
per_nom	Nom personne	VARCHAR	50	personne
per_prenom	Prénom personne	VARCHAR	50	personne
pro_id	Identifiant propriétaire	INT_AUTO_INCREMENT	0	proprietaire
red_distance	Distance régate	DECIMAL	0	regate
reg_date	Date régate	DATE	0	regate
reg_id	Identifiant régate	INT_AUTO_INCREMENT	0	regate
reg_libellé	Nom de la régate	VARCHAR	200	regate
res_id	Identifiant résultat	INT_AUTO_INCREMENT	0	resultat
res_points	Points résultat	INT	0	resultat
res_temps	Temps resultat	TIME	0	resultat
ser_id	Identifiant série	INT_AUTO_INCREMENT	0	serie
ser_nom	Nom série	VARCHAR	25	serie
voi_id	Identifiant voilier	INT_AUTO_INCREMENT	0	voilier
voi_nom	Nom voilier	VARCHAR	120	voilier
voi_num_voile	Numéro de voile du voilier	INT	0	voilier

Annexe 2 – Modèle entité/association

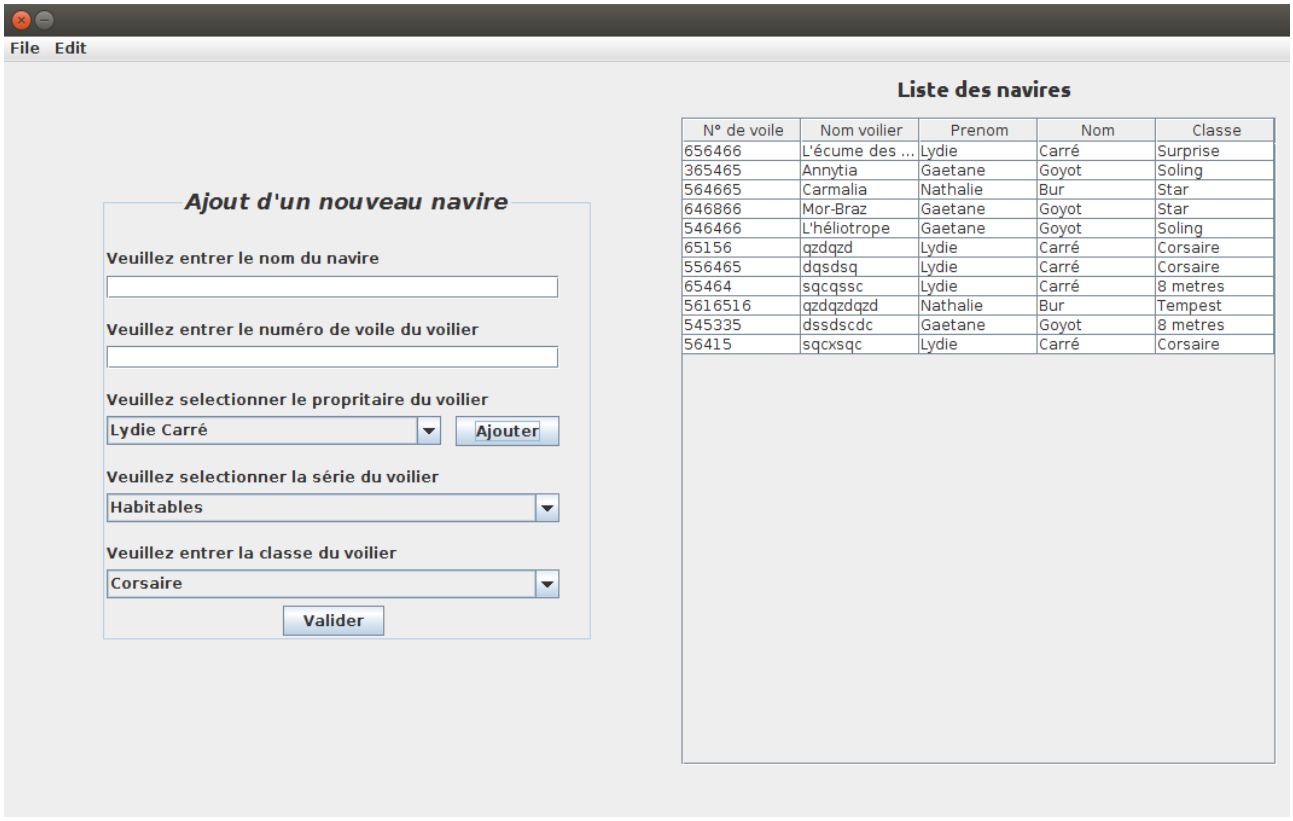


Annexe 3 – Modèle physique de données

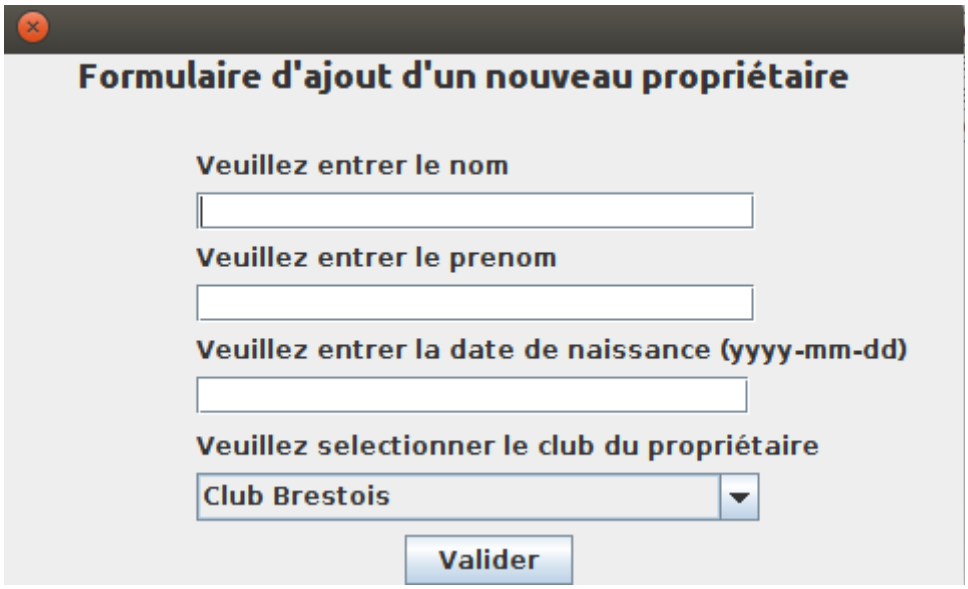


Annexe 4 – Maquette de logiciel shipRegister

En ouvrant le logiciel, la première page est la suivante. Elle permet l'ajout de navires à la bdd :

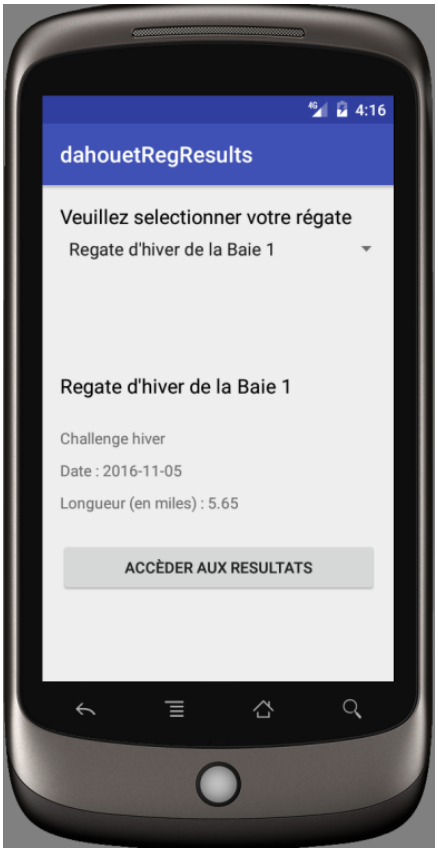


En cliquant sur le bouton « Ajouter », la fenêtre suivante s'ouvre (JDialog) et permet l'ajout d'un nouveau propriétaire :



Annexe 5 – Maquette de l'application mobile dahouetRegResults

L'application démarre sur la page suivante permettant d'accéder aux différentes régates via un menu déroulant. Le détails des régates s'affiche ensuite dans le bas de page.



En cliquant sur le bouton «accéder aux résultats », il est possible d'accéder à la page suivante affichant les résultats pour la régate sélectionnée