Projet base de données

Traitement des emplois du temps du département informatique

Enseignant responsable : Claude Sabatier

Lionel Dromer Éloi Perdereau

7 avril 2013

Table des matières

1 Cahier des charges

1.1 Étude de l'existant

Nous ne pouvons avoir qu'un point de vue de l'utilisateur final, mais en étudiant différents modèles d'emploi du temps, nous avons pu relever des fonctionnalités récurrentes, avec des subtilités :

- Deux sens possibles d'affichage des jours : vertical ou horizontal.
- Afficher l'emploi du temps de promotions, d'enseignants ou de salles. Possibilité de les combiner.
- Choisir la période.
- Différents formats d'exportation (PDF, impression, ...).
- Modification de l'emploi du temps par un administrateur.
- Couleurs par UE [1].
- Libelle d'un créneau adapté selon l'affichage.
- Version mobile disponible.

En voici deux exemples :

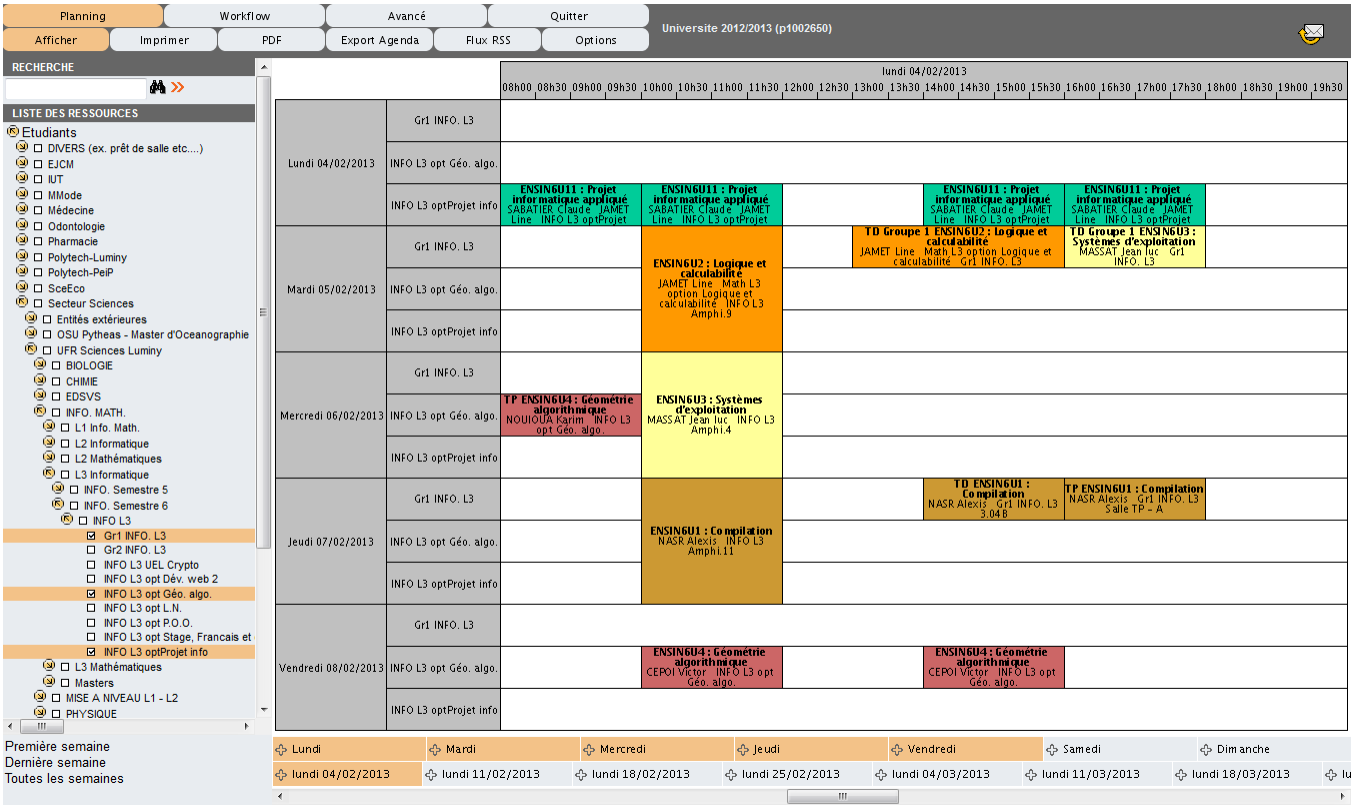


FIGURE 1 – ADE : Emploi du temps de l'AMU

1. UE = Union d'Enseignements

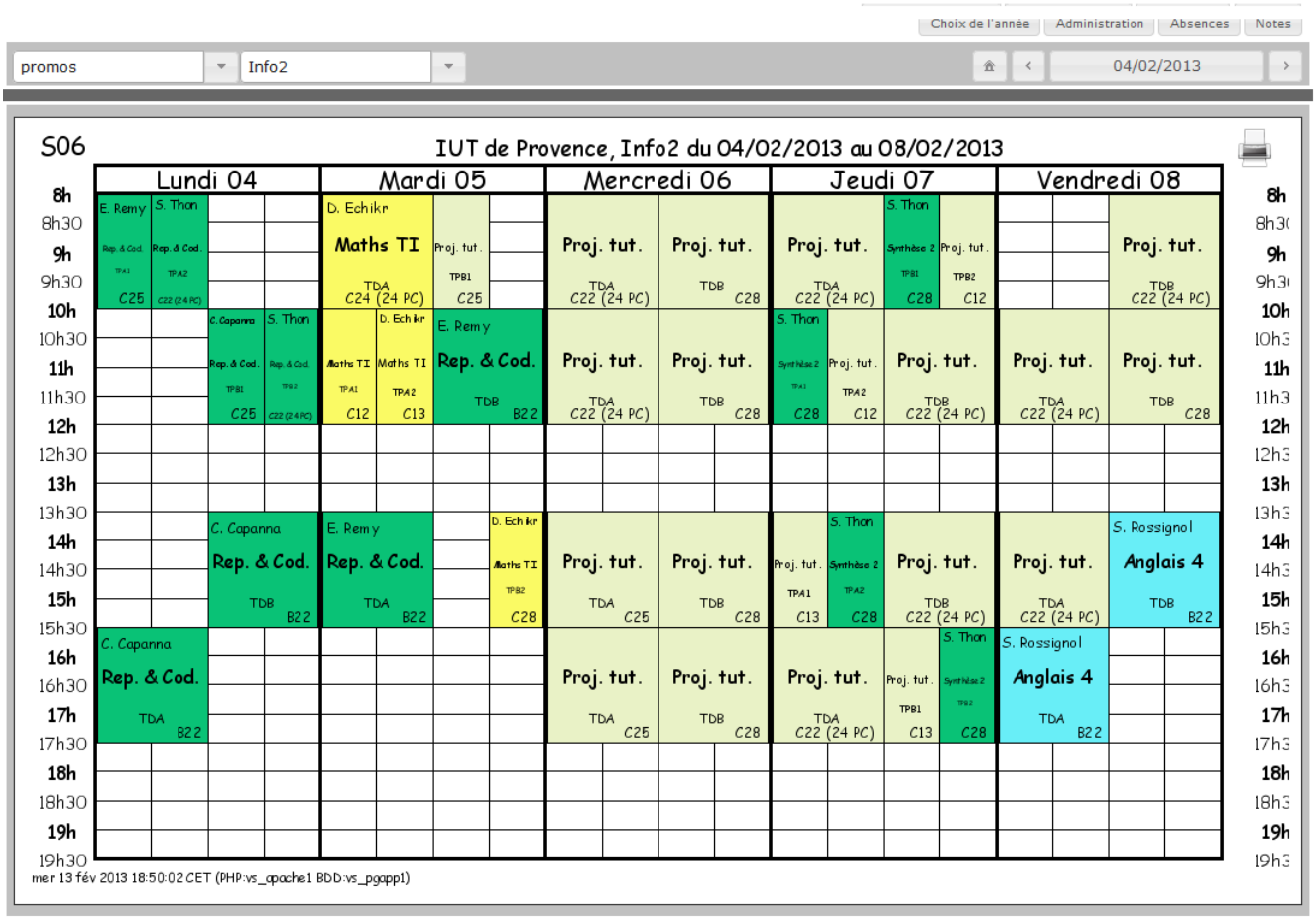


FIGURE 2 – Emploi du temps de l'IUT d'Arles

1.2 Étude des besoins

Afin de répondre à la problématique posée, on a pu relever les fonctionnalités suivantes :

- Vérifier la cohérence de toutes les horaires, au niveau des enseignants, des étudiants et des salles.
- Modification de l'emploi du temps : déplacer une séance. Vérifier la disponibilité des participants (étudiants, enseignant, salle) avant de l'enregistrer.
- Placement des séances selon un programme pédagogique donné.
- Éditer des récapitulatifs lisibles (couleurs et libellés adaptés) :
  - Emploi du temps d'une promotion donnée (tout groupes de TD confondus ou d'un groupe donné), pour une période donnée.
  - Emploi du temps d'un enseignant donné, pour une période donnée.
  - Occupation d'une salle donnée pour une période donnée.
  - Nombre total d'heures d'enseignement reçues par un étudiant d'une promotion donnée.
  - Nombre d'heures de Cours, TD et TP assurés par un enseignant donné (et les équivalents TD correspondant) pour toutes les UE et toutes les promotions.

1.3 Étude de faisabilité (outils)

Technologies à notre disposition :

- Langage Java
- JPA ou JDBC + DAO
- SGBD Oracle

Gestion de projet :

- Maven

– Gestionnaire de version Git
– Gestion de projet Gantt
– Équipe de deux développeurs

FIGURE 3 – Diagramme de Gantt

1.4 Analyse préalable

1.4.1 Scénarios

La structure pédagogique est fixe. Cela veux dire que les UE et leurs spécificités sont déjà présentes dans la base. Aucune interface n'est prévu pour modifier ces données. C'est à dire que seul un accès direct à la base de donnée permet la modification de la structure pédagogique.

Du point de vue de l'utilisateur final :
Il y a deux types d'utilisateurs finaux : les enseignants et le public. Le public ne peut que consulter l'emploi tu temps. Les enseignants, ont également le droit de consulter l'état de leur service. Certains enseignants peuvent être gestionnaire (cf. paragraphe suivant). Lorsqu'un utilisateur veut consulter l'emploi du temps, il doit d'abord dérouler un arbre dans lequel il sélectionne les différentes UE à afficher. Par salle, par année et groupe ou par enseignant.

Du point de vue des gestionnaires de l'emploi du temps :
Il y a plusieurs gestionnaires de l'emploi du temps ; ils travailleront de manière concurrente. On distingue en distingue deux types avec des droits spécifiques :
– Le gestionnaire d'année : il peut placer les séances auquel participerons les étudiants de l'année (des années) dont le gestionnaire à les droits.
– Le gestionnaire des professeurs : il peut modifier le nombre d'heures de services effectués par tous les enseignants, ainsi que leur ajouter une période d'indisponibilité.

Un gestionnaire peut faire les mêmes actions qu'un utilisateur classique. Lorsqu'il veut effectuer des tâches administratives, il doit se connecter via un bouton. Dès lors, une zone contenant différents boutons se présente à lui. Ces boutons affichent des panneaux dans lesquels le gestionnaire peut effectuer des tâches administratives.
– Pour un gestionnaire d'année(s) : La zone contient un bouton par année gérée. Chacun affiche un panneau contenant les cours/TD/TP/heures de projet restant à placer (pour l'année en question), avec le nombre d'heures associés. Le panneau contient également des champs à remplir pour placer une séance (date de début, durée). Lorsqu'il veux placer un cours, il sélectionne le cours en question dans le panneau. Une fois le cours sélectionné et les champs remplis, un bouton valider calcule si tous les participants sont disponibles (étudiants, enseignants, salle). Si oui, une salle est attribuée automatiquement et une boîte de dialogue s'affiche permettant au gestionnaire de choisir le professeur à affecter. Ensuite le créneau est placé dans l'emploi du temps. Si les étudiants ne sont pas disponibles où qu'aucune salle n'est libre, un message d'erreur averti le gestionnaire avec le motif de refus. Les champs précédemment remplis pourront être modifiés en conséquence. Un créneau est compris entre 8 heures et 20 heures du lundi au vendredi.

– Pour un gestionnaire d'enseignant : Le panneau se divise en deux avec une liste déroulante contenant la liste des enseignants en en-tête. Une partie lui montre un formulaire pour ajouter une période d'indisponibilité. Cet ajout peut être refusé si l'enseignant en question a déjà un cours de prévu dans le créneau indiqué. L'autre partie lui permet de modifier le nombre d'heures d'administrations effectués.

Un gestionnaire peut être à la fois gestionnaire de plusieurs années et gestionnaire d'enseignant en même temps.

1.4.2 Cas particuliers

– Avertissement lorsqu'une séance est placée dans un créneau peu commode (pause déjeuner, après 18 heures, etc...).

– Avertissement lorsqu'une séance est placée moins de 48 heures avant sa date de début.
– Avertissement si le nombre d'heures de TD, TP ou Projet est différent pour différents groupes d'une même promotion.

2 Analyse détaillée

2.1 Étude des données

2.1.1 Dictionnaire des données

Nom attribut	Description	Contrainte(s)
IdNiveau	Id du niveau d'une promotion (son année)	unique
LibelleNiveau	Libellé du niveau d'une promotion (e.g : 'L1', 'M2')	$\emptyset$
IdUE	Identifiant d'une UE	unique
LibelleUE	Libelle d'une UE	unique
NbHeuresCours	Nombre d'heures de cours concernant une UE	> 0
NbHeuresTD	Nombre d'heures de TD concernant une UE	> 0
NbHeuresTP	Nombre d'heures de TP concernant une UE	> 0
NbHeuresProjet	Nombre d'heures de Projet concernant une UE	≥ 0
IdPromo	Identifiant d'une promotion	unique
EffectifPromo	Effectif d'une promotion	> 0
IdGroupe	Identifiant d'un groupe de TD	unique
EffectifGroupe	Effectif d'un groupe	> 0
NomEns	Nom d'un enseignant	$\emptyset$
RattachementEns	Site de rattachement d'une enseignant	$\in$ {'Departement informatique', 'Exterieur'}
NoTelEns	Numéro de téléphone correspondant du bureau qu'un enseignant occupe	$\emptyset$
MailEns	Adresse e-mail d'un enseignant	$\emptyset$
NbHeuresAdminEns	Nombre d'heures qu'un enseignant a effectué	> 0
IdGrade	Identifiant d'un grade d'un enseignant	unique
LibelleGrade	Libelle d'un grade d'un enseignant	$\emptyset$
DateDebutIndisponibilite	Date de début d'une période d'indisponibilité pour un enseignant	$\emptyset$
DureeIndisponibilite	Durée d'une période d'indisponibilité pour un enseignant	> 0
IdSalle	Identifiant d'une salle	unique
NoSalle	Numéro d'une salle	$\emptyset$
EffectifMax	Effectif maximal d'une salle	> 0
IdTypeSalle	Identifiant d'un type de salle	unique
LibelleTypeSalle	Libelle d'un type de salle (e.g : 'Amphi', 'TD', 'TP').	$\emptyset$
IdTypeSeance	Identifiant du type de séance [2]	unique
LibelleTypeSeance	Libelle du type de séance (e.g : 'Cours', 'TD', 'TP', 'Projet')	$\emptyset$
CoefEquivalentTD	Coefficient d'équivalent TD d'un type de séance	> 0
DateDebutSeance	Date de début d'une séance	$\emptyset$
DureeSeance	Durée d'une séance	> 0 ; ≤ 4
IdGest	Identifiant numérique d'un gestionnaire	unique
LoginGest	Identifiant servant à un gestionnaire d'emploi du temps pour se connecter	$\emptyset$
PwGest	Mot de passe servant à un gestionnaire d'emploi du temps pour se connecter	$\emptyset$
IsGestEns	Indique si l'utilisateur est un gestionnaire d'enseignant ou non	$\emptyset$

2.1.2 Modèle Conceptuel des Données

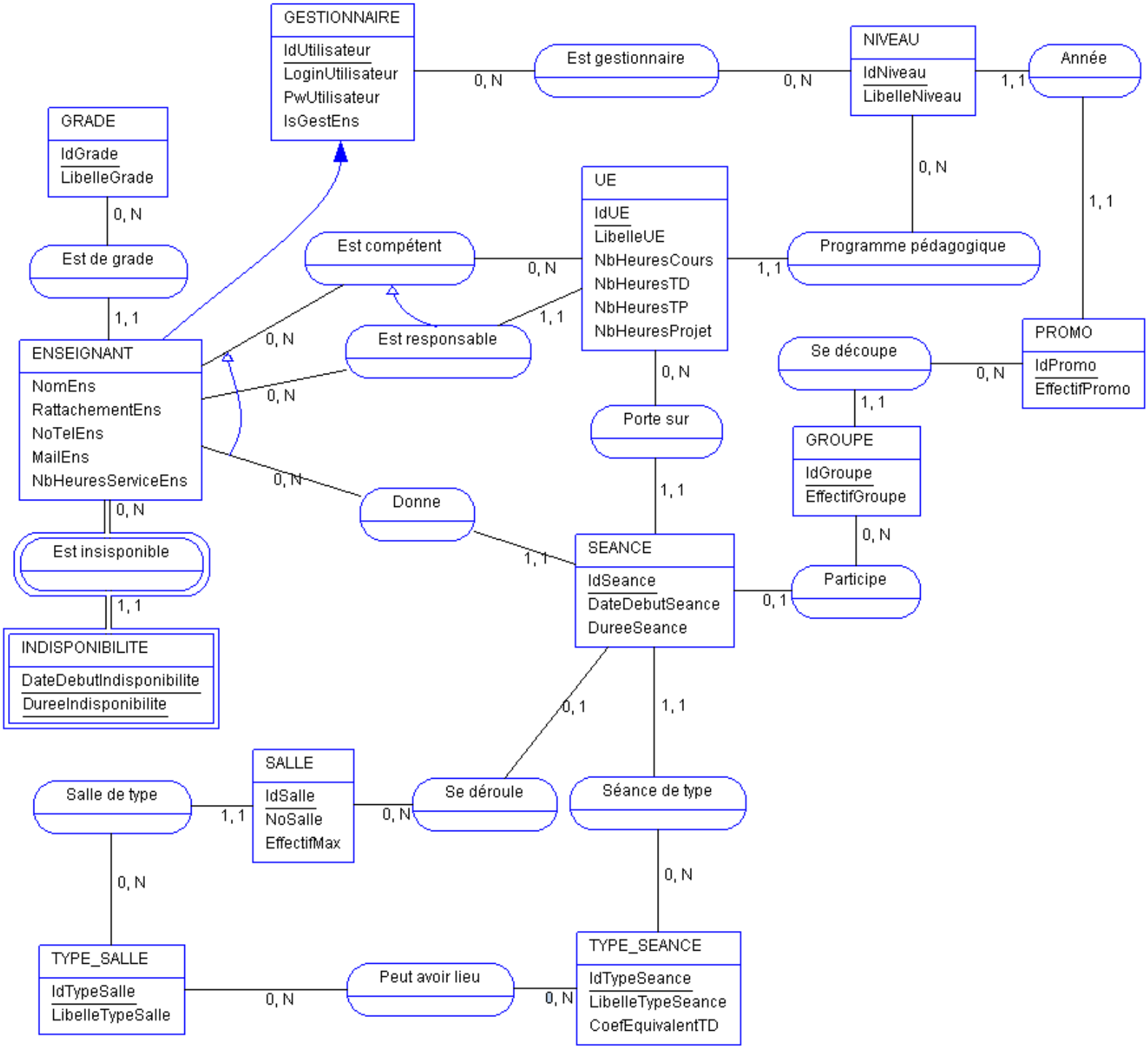


FIGURE 4 – MCD

On conviendra que si aucun groupe ne participe à une séance, alors c'est toute la promotion du niveau de l'UE qui y participe.

2.1.3 Contraintes

– Un enseignant ne peut participer à une séance uniquement s'il ne participe pas déjà à une séance dont la période chevauche celle de la séance en question. De plus, il ne doit pas être indisponible

durant la période de la séance.
- Une promotion ne peut participer à une séance uniquement si elle ne participe pas déjà à séance dont la période chevauche celle de la séance en question. Idem pour un groupe.
- Une salle ne peut être affecté à une séance uniquement si elle n'est pas déjà affecté à une séance dont la période chevauche celle de la séance en question.
- Une séance ne peut avoir lieu uniquement dans une salle qui a un effectif plus faible que l'effectif du groupe ou de la promotion qui participe à la séance.
- Une séance ne peut avoir lieu uniquement dans une salle dont le type est compatible à celui de la séance.
- Une séance ne peut pas durer plus de 4 heures.
- La somme des durées des séances pour une UE donnée et un type de séance donné ne doit pas dépasser le nombre d'heures prévu pour cette UE pour ce type de séance.
- Aucune séance ne doit avoir lieu avant 8 heures ou après 20 heures.
- La date de début d'une séance ajoutée ou modifiée doit être postérieur au présent. Idem pour la date de début d'une indisponibilité.
- L'effectif de l'ensemble des groupes d'une promotion doit être égal à l'effectif de la promotion concernée.
- Une promotion (ou un groupe) ne peut participer uniquement à des séances de son niveau.
- Un groupe ne peut pas participer à une séance de type 'Cours'.
- Une période d'indisponibilité ne peut être ajoutée (ou modifiée) si l'enseignant concerné participe à une séance chevauchant la période sus-dite.
- Si un individu est connecté et qu'il n'est pas gestionnaire, c'est un enseignant. En effet, seulement trois types d'individus pourront se connecter : les enseignants, les gestionnaires d'années et les gestionnaires d'enseignants. Malheureusement, cette contrainte ne pourra pas être vérifiée en base. Prenons un exemple : nous voulons ajouter un nouveau gestionnaire d'année. Il faut d'abord insérer le gestionnaire en question avec ses identifiants, puis insérer le (premier) lien qui le lie à une année. Entre ces deux insertions, la contrainte ne sera pas satisfaite. Néanmoins, nous pouvons empêcher un individu de se connecter s'il n'est ni gestionnaire ni enseignant.

2.2 Étude des fonctionnalités de l'application

2.2.1 Scénarios

2.2.2 Modèle Conceptuel des Traitements

3 Programmation et interfaces

3.1 Les DAO (Data Access Object)

3.2 Les rendus

Les conteneurs graphiques (`JPanel`, `JTable`, `JTree`, ...) contiennent par définition des objets qui seront disposés à la vue de l'utilisateur. Lorsqu'un conteneur est dessiné, les objets contenus sont eux même dessinés. Cela se fait principalement avec la méthode `paint()` pour des `JComponent`, ou pour des objets dits utilisateurs (*user object*), en mettant dans un "porteur" (souvent un `JLabel`) le résultat de l'appel à la méthode `toString()` (hérité de `Object`) sur l'objet à dessiner. Il suffirait donc de redéfinir cette méthode correctement pour chaque classes d'objet à afficher. Hors nos classes métiers ont déjà leur méthode `toString()` implémentée pour permettre la sérialisation. Cette méthode n'est donc plus adaptée pour l'affichage car non *user friendly*. Pour afficher les objets métiers correctement sans perdre leur aspect sérialisable, nous avons donc du utiliser un autre mécanisme. Bien qu'assez complexe à mettre en place, il permet au final une plus grande maniabilité.

La difficultée ici, c'est que pour un objet, par exemple un `JTree`, nous ne pouvons spécifier qu'un seul moteur de rendu (*renderer*) qui sera utilisé pour dessiner tous les objets (ce n'est pas le cas pour les `JTable`s car l'API nous oblige à spécifier la classe des objets lorsqu'on ajoute des moteurs de rendus [3]). C'est pour cela que la méthode `toString()` est très utilisée (car partagée par tous les objets).
Notre solution se base sur un moteur de rendu d'affichage délégué, basé sur la classe de l'objet (*Class-Based Display Delegation Renderer*). On a un objet qui fait office de moteur de rendu pour le conteneur. Lorsq'il est appelé, au lieu de créer directement un rendu, il délègue ce travail à un autre moteur de rendu selon un tableau associatif ayant pour clé la classe de l'objet à dessiner et pour valeur le moteur de rendu à appeler. Ainsi, peu importe la classe des objets à dessiner, nous pouvons utiliser le bon moteur de rendu (si aucun n'est trouvé, on utilise celui par défaut). Il faut bien entendu enregistrer ceux-ci avant tout affichage (c'est le rôle du présentateur).

Mais en pratique cette solution ne peux que très rarement être utilisée tel quel. En effet, les conteneurs graphiques requièrent des moteurs de rendus spécialisés. Par exemple, les `JComboBox`s ne peuvent qu'utiliser des rendus implémentant l'interface `ListCellRenderer`. La solution la plus simple est d'encapsuler un DDR [4] dans une classe dérivant de `DefaultListCellRenderer` qui implémente `ListCellRenderer` et dérive de `JLabel`. Lorsqu'un rendu est demandé, on appel d'abord la méthode `DefaultListCellRenderer` qui créera un rendu par défaut, puis on le modifie avec notre DDR.
Un autre problème se pose avec les `JTree`s : ce ne sont pas les objets métiers qui sont dessinés mais les noeuds (`TreeNode`s). La solution est similaire à celle du problème des moteurs de rendus spécialisés. Pour résumer, voici les états d'encapsulations des moteurs de rendus pour notre `JTree` : Le renderer de l'arbre créé un rendu par défaut et le modifie en appelant le renderer de noeud qui extrait l'objet métier et appel le DDR qui appel le bon renderer et renvoie le rendu tant désiré au premier renderer. Voici le diagramme de classes.

3. Nous pouvons toujours appliquer notre méthode en utilisant `Object.class`
4. DDR : Display Delegation Renderer

3.3 La table

Pour la représentation de l'emploi du temps lui-même, nous utilisons la classe fournie par `swing` : `JTable`. Ce faisant, plusieurs difficultées se sont posées :

- Il a fallu trouver les coordonnées d'une séance dans la table. C'est à dire convertir le jour de la semaine en index de ligne, et l'heure dans la journée en index de colonne. Avec des colonnes représentant deux heures, on peux utiliser la formule suivante :

$$numero\ colonne = heure - (7 + \frac{heure - 8}{2})$$

- Pour les séances durant plusieurs heures, nous voulions fusionner horizontalement les cellules correspondantes dans la table (*column spanning*). Cela n'est pas initialement prévu par la classe `JTable`. Mais celle-ci semble avoir été concue pour autoriser la distribution non-uniforme des cellules.
  Les méthodes intéressantes à redéfinir sont :
  `public Rectangle getCellRect(int row, int column, boolean includeSpacing);`
  Retourne un rectangle englobant la cellule aux positions donnés.
  `public int columnAtPoint(Point p);`
  `public int rowAtPoint(Point p);`
  Retournent respectivement la ligne et la colonne de la cellule contenant un point donné dans la table. Dans notre cas, nous n'auront que `columnAtPoint()` à redéfinir, car les cellules ne seront fusionner que horizontalement.
  Une fois ceci accompli, le `TableCellRenderer` s'occupe du rendu des cellules.
  Mais cela ne suffit pas. Contrairement à ce que l'on pourrais penser, beaucoups de composants `swing` ne se dessinent pas directement avec la méthodes `paint` mais délèguent le travail à un objet `ComponentUI` pour permettre une plus grande flexibilité. C'est le cas avec les `JTable`s. Il nous faut donc définir une classe qui va dessiner la table à notre convenance. Pour cela, nous dérivons la classe `BasicTableUI` et redéfinissons la méthode paint.
  Enfin, il nous faut une structure pour stocker les cellules qui seront fusionnées. Elle fera parti du modèle de notre table que nous redéfinirons car le simple `DefaultTableModel` ne suffit plus. Voici le diagramme de classes :

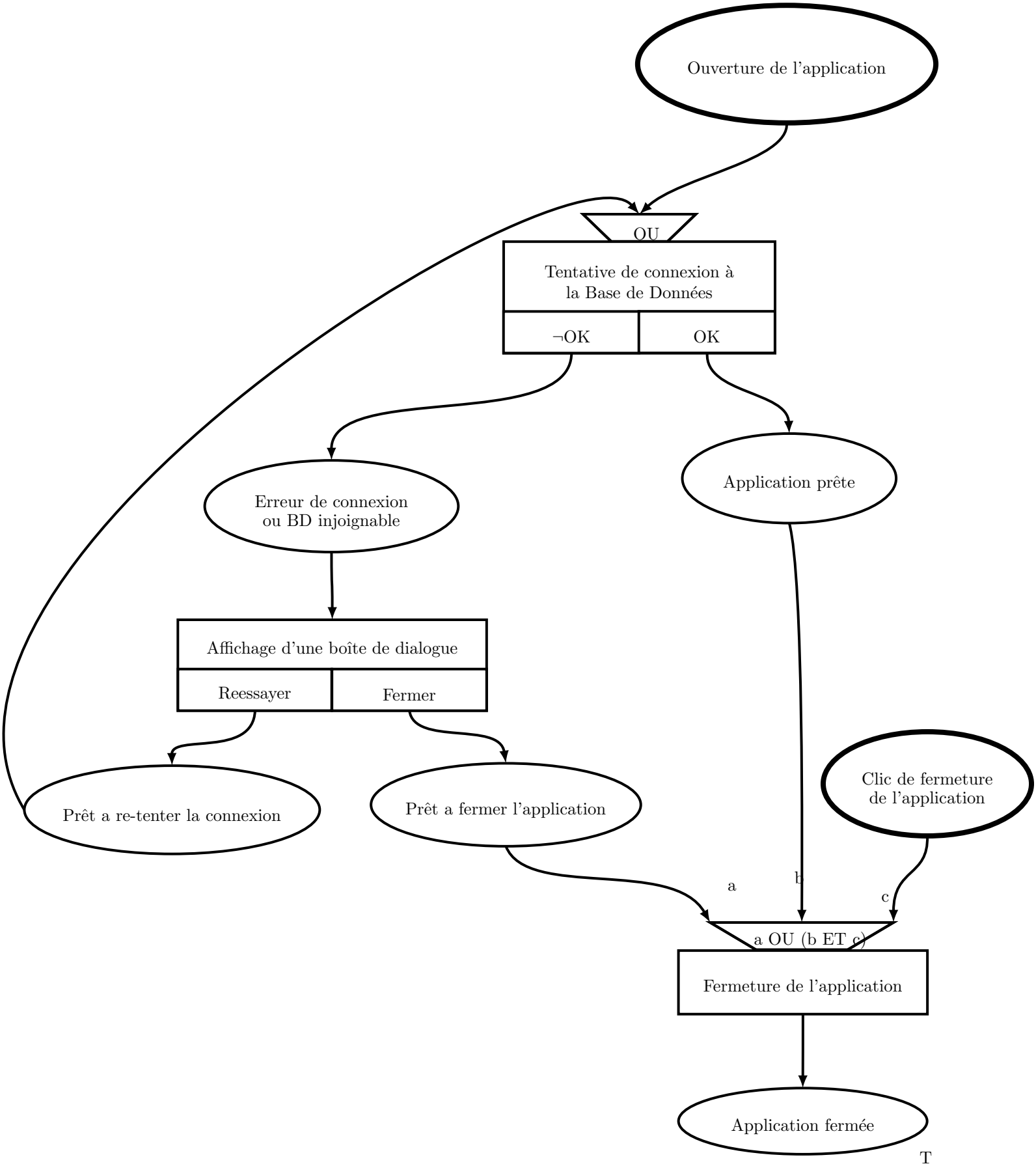
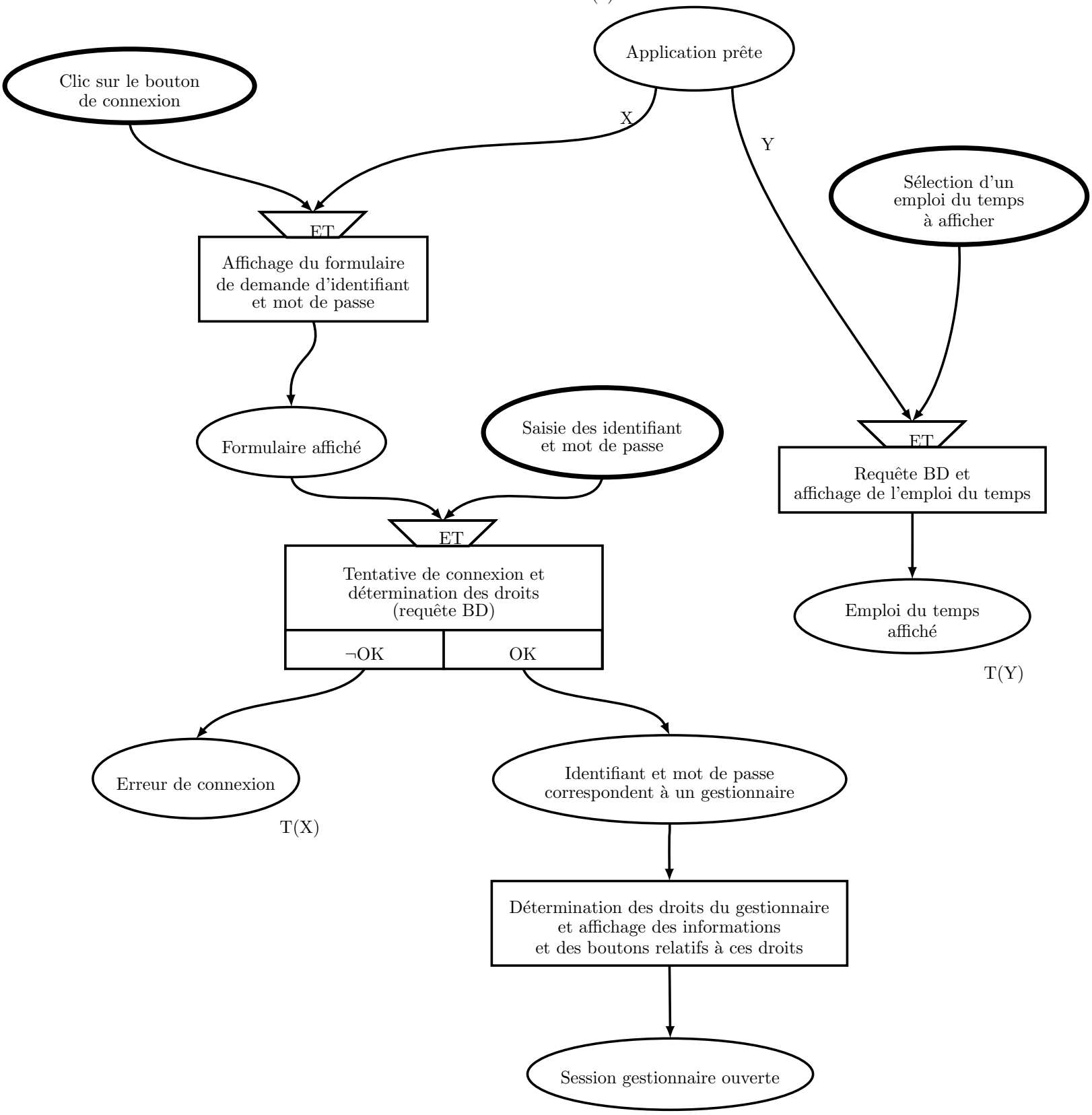


Figure 5 – MCT (1)

FIGURE 6 – MCT (2)



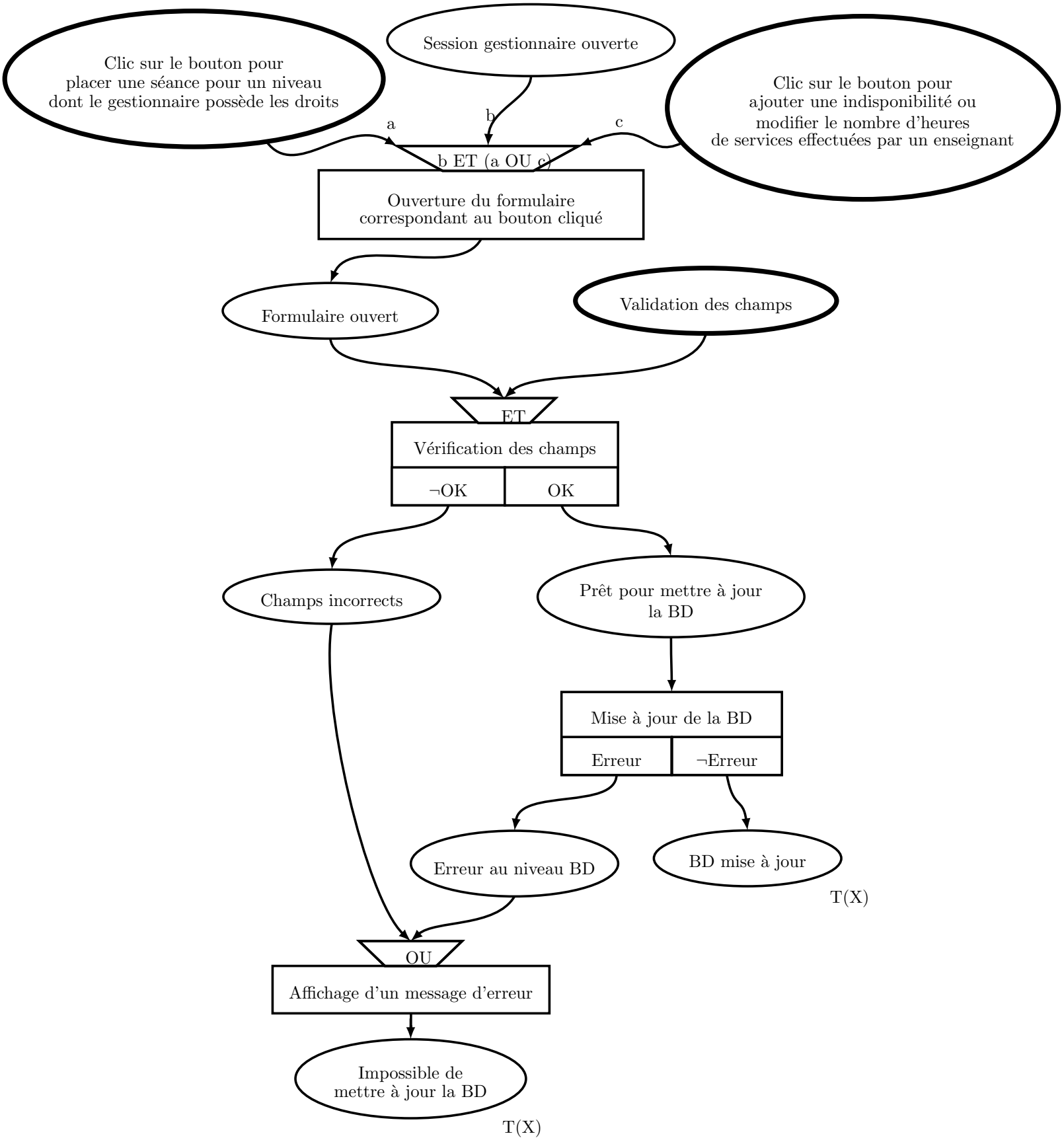


Figure 7 – MCT (3)

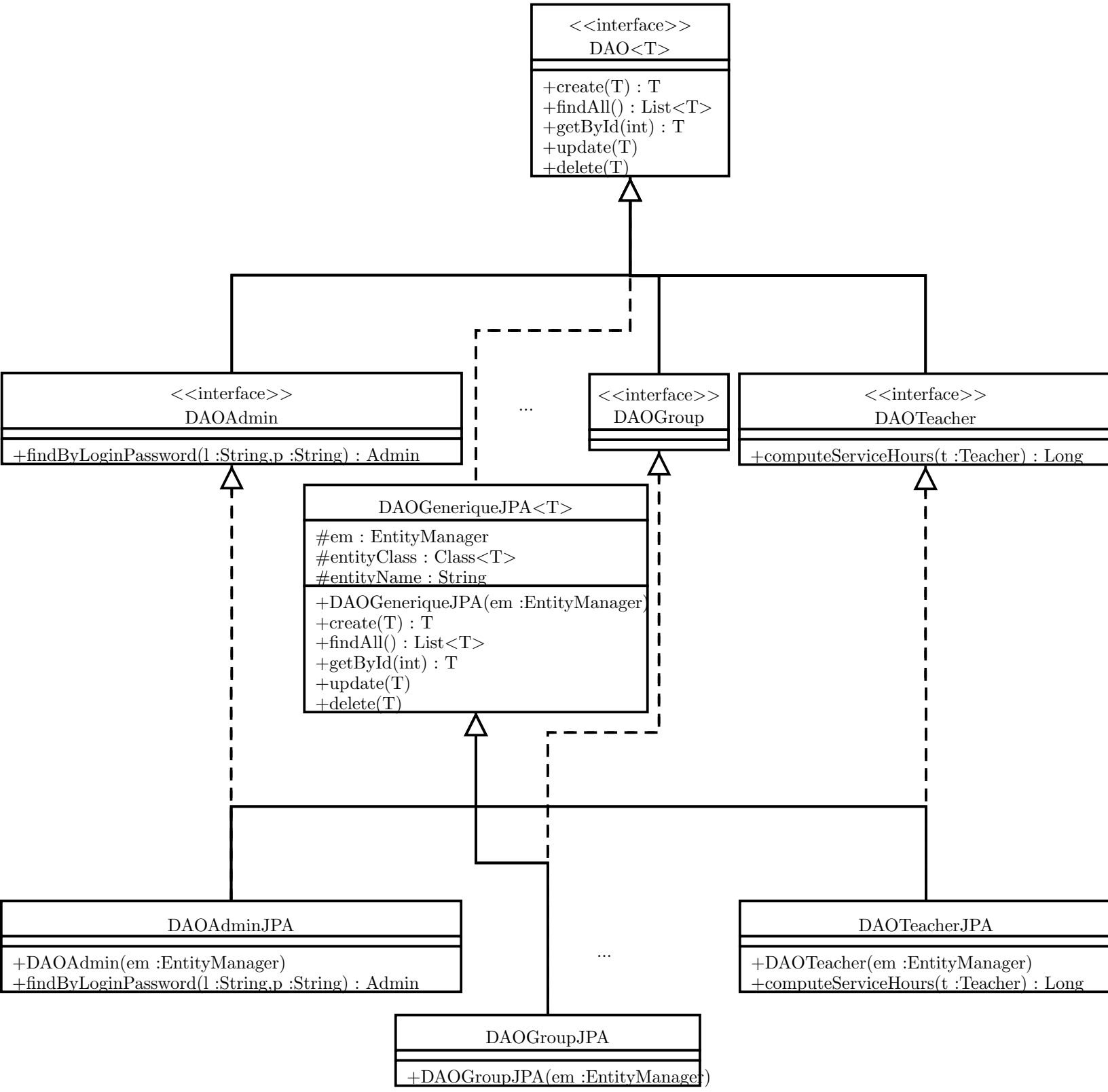


FIGURE 8 – Diagramme de classes des DAOs

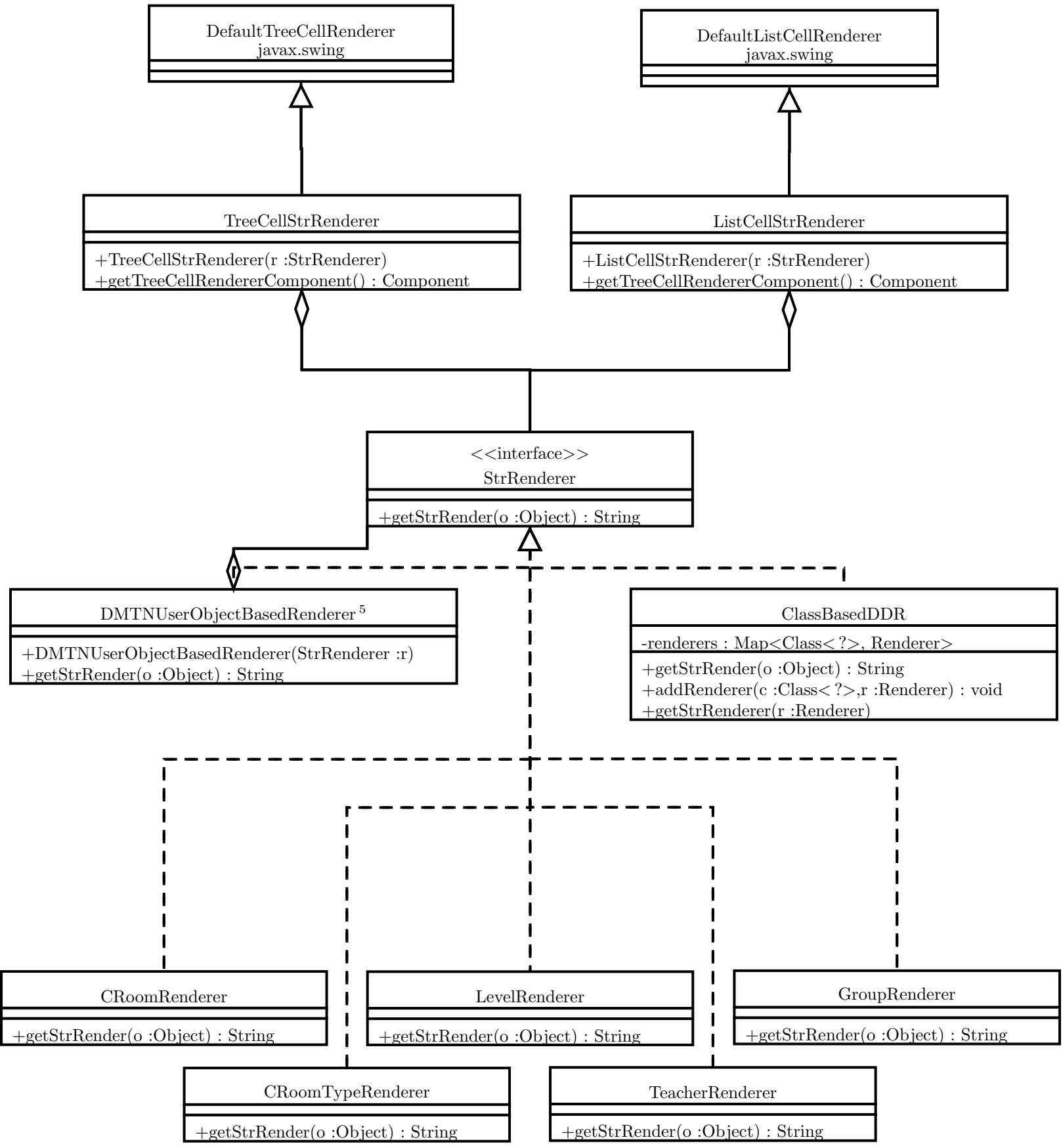


FIGURE 9 – Diagramme de classes des rendus

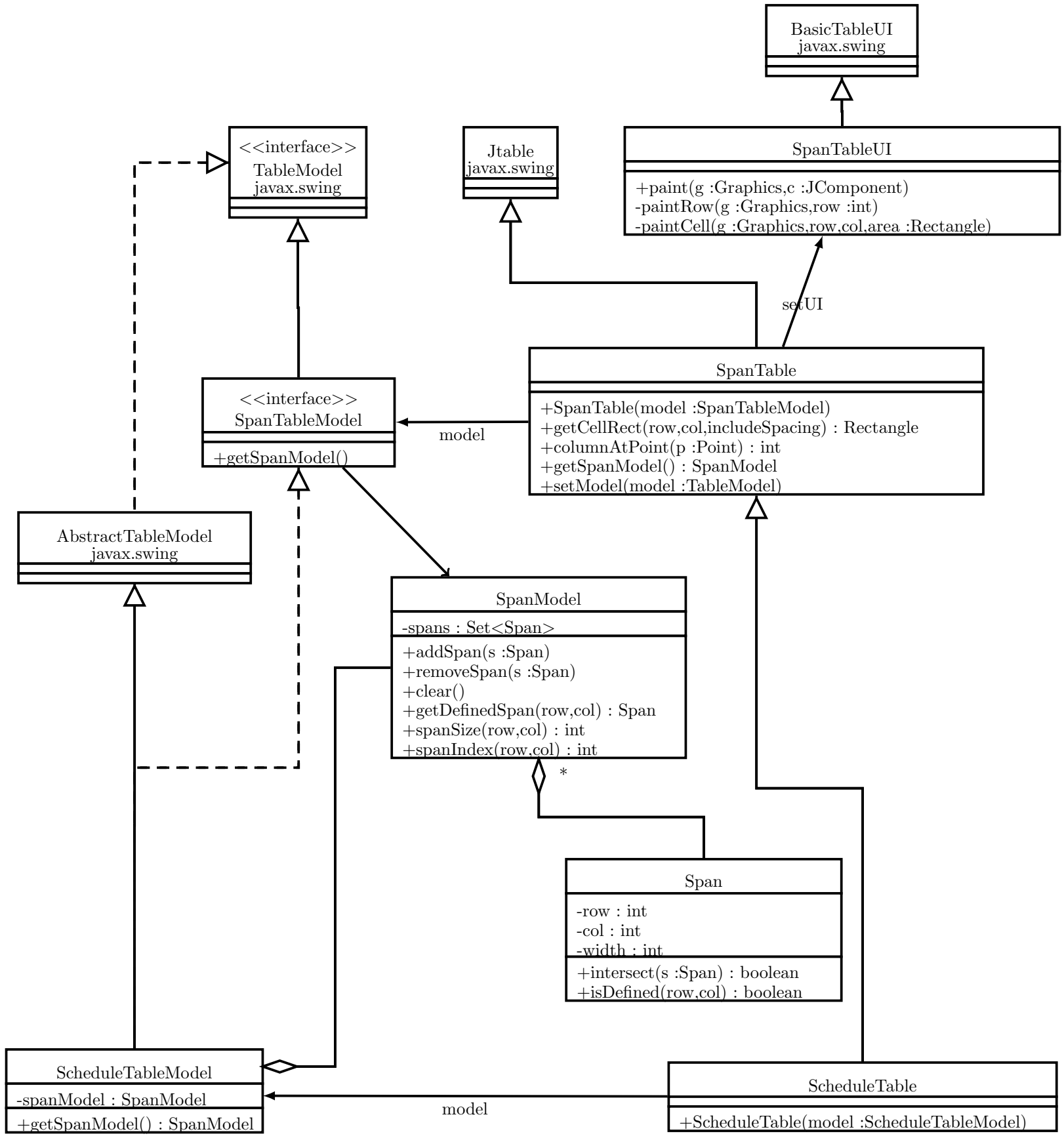


FIGURE 10 – Diagramme de classes de la table