

ÉVALUATION

Module 1 – Développer l'interface d'une application informatique

Module 2 – Concevoir et mettre en place une base de donnée

Module 5 – Développer une application simple de mobilité numérique

Dossier de synthèse – Projet Dahouet



Table des matières

I. MODULE 2 – CONCEVOIR ET METTRE EN PLACE UNE BASE DE DONNÉES.....	3
I.1 Construire le schéma entité/association.....	4
I.2 Construire le schéma physique.....	7
I.3 Script SQL de définition des données et des contraintes.....	8
I.4 Administrer la base de données de test.....	14
I.5 Manipuler les données avec SQL.....	14
I.6 Programmer dans le langage du SGBD.....	16
II. <i>MODULE 1 – DÉVELOPPER L'INTERFACE D'UNE APPLICATION INFORMATIQUE</i>	20
II.1 Écrire un algorithme.....	20
II.2 Développement objet :.....	21
II.3 Développer l'interface graphique client/serveur.....	27
II.4 Mettre en œuvre un outil de génération d'état.....	29
III. <i>MODULE 5 – DÉVELOPPER UNE APPLICATION SIMPLE DE MOBILITÉ NUMÉRIQUE</i>	33
III.1 Arbrescences :.....	33
III.2 Maquette de l'application :.....	35

I. MODULE 2 – CONCEVOIR ET METTRE EN PLACE UNE BASE DE DONNÉES

I.1 Construire le schéma entité/association

Nom des données	ID	Type	Taille	Entité	Champ
nom de challenge	nom_de_challenge	VARCHAR	150	Challenge	challenge.name
date de début	date_de_début	DATETIME	0	Challenge	challenge.begin
date de fin	date_de_fin	DATETIME	0	Challenge	challenge.end
id challenge	id_challenge	INT	11	Challenge	challenge.id
code challenge	code_challenge	VARCHAR	5	Challenge	challenge.code
nom de classe	nom_de_classe	VARCHAR	150	Classe	sbclass.name
coef	coef	FLOAT	5	Classe	sbclass.coef
id classe	id_classe	INT	11	Classe	sbclass.id
nom du club	nom_du_club	VARCHAR	150	Club	club.name
id club	id_club	INT	11	Club	club.id
nom du comité	nom_du_comité	VARCHAR	150	Comité	committee.name
id comité	id_comité	INT	11	Comité	committee.id
id commissaire	id_commissaire	INT	11	Commissaire	auditor.id
num inscription	num_inscription	INT	5	concourt	compete.id
temps réel	temps_réel	BIGINT	11	concourt	compete.realtime
num de points	num_de_points	INT	4	concourt	compete.point
inscription validé	inscription_validé	BOOL	1	concourt	compete.valid
position	position	INT	5	concourt	compete.position
id équipage	id_équipage	INT	11	Equipage	(limitation analyseSI)
id jury	id_jury	INT	11	Jury	jury.id
affiliationFFV	affiliationFFV	BOOL	0	Participant	entrant.ffv
année de licence	année_delicence	INT	4	Participant	entrant.year_permit
date de naissance	date_de_naissance	DATETIME	0	Participant	entrant.birth
id participant	id_participant	INT	11	Participant	entrant.id
num licence	num_licence	INT	11	Participant	entrant.num_licence
nom de la personne	nom_de_la_personne	VARCHAR	150	Personne	person.lastname
prénom de la personne	prénom_de_la_personne	VARCHAR	150	Personne	person.firstname
email de la personne	email_de_la_personne	VARCHAR	150	Personne	person.email
mot de passe	mot_de_passe	VARCHAR	10	Personne	person.password
id personne	id_personne	INT	11	Personne	person.id
nom de présence	nom_de_présence	VARCHAR	150	Présence	report.name
id présence	id_présence	INT	11	Présence	report.id
code de présence	code_de_présence	VARCHAR	4	Présence	report.code
id propriétaire	id_propriétaire	INT	11	Propriétaire	owner.id
nom de régate	nom_de_régate	VARCHAR	150	Régate	regatta.name
date de régate	date_de_régate	DATETIME	0	Régate	regatta.date
distance de course	distance_de_course	INT	10	Régate	regatta.distance
id régate	id_régate	INT	11	Régate	regatta.id
code régate	code_régate	VARCHAR	10	Régate	regatta.code
id secrétaire	id_secrétaire	INT	11	Secrétaire	secretary.id
nom de série	nom_de_série	VARCHAR	150	Série	serie.name
id série	id_série	INT	11	Série	serie.id
num de voile	num_de_voile	BIGINT	0	Voilier	sailboat.num_sail
id voilier	id_voilier	INT	11	Voilier	sailboat.id

Règles de gestions :

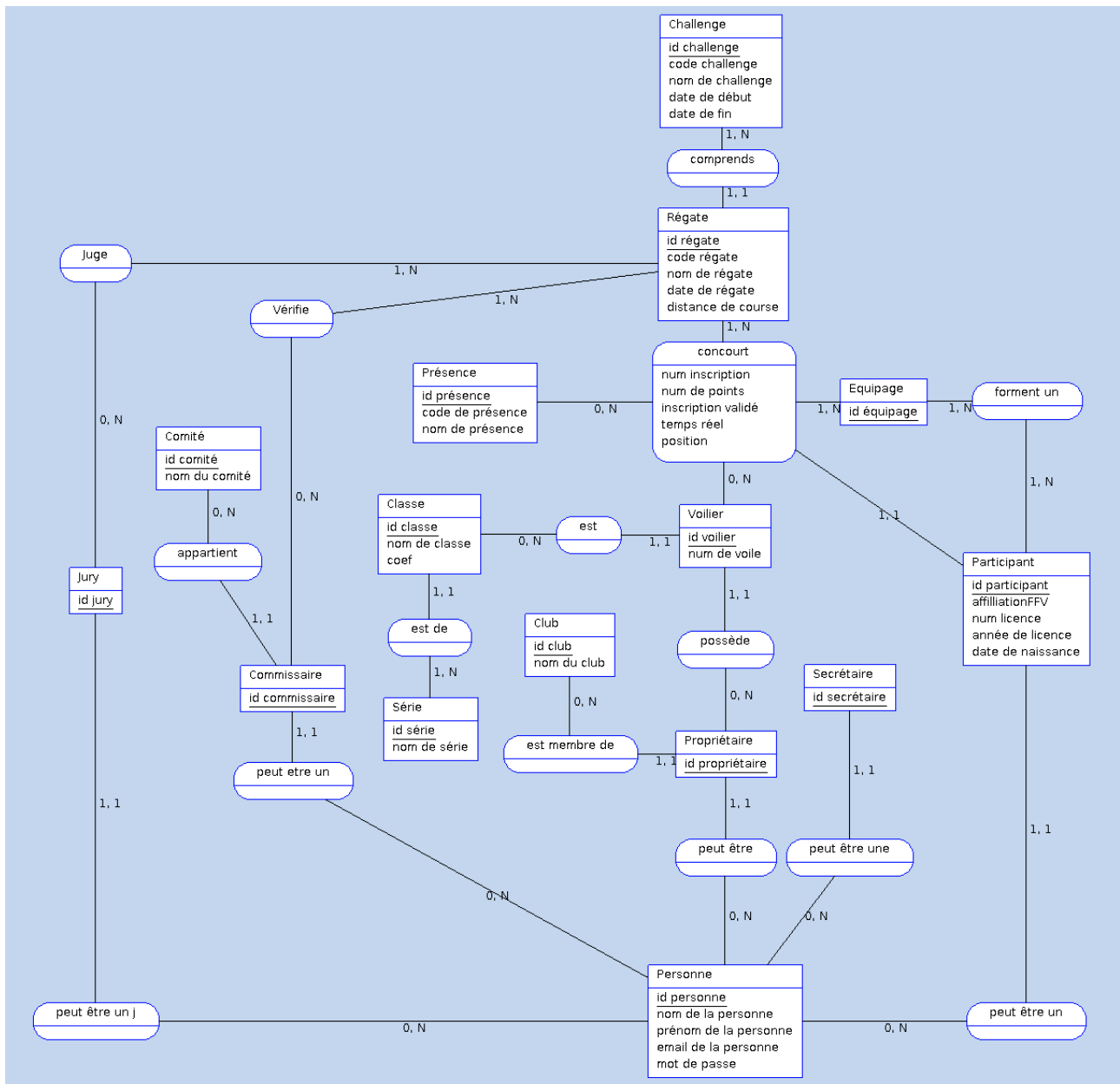
1. Il y a deux challenges par ans,
2. un challenge comprend 12 régates,
3. le temps compensé est calculé par une formule en fonction du temps réel, de la distance d'une régate et du coefficient de la classe du voilier,
4. Le classement se fait dans l'ordre croissant des temps compensés,
5. Le classement final se fait dans l'ordre croissant du nombre de points obtenu lors de chaque régate,
6. une régate comprend plusieurs voiliers
7. un propriétaire peut posséder plusieurs voiliers,
8. le propriétaire peut inscrire ses voiliers à plusieurs régates,
9. un propriétaire fait partie d'un club,
10. un propriétaire est une personne,
11. un voilier est d'une seule classe,
12. un voilier possède plusieurs membres d'équipage,
13. un voilier possède un seul skipper par régate,
14. un skipper est un membre d'équipage,
15. un membre d'équipage est une personne,
16. une classe est comprise dans une seule série,
17. une série peut posséder plusieurs classes,
18. une régate peut posséder plusieurs jurys,
19. un jury est une personne,
20. une régate peut posséder plusieurs commissaires,
21. un commissaire fait partie d'un comité de région,
22. un commissaire est une personne,
23. une secrétaire est une personne.

```
classDiagram
    class Challenge {
        code_challenge
        nom_de_challenge
        date_de_debut
        date_de_fin
    }
    class Régate {
        code_régate
        nom_de_régate
        date_de_régate
        distance_de_course
    }
    class Présence {
        code_de_présence
        nom_de_présence
    }
    class Classe {
        nom_de_classe
        coef
    }
    class Voilier {
        num_de_voile
    }
    class Club {
        nom_du_club
    }
    class Série {
        nom_de_série
    }
    class Participant {
        affiliationFFV
        num_licence
        année_de_licence
        date_de_naissance
    }
    class Comité {
        nom_du_comité
    }
    class Jury {
    }
    class Commissaire {
    }
    class Propriétaire {
    }
    class Secrétaire {
    }
    class Personne {
        nom_de_la_personne
        prénom_de_la_personne
        email_de_la_personne
        mot_de_passe
    }

    Challenge "1" -- "N" Régate : comprends
    Régate "1" -- "1" Présence
    Régate "1" -- "N" concourt
    Présence "0" -- "N" concourt
    concourt "1" -- "N" Equipage
    Equipage "1" -- "N" forment_un
    forment_un "1" -- "N" Participant
    Participant "1" -- "1" concourt
    Classe "1" -- "N" est Voilier
    Voilier "1" -- "1" possède
    possède "0" -- "N" Propriétaire
    Propriétaire "1" -- "1" peut_être
    peut_être "0" -- "N" Personne
    Club "0" -- "N" est_membre_de
    est_membre_de "1" -- "1" Propriétaire
    Propriétaire "1" -- "1" peut_être_une
    peut_être_une "0" -- "N" Personne
    Série "1" -- "N" est_de
    est_de "1" -- "N" Participant
    Comité "0" -- "N" appartient
    appartient "1" -- "1" Commissaire
    Commissaire "1" -- "1" peut_être_un
    peut_être_un "0" -- "N" Personne
    Jury "0" -- "N" Juge
    Juge "1" -- "N" Régate
    Régate "1" -- "N" Vérifie
    Vérifie "1" -- "N" concourt
    Personne "0" -- "N" peut_être_un_j
    peut_être_un_j "1" -- "1" Jury
```

I.2 Construire le schéma physique

(À cause des limitations du logiciel AnalyseSI, j'ai été dans l'obligation de ne ressortir que le MCD modifié, le MPD n'était pas présentable)



Modèle logique des données :

Challenge (id_challenge, code_challenge, nom_de_challenge, date_de_début, date_de_fin)

```

Régate (id_régate, code_régate, nom_de_régate, date_de_régate,
distance_de_course, #id_challenge)

```

```
Voilier (id_voilier, num_de_voile, #id_proprietaire, #id_classe)
```

Propriétaire (id_propriétaire, #id_personne, #id_club)

Personne (id_personne, nom_de_la_personne, prénom_de_la_personne,
 email_de_la_personne, mot_de_passe)
 Participant (id_participant, affiliationFFV, num_licence, année_de_licence,
 date_de_naissance, #id_personne)
 Secrétaire (id_secrétaire, #id_personne)
 Commissaire (id_commissaire, #id_comité, #id_personne)
 Jury (id_jury, #id_personne)
 Classe (id_classe, nom_de_classe, coef, #id_série)
 Série (id_série, nom_de_série)
 Comité (id_comité, nom_du_comité)
 Club (id_club, nom_du_club)
 Présence (id_présence, code_de_présence, nom_de_présence)
 Equipage (idéquipage)
 concourt (id_régate, id_voilier, id_présence, id_participant, idéquipage,
 num_inscription, num_de_points, inscription_validé, temps_réel, position)
 Vérifie (id_commissaire, id_régate)
 Juge (id_jury, id_régate)
 forment_un (id_participant, idéquipage)

1.3 Script SQL de définition des données et des contraintes

```

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

```

```

CREATE TABLE `auditor` (
  `id` int(11) NOT NULL,
  `committee_id` int(11) NOT NULL,
  `person_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

CREATE TABLE `board` (
  `jury_id` int(11) NOT NULL,
  `regatta_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

CREATE TABLE `challenge` (
  `id` int(11) NOT NULL,
  `code` varchar(5) NOT NULL,
  `name` varchar(150) NOT NULL,
  `begin` datetime NOT NULL,
  `end` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

CREATE TABLE `club` (
  `id` int(11) NOT NULL,
  `name` varchar(150) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

CREATE TABLE `committee` (
  `id` int(11) NOT NULL,
  `name` varchar(150) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```
CREATE TABLE `compete` (  
  `id` int(11) NOT NULL,  
  `regatta_id` int(11) NOT NULL,  
  `sailboat_id` int(11) NOT NULL,  
  `report_id` int(11) DEFAULT NULL,  
  `entrant_id` int(11) NOT NULL,  
  `point` int(11) NOT NULL DEFAULT '0',  
  `valid` tinyint(1) DEFAULT NULL,  
  `realtime` float NOT NULL,  
  `position` int(11) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `crew` (  
  `entrant_id` int(11) NOT NULL,  
  `compete_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `entrant` (  
  `id` int(11) NOT NULL,  
  `ffv` tinyint(1) NOT NULL,  
  `num_licence` int(11) NOT NULL,  
  `year_permit` int(4) NOT NULL,  
  `birth` datetime NOT NULL,  
  `person_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `jury` (  
  `id` int(11) NOT NULL,  
  `person_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `owner` (  
  `id` int(11) NOT NULL,  
  `person_id` int(11) NOT NULL,  
  `club_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `panel` (  
  `auditor_id` int(11) NOT NULL,  
  `regatta_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `person` (  
  `id` int(11) NOT NULL,  
  `firstname` varchar(150) NOT NULL,  
  `lastname` varchar(150) NOT NULL,  
  `email` varchar(150) NOT NULL,  
  `password` varchar(150) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `regatta` (  
  `id` int(11) NOT NULL,  
  `code` varchar(10) NOT NULL,  
  `name` varchar(150) NOT NULL,  
  `date` datetime NOT NULL,  
  `distance` float NOT NULL,  
  `challenge_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```



```
CREATE TABLE `report` (  
  `id` int(11) NOT NULL,  
  `code` varchar(4) NOT NULL,  
  `name` varchar(150) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `sailboat` (  
  `id` int(11) NOT NULL,  
  `num_sail` int(11) NOT NULL,  
  `owner_id` int(11) NOT NULL,  
  `class_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `sbclass` (  
  `id` int(11) NOT NULL,  
  `name` varchar(150) NOT NULL,  
  `coef` float NOT NULL,  
  `serie_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `secretary` (  
  `id` int(11) NOT NULL,  
  `person_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `serie` (  
  `id` int(11) NOT NULL,  
  `name` varchar(150) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `auditor`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `person_id` (`person_id`),  
  ADD KEY `committee_id` (`committee_id`,`person_id`);
```

```
ALTER TABLE `board`  
  ADD PRIMARY KEY (`jury_id`,`regatta_id`),  
  ADD UNIQUE KEY `jury_id` (`jury_id`,`regatta_id`),  
  ADD KEY `board_regatta` (`regatta_id`);
```

```
ALTER TABLE `challenge`  
  ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `club`  
  ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `committee`  
  ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `compete`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `regatta_id_2` (`regatta_id`,`sailboat_id`),  
  ADD KEY `regatta_id` (`regatta_id`,`sailboat_id`,`entrant_id`),  
  ADD KEY `report_id` (`report_id`),  
  ADD KEY `compete_entrant` (`entrant_id`),  
  ADD KEY `compete_sailboat` (`sailboat_id`);
```

```
ALTER TABLE `crew`
  ADD PRIMARY KEY (`entrant_id`,`compete_id`),
  ADD UNIQUE KEY `entrant_id` (`entrant_id`,`compete_id`),
  ADD KEY `crew_compete` (`compete_id`);

ALTER TABLE `entrant`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `person_id` (`person_id`),
  ADD KEY `sailboat_id` (`person_id`);

ALTER TABLE `jury`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `person_id_2` (`person_id`),
  ADD KEY `person_id` (`person_id`);

ALTER TABLE `owner`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `person_id_2` (`person_id`),
  ADD KEY `person_id` (`person_id`,`club_id`),
  ADD KEY `owner_club` (`club_id`);

ALTER TABLE `panel`
  ADD PRIMARY KEY (`auditor_id`,`regatta_id`),
  ADD UNIQUE KEY `auditor_id` (`auditor_id`,`regatta_id`),
  ADD KEY `panel_regatta` (`regatta_id`);

ALTER TABLE `person`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `regatta`
  ADD PRIMARY KEY (`id`),
  ADD KEY `challenge_id` (`challenge_id`);

ALTER TABLE `report`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `sailboat`
  ADD PRIMARY KEY (`id`),
  ADD KEY `owner_id` (`owner_id`,`class_id`),
  ADD KEY `sailboat_class` (`class_id`);

ALTER TABLE `sbclass`
  ADD PRIMARY KEY (`id`),
  ADD KEY `serie_id` (`serie_id`);

ALTER TABLE `secretary`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `person_id` (`person_id`),
  ADD KEY `person_id_2` (`person_id`);

ALTER TABLE `serie`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `auditor`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE `challenge`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
```

```
ALTER TABLE `club`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

```
ALTER TABLE `committee`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

```
ALTER TABLE `compete`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=15;
```

```
ALTER TABLE `entrant`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=25;
```

```
ALTER TABLE `jury`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
```

```
ALTER TABLE `owner`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=27;
```

```
ALTER TABLE `person`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=40;
```

```
ALTER TABLE `regatta`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
```

```
ALTER TABLE `report`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;
```

```
ALTER TABLE `sailboat`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=14;
```

```
ALTER TABLE `sbclass`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=12;
```

```
ALTER TABLE `secretary`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
```

```
ALTER TABLE `serie`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
```

```
ALTER TABLE `auditor`  
  ADD CONSTRAINT `auditor_committee` FOREIGN KEY (`committee_id`) REFERENCES  
  `committee` (`id`),  
  ADD CONSTRAINT `auditor_person` FOREIGN KEY (`person_id`) REFERENCES `person`  
  (`id`) ON UPDATE CASCADE;
```

```
ALTER TABLE `board`  
  ADD CONSTRAINT `board_jury` FOREIGN KEY (`jury_id`) REFERENCES `jury` (`id`)  
  ON DELETE CASCADE ON UPDATE CASCADE,  
  ADD CONSTRAINT `board_regatta` FOREIGN KEY (`regatta_id`) REFERENCES `regatta`  
  (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```

ALTER TABLE `compete`
  ADD CONSTRAINT `compete_entrant` FOREIGN KEY (`entrant_id`) REFERENCES
`entrant` (`id`) ON UPDATE CASCADE,
  ADD CONSTRAINT `compete_regatta` FOREIGN KEY (`regatta_id`) REFERENCES
`regatta` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `compete_report` FOREIGN KEY (`report_id`) REFERENCES `report`
(`id`) ON DELETE SET NULL ON UPDATE CASCADE,
  ADD CONSTRAINT `compete_sailboat` FOREIGN KEY (`sailboat_id`) REFERENCES
`sailboat` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `crew`
  ADD CONSTRAINT `crew_compete` FOREIGN KEY (`compete_id`) REFERENCES `compete`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `crew_entrant` FOREIGN KEY (`entrant_id`) REFERENCES `entrant`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `entrant`
  ADD CONSTRAINT `entrant_person` FOREIGN KEY (`person_id`) REFERENCES `person`
(`id`) ON UPDATE CASCADE;

ALTER TABLE `jury`
  ADD CONSTRAINT `jury_person` FOREIGN KEY (`person_id`) REFERENCES `person`
(`id`) ON UPDATE CASCADE;

ALTER TABLE `owner`
  ADD CONSTRAINT `owner_club` FOREIGN KEY (`club_id`) REFERENCES `club` (`id`)
ON UPDATE CASCADE,
  ADD CONSTRAINT `owner_person` FOREIGN KEY (`person_id`) REFERENCES `person`
(`id`) ON UPDATE CASCADE;

ALTER TABLE `panel`
  ADD CONSTRAINT `panel_auditor` FOREIGN KEY (`auditor_id`) REFERENCES `auditor`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `panel_regatta` FOREIGN KEY (`regatta_id`) REFERENCES `regatta`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `regatta`
  ADD CONSTRAINT `regatta_challenge` FOREIGN KEY (`challenge_id`) REFERENCES
`challenge` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `sailboat`
  ADD CONSTRAINT `sailboat_class` FOREIGN KEY (`class_id`) REFERENCES `sbclass`
(`id`) ON UPDATE CASCADE,
  ADD CONSTRAINT `sailboat_owner` FOREIGN KEY (`owner_id`) REFERENCES `owner`
(`id`) ON UPDATE CASCADE;

ALTER TABLE `sbclass`
  ADD CONSTRAINT `class_serie` FOREIGN KEY (`serie_id`) REFERENCES `serie`
(`id`);

ALTER TABLE `secretary`
  ADD CONSTRAINT `secretary_person` FOREIGN KEY (`person_id`) REFERENCES
`person` (`id`) ON UPDATE CASCADE;

```

Base de donnée :

- SQL : <https://github.com/malahx/AFPA-PhysicalRegatta/blob/master/BDD/physicalregatta.sql>
- Export ODS : <https://github.com/malahx/AFPA-PhysicalRegatta/raw/master/BDD/physicalregatta.ods>

I.4 Administrer la base de données de test

Procédure pour créer un utilisateur dans phpMyAdmin : dans la table physicalregatta, cliquer sur l'onglet « Privileges », puis sur « Add user account », saisir le nom du nouvel utilisateur et désélectionner « Grant all privileges on database physicalregatta » puis cliquer sur Go. Ensuite cliquer sur l'onglet Database, sélectionner la table physicalregatta et cliquer sur Go. Pour finir cliquer sur Table et sélectionner toutes les tables nécessaires pour afficher les résultats.

Procédure pour sauvegarder la base de donnée : dans la table physicalregatta, cliquer sur export, puis cliquer sur Go, ce qui vous téléchargera par défaut la base de donnée complète (Structure et Donnée). Dans le cas où l'on souhaite juste sauvegarder les données il faut sélectionner l'option « Export method : Custom », désélectionner l'option Structure de toutes nos tables et ensuite cliquer sur Go.

Procédure pour restaurer une sauvegarde des données : dans la table physicalregatta cliquer sur « Import » puis cliquer sur « Parcourir » et sélectionner votre sauvegarde au format SQL et ensuite cliquer sur Go et les données seront automatiquement restaurées.

I.5 Manipuler les données avec SQL

Requête 1 :

```
SELECT c.name AS CHALLENGE, AVG(r.distance) AS AVERAGE
FROM challenge c
INNER JOIN regatta r
ON c.id = r.challenge_id
GROUP BY c.name
```

Résultat :

CHALLENGE	AVERAGE
Challenge d'hiver	300
Challenge d'été	275

Requête 2 :

```

SELECT p.firstname AS FIRSTNAME,
       p.lastname AS LASTNAME,
       e.num_licence AS LICENCE
FROM crew cr
INNER JOIN entrant e
ON e.id = cr.entrant_id
INNER JOIN person p
ON p.id = e.person_id
INNER JOIN compete co
ON co.id = cr.compete_id
WHERE co.regatta_id = 3

```

Résultat :

FIRSTNAME	LASTNAME	LICENCE
Marcel	Martin	105451
Pierre	Letroc	105452
Roger	Le Jay	105453
Hubert	de La Montagne	105454
Etienne	LE Baux	105455
Emilie	du Port	105456
Corentin	Quintin	105457
Clarisse	Le Fort	105458
Hélène	De Carquefou	105459
Alex	Thomson	105466
Louis	Burton	105471
Camille	Gicquel	105460
Jérémie	Beyou	105467
Nándor	Fa	105472
René	La Poste	105461
Jean-Pierre	Dick	105468
Éric	Bellion	105473
Rémy	Bellamy	105462
Yann	Eliès	105469
Amaud	Boissières	105474
Hector	Le Bel	105463
Armel	Le Cléac'h	105465
Jean	Le Cam	105470

Requête 3 :

```

SELECT r.name AS REGATTA,
       r.date AS DATE,
       pe.firstname AS AUDITOR_FIRSTNAME,
       pe.lastname AS AUDITOR_LASTNAME,
       c.name AS COMMITTEE
FROM panel pa
INNER JOIN auditor a
ON a.id = pa.auditor_id
INNER JOIN committee c
ON c.id = a.committee_id
INNER JOIN person pe
ON pe.id = a.person_id
INNER JOIN regatta r
ON r.id = pa.regatta_id
WHERE r.date > '2017-02-01'

```

Résultat :

REGATTA	DATE	AUDITOR_FIRSTNAME	AUDITOR_LASTNAME	COMMITTEE
Les régates de la baie de Saint Brieuc	2017-02-15 09:00:00	Marcel	le Hegarat	Bretagne
La baie en fête	2017-02-22 00:00:00	Marcel	le Hegarat	Bretagne

Requêtes : <https://github.com/malahx/AFPA-PhysicalRegatta/blob/master/BDD/query.sql>

I.6 Programmer dans le langage du SGBD

Fonction :

```

CREATE DEFINER=`root`@`localhost`
FUNCTION `newCode`(challenge_id INT, regatta_date DATETIME)
RETURNS varchar(10) CHARSET latin1
BEGIN
    DECLARE chalCode VARCHAR(5);
    DECLARE numRegatta INT(4) DEFAULT 0;
    SELECT code INTO chalCode
    FROM challenge c
    WHERE c.id = challenge_id;
    SELECT COUNT(r.id)+1 INTO numRegatta
    FROM regatta r
    INNER JOIN challenge c
    ON c.id = r.challenge_id
    WHERE c.id = 1
    GROUP BY c.id;

    RETURN CONCAT(chalCode, '-', MONTH(regatta_date), '-', numRegatta);
END

```

Résultat :

newCode(1, "2017-03-01")
CHHI-3-3

Fonctions : <https://github.com/malahx/AFPA-PhysicalRegatta/blob/master/BDD/fonctions.sql>

Déclencheurs :

De créations :

```
CREATE DEFINER=`root`@`localhost`
TRIGGER `physicalregatta`.`regatta_BEFORE_INSERT`
BEFORE INSERT ON `regatta`
FOR EACH ROW
BEGIN
    DECLARE chalBegin DATETIME;
    DECLARE chalEnd DATETIME;
    SELECT begin, end
        INTO chalBegin, chalEnd
        FROM challenge
        WHERE challenge.id = NEW.challenge_id;
    IF NEW.date < chalBegin OR NEW.date > chalEnd THEN
        SIGNAL SQLSTATE '09000' SET MESSAGE_TEXT = 'Wrong date regatta',
        MYSQL_ERRNO = 9000;
    END IF;
    SET NEW.code = newCode(NEW.challenge_id, NEW.date);
END
```

De mise à jour :

```
CREATE DEFINER=`root`@`localhost`
TRIGGER `physicalregatta`.`compete_BEFORE_UPDATE`
BEFORE UPDATE ON `compete`
FOR EACH ROW
BEGIN
    DECLARE num_sailboat INT(4) DEFAULT 0;
    DECLARE class_id_sailboat INT(11);
    SELECT class_id
        INTO class_id_sailboat
        FROM sailboat s
        WHERE s.id = NEW.sailboat_id;
    SELECT COUNT(co.id)
        INTO num_sailboat
        FROM compete co
        INNER JOIN sailboat s
        ON s.id = co.sailboat_id
        INNER JOIN sbclass cl
        ON cl.id = s.class_id
        WHERE co.regatta_id = NEW.regatta_id
        AND s.class_id = class_id_sailboat
        GROUP BY co.regatta_id;
    IF num_sailboat < NEW.position THEN
        SIGNAL SQLSTATE '09001' SET MESSAGE_TEXT = 'Wrong position', MYSQL_ERRNO
        = 9001;
    END IF;
END
```


De suppression :

```

CREATE DEFINER=`root`@`localhost`
TRIGGER `physicalregatta`.`regatta_BEFORE_DELETE`
BEFORE DELETE ON `regatta`
FOR EACH ROW
BEGIN
    DECLARE chalEnd DATETIME;
    SELECT end
        INTO chalEnd
        FROM challenge
        WHERE challenge.id = OLD.challenge_id;
    IF now() < chalEnd THEN
        SIGNAL SQLSTATE '09002' SET MESSAGE_TEXT = 'Can not delete a not
finished challenge', MYSQL_ERRNO = 9002;
    END IF;
END

```

Déclencheurs : <https://github.com/malahx/AFPA-PhysicalRegatta/blob/master/BDD/triggers.sql>

Procédures stockées :**Procédure stockée 1 :**

```

CREATE DEFINER=`root`@`localhost`
PROCEDURE `average`(IN challenge_id INT)
BEGIN
    SELECT c.name AS CHALLENGE, AVG(r.distance) AS AVERAGE
    FROM challenge c
    INNER JOIN regatta r
    ON c.id = r.challenge_id
    WHERE c.id = challenge_id
    GROUP BY c.name;
END

```

Résultat : **CALL** average(1)

CHALLENGE	AVERAGE
Challenge d'hiver	300

Procédure stockée 2 :

```

CREATE DEFINER=`root`@`localhost`
PROCEDURE `crews`(IN regate_id INT, IN sailboat_id INT)
BEGIN
    SELECT p.firstname AS FIRSTNAME,
           p.lastname AS LASTNAME
    FROM crew cr
    INNER JOIN compete co
    ON co.id = cr.compete_id
    INNER JOIN entrant e
    ON e.id = cr.entrant_id
    INNER JOIN person p
    ON p.id = e.person_id
    WHERE co.sailboat_id = sailboat_id
    AND co.regatta_id = regate_id;
END

```

Résultat : **CALL** crews(3, 13)

FIRSTNAME	LASTNAME
Hector	Le Bel
Armel	Le Cléac'h
Jean	Le Cam

Procédure stockée 3 :

```

CREATE DEFINER=`root`@`localhost`
PROCEDURE `getAuditorFrom`(IN challenge_id INT, IN date_début DATETIME, IN
date_fin DATETIME)
BEGIN
    SELECT r.date AS DATE,
           pe.firstname AS AUDITOR_FIRSTNAME,
           pe.lastname AS AUDITOR_LASTNAME,
           co.name AS COMMITTEE
    FROM panel pa
    INNER JOIN regatta r
    ON r.id = pa.regatta_id
    INNER JOIN auditor a
    ON a.id = pa.auditor_id
    INNER JOIN person pe
    ON pe.id = a.person_id
    INNER JOIN committee co
    ON co.id = a.committee_id
    INNER JOIN challenge ch
    ON ch.id = r.challenge_id
    WHERE ch.id = challenge_id
    AND r.date > date_début
    AND r.date < date_fin;
END

```

Résultat : **CALL** getAuditorFrom(1, "2017-02-02", "2017-03-03")

DATE	AUDITOR_FIRSTNAME	AUDITOR_LASTNAME	COMMITTEE
2017-02-15 09:00:00	Marcel	le Hegarat	Bretagne
2017-02-22 00:00:00	Marcel	le Hegarat	Bretagne

II. MODULE 1 – DÉVELOPPER L'INTERFACE D'UNE APPLICATION INFORMATIQUE

II.1 Écrire un algorithme

A des fins de test, j'ai fait cette exercice avec deux solutions possible, une version normal et une version avec des expressions régulières.

Code :

```
/**
 *
 * @param email
 * @return true si le paramètre est un email
 */
public static boolean isEmail(String email) {
    // Décomposition des @
    String[] e = email.split("@");

    // Vérification du nombre de @
    if (e.length == 2) {

        // Inverser l'host (nécessaire pour limiter le nombre de split)
        String revHost = new StringBuilder(e[1]).reverse().toString();

        // Décomposition des points
        String[] f = revHost.split("\\.", 2);

        // Test que toutes les longueurs soient respectée
        return e[0].length() > 1
            && f.length > 1
            && f[0].length() > 1
            && f[1].length() > 1;
    }
    return false;
}

/**
 *
 * @param email
 * @return true si le paramètre est un email
 */
public static boolean isEmailReg(String email) {
    // Vérification du pattern
    return Pattern.matches("^([@]{2,}?@[^@]{2,}\\.[^@.]{2,}$", email);
}
```

Jeu de test :

```

@Test
public void test_isEmail() {
    assertTrue(Algo.isEmail("ab@cd.ef"));
    assertTrue(Algo.isEmail("abcdef@ghij.klmn"));
    assertTrue(Algo.isEmail("ab@c.f.ef"));
    assertFalse(Algo.isEmail("a@ghij.klmn"));
    assertFalse(Algo.isEmail("ab@c.de"));
    assertFalse(Algo.isEmail("ab@cd.e"));
    assertFalse(Algo.isEmail("abcdefghij"));
    assertFalse(Algo.isEmail("abcdefghij.kl"));
    assertFalse(Algo.isEmail("ab@cdef"));
}

@Test
public void test_isEmailReg() {
    assertTrue(Algo.isEmailReg("ab@cd.ef"));
    assertTrue(Algo.isEmailReg("abcdef@ghij.klmn"));
    assertTrue(Algo.isEmailReg("ab@c.f.ef"));
    assertFalse(Algo.isEmailReg("a@ghij.klmn"));
    assertFalse(Algo.isEmailReg("ab@c.de"));
    assertFalse(Algo.isEmailReg("ab@cd.e"));
    assertFalse(Algo.isEmailReg("abcdefghij"));
    assertFalse(Algo.isEmailReg("abcdefghij.kl"));
    assertFalse(Algo.isEmailReg("ab@cdef"));
}

```

II.2 Développement objet :

Modèles :

Personne :

```

package afpa.ecf.objet;

import java.util.Calendar;

/**
 * @author gwenole
 */
public class Personne {
    String nom, prenom, email;
    int anneeNaissance;
    /**
     * @param nom le nom de la personne

```

```

    * @param prenom le prénom de la personne
    * @param email l'email de la personne
    * @param anneeNaissance l'année de naissance de la personne
    */
    public Personne(String nom, String prenom, String email,
                    int anneeNaissance) {
        super();
        this.nom = nom;
        this.prenom = prenom;
        this.email = email;
        this.anneeNaissance = anneeNaissance;
    }

    /**
     *
     * @return l'age de la personne en fonction de l'année courante
     */
    public int getAge() {
        return Calendar.getInstance().get(Calendar.YEAR) - anneeNaissance;
    }

    /**
     *
     * @return la personne en format String
     */
    @Override
    public String toString() {
        return "Personne [nom=" + nom + ", prenom=" + prenom + ", email=" +
            email + ", age=" + getAge() + "]";
    }
}

```

Propriétaire :

```

package afpa.ecf.objet;

/**
 *
 * @author gwenole
 */
public class Proprietaire extends Personne {

    /**
     *
     * @param nom le nom du propriétaire
     * @param prenom le prénom du propriétaire
     * @param email l'email du propriétaire
     * @param anneeNaissance l'année de naissance du propriétaire
     */
    public Proprietaire(String nom, String prenom, String email,
                        int anneeNaissance) {
        super(nom, prenom, email, anneeNaissance);
    }

    /**
     *
     * @return le propriétaire au format String
     */
}

```

```

    */
    @Override
    public String toString() {
        return "Proprietaire [" + super.toString() + "];"
    }
}

```

Licencié :

```

package afpa.ecf.objet;

import java.util.Calendar;

/**
 *
 * @author gwenole
 */
public class Licencie extends Personne {

    int numLicence, anneeLicence;
    double pointsFFV;

    /**
     *
     * @param nom le nom du licencié
     * @param prenom le prénom du licencié
     * @param email l'email du licencié
     * @param anneeNaissance l'année de naissance du licencié
     * @param numLicence le numéro de licence du licencié
     * @param anneeLicence l'année de licence du licencié
     * @param pointsFFV le nombre de points FFV du licencié
     */
    public Licencie(String nom, String prenom, String email, int anneeNaissance,
                    int numLicence, int anneeLicence, double pointsFFV) {
        super(nom, prenom, email, anneeNaissance);
        this.numLicence = numLicence;
        this.anneeLicence = anneeLicence;
        this.pointsFFV = pointsFFV;
    }

    /**
     *
     * @param nombre le nombre de points à ajouter
     * @param cal la date ou il faut ajouter le nombre de points
     * @throws Exception
     */
    public void calculPoints(int nombre, Calendar cal) throws Exception {
        if (anneeLicence != cal.get(Calendar.YEAR)) {
            System.out.println(cal.get(Calendar.YEAR));
            throw new Exception();
        }
        pointsFFV += nombre;
    }

    /**
     *
     */
}

```

```

    * @return le licencié au format String
    */
    @Override
    public String toString() {
        return "Licencie [numLicence=" + numLicence + ", anneeLicence=" +
            anneeLicence + ", pointsFFV=" + pointsFFV + ", " +
            super.toString() + "];"
    }
}

```

Commissaire :

```

package afpa.ecf.objet;

/**
 *
 * @author gwenole
 */
public class Commissaire extends Personne {

    String commite;

    /**
     *
     * @param nom le nom du commissaire
     * @param prenom le prénom du commissaire
     * @param email l'email du commissaire
     * @param anneeNaissance l'année de naissance du commissaire
     * @param commite le comité du commissaire
     */
    public Commissaire(String nom, String prenom, String email,
        int anneeNaissance, String commite) {
        super(nom, prenom, email, anneeNaissance);
        this.commite = commite;
    }

    /**
     *
     * @return le commissaire au format String
     */
    @Override
    public String toString() {
        return "Commissaire [commite=" + commite + ", " +
            super.toString() + "];"
    }
}

```

Main :

```
package afpa.ecf.objet;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.List;

/**
 *
 * @author gwenole
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        // Instanciation du jeu d'essai
        Proprietaire pro1 = new Proprietaire("Duchou", "Jean", "jean@duchou.fr",
                                             1965);
        Proprietaire pro2 = new Proprietaire("Duduche", "Fred",
                                             "fred@duduche.fr", 1955);
        Proprietaire pro3 = new Proprietaire("Kikoo", "Pierre",
                                             "pierre@kikoo.fr", 1945);
        Licencie lic1 = new Licencie("Le Cléac'h", "Armel",
                                     "armel@lechleach.fr", 1977, 123456,
                                     2017, 0);
        Licencie lic2 = new Licencie("Thomson", "Alex", "alex@thomson.uk", 1974,
                                     123457, 2017, 0);
        Licencie lic3 = new Licencie("Beyou", "Jérémie", "jeremie@beyou.fr",
                                     1976, 123458, 2017, 0);
        Commissaire com = new Commissaire("Squarcini", "Bernard",
                                           "bernard@squarcini.fr", 1955,
                                           "Corse");

        // Préparation des dates du calculPoints
        Calendar cal1 = Calendar.getInstance();
        cal1.set(2017, 01, 01);

        Calendar cal2 = Calendar.getInstance();
        cal2.set(1917, 01, 01);

        // Test de la méthode calculPoints avec la bonne date
        try {
            lic1.calculPoints(100, cal1);
            lic1.calculPoints(50, cal1);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



```

// Test de la méthode calculPoints avec la mauvaise date
try {
    lic1.calculPoints(100, cal2);
} catch (Exception e) {
    System.out.println(e);
}

// Ajout du jeu d'essai dans une collection
List<Personne> personnes = new ArrayList<>();
personnes.add(pro1);
personnes.add(pro2);
personnes.add(pro3);
personnes.add(lic1);
personnes.add(lic2);
personnes.add(lic3);
personnes.add(com);

// Affichage de la collection
for (Personne p : personnes) {
    System.out.println(p);
}

// Calcul de l'age moyen et de l'age médian
System.out.println("Age moyen : " + moyAge(personnes));
System.out.println("Age médian : " + medAge(personnes));
}

/**
 * *
 * *
 * @param personnes
 * @return l'age moyen
 */
public static double moyAge(List<Personne> personnes) {

    double ageTot = 0;

    for (Personne p : personnes) {
        ageTot += p.getAge();
    }

    return ageTot / personnes.size();
}

/**
 * *
 * *

```

```

* @param personnes
* @return l'age médian
*/
public static double medAge(List<Personne> personnes) {

    // Passage en tableau d'age
    int[] ages = new int[personnes.size()];

    for (int i = personnes.size() - 1; i >= 0; i--) {
        ages[i] = personnes.get(i).getAge();
    }

    // Triage des ages
    Arrays.sort(ages);

    // Récupération de l'index du centre
    int i = ages.length / 2;

    // Calcul de l'age médian
    if (ages.length % 2 == 1) {
        return ages[i];
    } else {
        return (ages[i - 1] + ages[i]) / 2.0;
    }
}
}

```

II.3 Développer l'interface graphique client/serveur

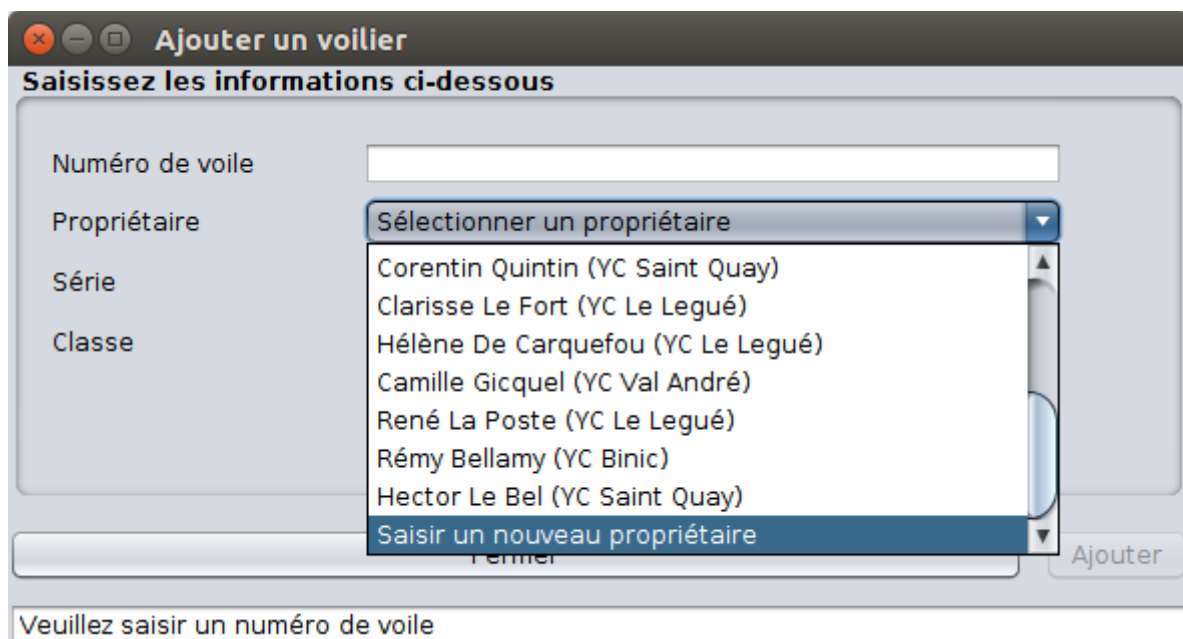
Maquette de l'ajout d'un voilier :

La maquette est une fenêtre graphique intitulée "Ajouter un voilier". Elle contient un titre "Saisissez les informations ci-dessous".

Numéro de voile	<input type="text"/>
Propriétaire	Sélectionner un propriétaire ▼
Série	Sélectionner une série ▼
Classe	Sélectionner une classe ▼

En bas de la fenêtre, il y a deux boutons : "Fermer" et "Ajouter".

En dessous de la fenêtre, il y a un message d'erreur : "Veuillez saisir un numéro de voile".

Maquette de l'ajout d'un propriétaire :

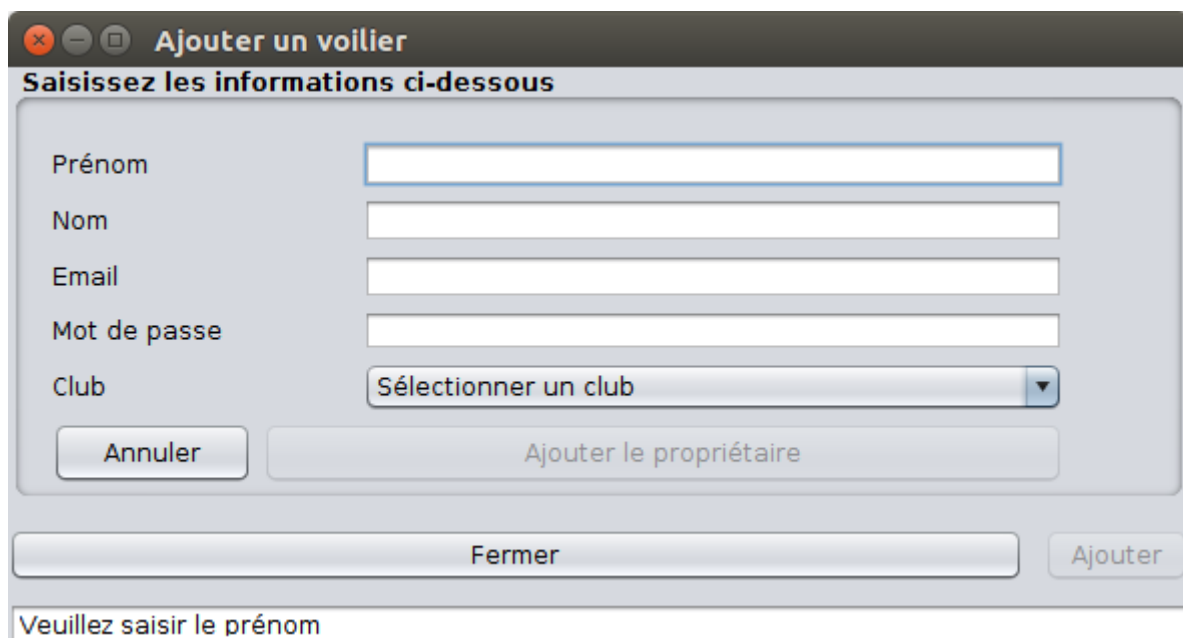
Maquette d'une fenêtre d'ajout d'un voilier. Le titre de la fenêtre est "Ajouter un voilier". Le sous-titre est "Saisissez les informations ci-dessous". Les champs de saisie sont :

- Numéro de voile : champ de saisie.
- Propriétaire : menu déroulant avec l'option "Sélectionner un propriétaire".
- Série : champ de saisie.
- Classe : champ de saisie.

Le menu déroulant "Propriétaire" est ouvert, montrant une liste de noms et de clubs :

- Corentin Quintin (YC Saint Quay)
- Clarisse Le Fort (YC Le Legué)
- Hélène De Carquefou (YC Le Legué)
- Camille Gicquel (YC Val André)
- René La Poste (YC Le Legué)
- Rémy Bellamy (YC Binic)
- Hector Le Bel (YC Saint Quay)
- Saisir un nouveau propriétaire

Les boutons "Ajouter" et "Fermer" sont présents. Une barre de message en bas indique : "Veuillez saisir un numéro de voile".



Maquette d'une fenêtre d'ajout d'un propriétaire. Le titre de la fenêtre est "Ajouter un voilier". Le sous-titre est "Saisissez les informations ci-dessous". Les champs de saisie sont :

- Prénom : champ de saisie.
- Nom : champ de saisie.
- Email : champ de saisie.
- Mot de passe : champ de saisie.
- Club : menu déroulant avec l'option "Sélectionner un club".

Les boutons "Annuler" et "Ajouter le propriétaire" sont présents. Les boutons "Fermer" et "Ajouter" sont également présents. Une barre de message en bas indique : "Veuillez saisir le prénom".

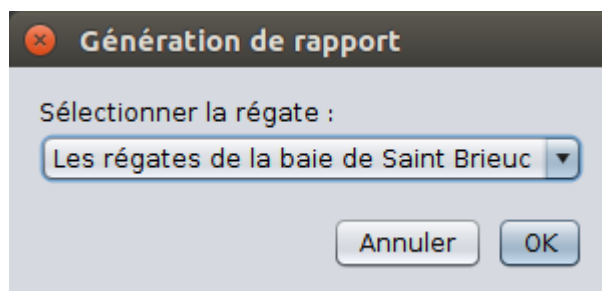
Dans un souci de clarté, le code a été hébergé sur GitHub :

<https://github.com/malahx/AFPA-PhysicalRegatta/tree/master/src/afpa/ecf/physicalregatta>

Packages :

- afpa.ecf.physicalregatta.view : Fenêtre JFrame Swing,
- afpa.ecf.physicalregatta.model : Modèle des objets utilisés,
- afpa.ecf.physicalregatta.Settings : Paramètres de configurations,
- afpa.ecf.physicalregatta.Utills : Fonctions génériques.

II.4 Mettre en œuvre un outil de génération d'état





Régate

LISTE DE DEPART

Challenge d'hiver, Les régates de la baie de Saint Brieuc Date : 2017-02-15

Toutes séries, toutes classes

Nom des commissaires :

Signature :

N°	CONCURRENT	N° VOILE	P	A/R
1	Marcel Martin	7		
2	Pierre Letroc	22		
3	Roger Le Jay	1478		
4	Hubert de La Montagne	199		
5	Etienne LE Baux	12		
6	Emilie du Port	13		
7	Corentin Quintin	1		
8	Clarisse Le Fort	25		
9	Hélène De Carquefou	14		
10	Camille Gicquel	2		
11	René La Poste	10		
12	Rémy Bellamy	23		
13	Hector Le Bel	24		

Code :

```
package afpa.ecf.physicalregatta.report;

import afpa.ecf.physicalregatta.Utills;
import afpa.ecf.physicalregatta.dao.ConnectDAO;
import afpa.ecf.physicalregatta.model.Regatta;
import afpa.ecf.physicalregatta.view.SailboatRegister;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.util.HashMap;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.Query;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.util.JRLoader;
import net.sf.jasperreports.swing.JRViewer;

/**
 *
 * @author gwenole
 */
public class StartList {

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

        }
        java.util.logging.Logger.getLogger(SailboatRegister.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

    }
        java.util.logging.Logger.getLogger(SailboatRegister.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    }
```

```

    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(SailboatRegister.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(SailboatRegister.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {

        Query qRegatta =
Utils.getEntityManager().createNamedQuery("Regatta.findAll");
        List<Regatta> regattas = qRegatta.getResultList();
        Regatta regatta = regattas.size() > 0 ? regattas.get(0) : null;

        Regatta r = (Regatta) JOptionPane.showInputDialog(
            null,
            "Sélectionner la régata :\n",
            "Génération de rapport",
            JOptionPane.PLAIN_MESSAGE,
            null,
            regattas.toArray(),
            regatta);

        if (r != null) {
            view(r.getId());
        }
    }
});

private static JasperPrint generate(int regatta_id) {
    try {
        HashMap<String, Object> param = new HashMap<>();
        param.put("regatta_id", regatta_id);
        JasperReport report = (JasperReport)
JRLoader.loadObject(StartList.class.getResource("start_list.jasper"));
        report.setProperty("net.sf.jasperreports.awt.ignore.missing.font",
"false");

report.setProperty("net.sf.jasperreports.default.font.name=SansSerif", "true");
        JasperPrint jPrint = JasperFillManager.fillReport(report, param,
ConnectDAO.Instance().get());
        return jPrint;
    } catch (JRException ex) {
        Logger.getLogger(StartList.class.getName()).log(Level.SEVERE, null,
ex);
    }
    return null;
}

public static void view(int regatta_id) {
    JasperPrint jPrint = generate(regatta_id);
    JFrame pdfFrame = new JFrame("Rapport");

```

```

pdfFrame.getContentPane().add(new JViewer(jPrint));
pdfFrame.pack();
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
pdfFrame.setSize((int) (screenSize.getWidth() / 3), (int)
(screenSize.getHeight() - 100));
pdfFrame.setLocationRelativeTo(null);
pdfFrame.setVisible(true);
    }
}

```

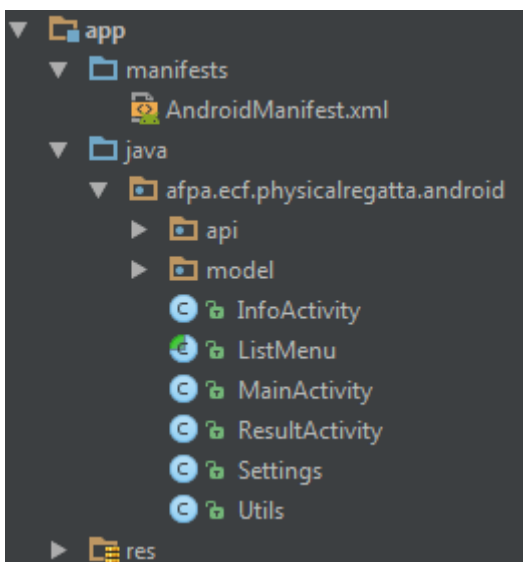
Génération d'état : <https://github.com/malahx/AFPA-PhysicalRegatta/blob/master/src/afpa/ecf/physicalregatta/report/StartList.java>

III. MODULE 5 – DÉVELOPPER UNE APPLICATION SIMPLE DE MOBILITÉ NUMÉRIQUE

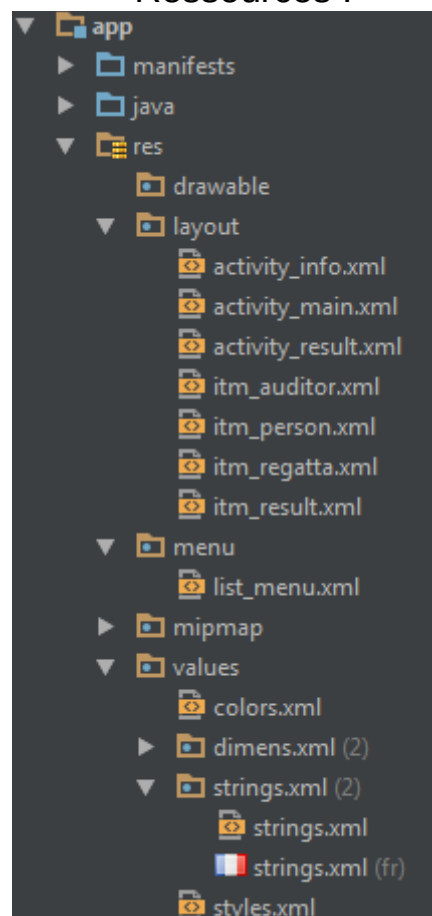
A des fins de tests, j'ai développé le serveur API REST de l'application mobile avec deux frameworks Jersey et Spark, pour la présentation, j'ai donc utilisé la version Spark.

III.1 Arborescences :

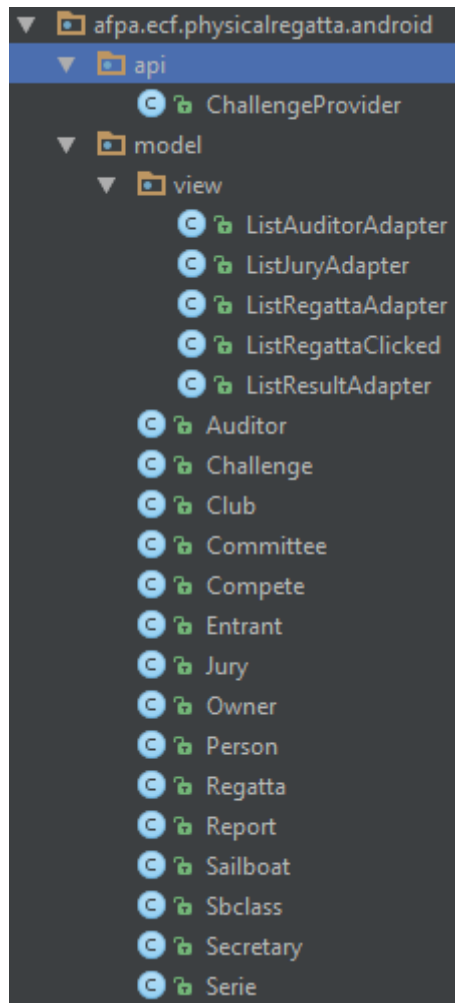
Activités :



Ressources :



Modèles :



III.2 Maquette de l'application :

Fenêtre principale affichant le challenge en cours avec la liste de régate à sélectionner

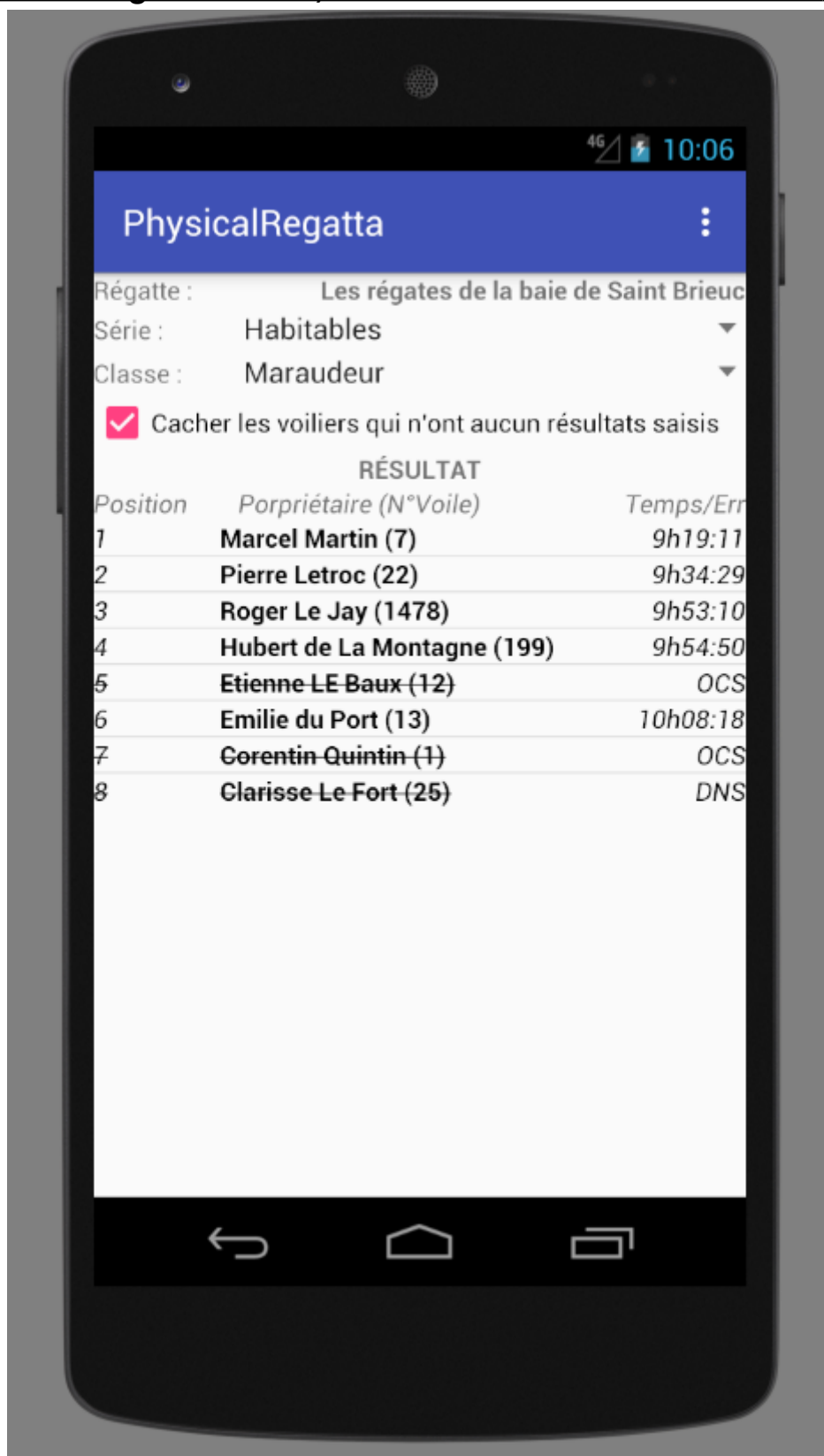


Le menu disponible :

La version localisé en anglais :

L'activité d'information d'une régate :

L'affichage des résultat non filtré :

Le résultat des régates filtrés, suivant une classe et une série :

Application mobile : <https://github.com/malahx/AFPA-PhysicalRegatta/tree/master/Android/app/src/main>