Clement YOUDEC - Laure GAGNANT - Nicolas GUZZO Yohan PERROT
IMR2

Rapport JEE :

Application de gestion de congés

Introduction

L'objectif de ce rapport est de présenter l'application JEE que nous avons développée en nous appuyant sur la partie UML, réalisé plus tôt dans la période. Ce projet nous permettra de développer des compétences en JEE mais aussi de découvrir le serveur tomcat et le gestionnaire de base de données MySQL.

Pour présenter notre réalisation, nous commencerons par expliquer notre diagramme de classes, puis nous parlerons de notre modèle de données. Nous finirons par expliquer les implémentations techniques des différentes fonctionnalités.

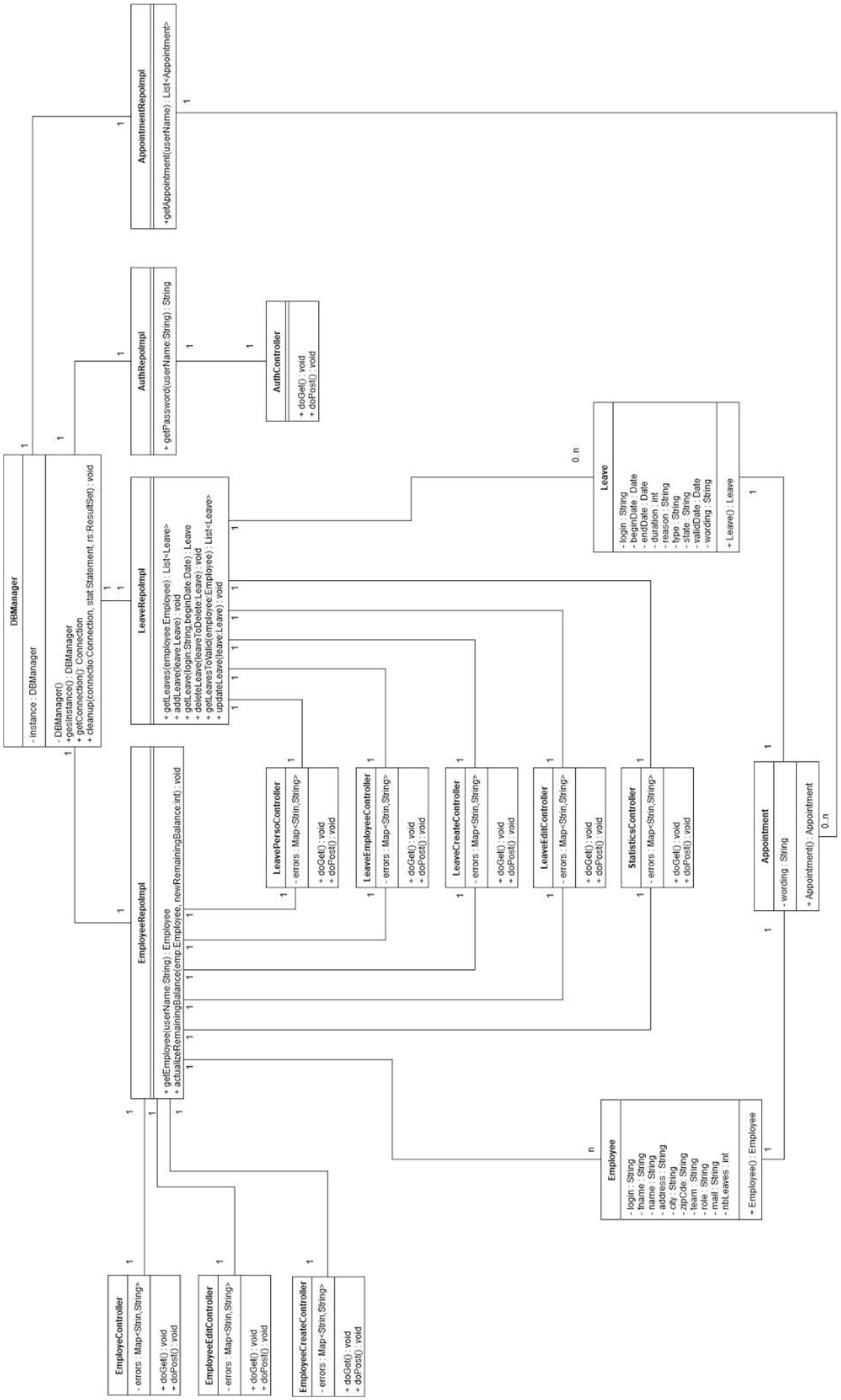
Diagramme de classes

Nous avons décidé d'utiliser le modèle MVC lors de la conception du projet car cela nous semblait naturel au vue de sa nature. En effet, nous utilisons tomcat (partie contrôleur) et mysql (partie modèle) pour développer une application WEB (partie vue) . Notre diagramme de classe est découpé en trois partie : une partie pour l'accès aux données , une partie pour la modélisation des données et une partie pour le traitement des données et la gestion des vues.

La partie accès aux données (DAO) doit permettre de relier l'application au système de gestion de base de données. Le lien avec de système doit être réalisé par la classe "DAOManager". Les classes "EmployeeRepoImp", "LeaveRepoImp", "AppointmentRepoImp" et "AuthRepoImp", appelant toutes "DAOManager" doivent permettrent l'accès à un type de données précis dans la BDD. Ces types sont respectivement : les employés, les congés, les rendez-vous et les informations de connections.

La partie modélisation des données est composé de trois classes : "Appointment", "Employee" et "Leave", qui représente respectivement les rendez-vous, les employés et les congés. Ces classes permettent de contenir et de rendre accessible les informations importantes liées aux donnés qu'elles représentent.

La dernière partie désigne les classes contrôleurs permettant le traitement des informations et la gestion des vues. Il existe une classe par vue à mettre en place. Ces classes sont les suivantes : "AuthController", "EmployeeController", "EmployeeCreateController", "EmployeeEditController", "LeaveCreateController", "LeaveEditController", "LeaveEmpController", "LeavePersoController", "StatisticsController".


DBManager
- instance : DBManager
- DBManager()
+ getInstance() : DBManager
+ getConnection() : Connection
+ Cleanup(connectio:Connection, stat:Statement, rs:ResultSet) : void
AppointmentRepoImpl
+getAppointment(userName) : List<Appointment>
AuthRepoImpl
+ getPassword(userName:String) : String
AuthController
+ doGet() : void
+ doPost() : void
LeaveRepoImpl
+ getLeaves(employee:Employee) : List<Leave>
+ addLeave(leave:Leave) : void
+ getLeave(login:String,beginDate:Date) : Leave
+ deleteLeave(leaveToDelete:Leave) : void
+ getLeavesToValid(employee:Employee) : List<Leave>
+ updateLeave(leave:Leave) : void
Leave
- login : String
- beginDate : Date
- endDate : Date
- duration : int
- reason : String
- type : String
- state : String
- validDate : Date
- wording : String
+ Leave() : Leave
EmployeeRepoImpl
+ getEmployee(userName:String) : Employee
+ actualizeRemainingBalance(emp:Employee, newRemainingBalance:int) : void
LeavePersoController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
LeaveEmployeeController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
LeaveCreateController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
LeaveEditController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
StatisticsController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
Appointment
- wording : String
+ Appointment() : Appointment
Employee
- login : String
- fname : String
- name : String
- address : String
- city : String
- zipCde : String
- team : String
- role : String
- mail : String
- nbLeaves : int
+ Employee() : Employee
EmployeController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
EmployeeEditController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void
EmployeeCreateController
- errors : Map<Strin,String>
+ doGet() : void
+ doPost() : void

Modèle de données

Les classes JAVA “Employee”, “Leave” et “Appointment” possèdent chacune une table au sein de la BDD, ce sont les tables “Employe”, “Conge” et “Rendez_vous”.

La table “Employe” contient toutes les informations pour identifier un utilisateur (login, prénom, nom, adresse, ville, code postal, equipe, fonction, mail, solde de congés) et a comme clef primaire “login”. L’équipe et la fonction d’un employé doivent exister dans les tables “Ref_Equipes” et “Ref_Fonction” qui contiennent l’ensemble des équipe et des fonctions existantes au sein de l’application. Ce sont les clefs étrangères.

La table “Conge” contient toutes les informations pour identifier un congé utilisateur (login, date de début, date de fin, durée, motif, type de congés, état, date de validation, commentaire) et a comme clef primaire “login” et “date_debut”. Le motif, le type et l’état d’un congé doivent exister dans les tables “Ref_Motif_Conges”, “Ref_Type_Conges” et “Ref_Etat_Conges” qui contiennent les motifs, les types et les états valides de congés.

La table “Rendez_vous”, permettant de stocker tous les rendez-vous des employés est composé du login de l’utilisateur concerné, de la date du rendez-vous et d’un libellé. Tous ces éléments font partis de la clef primaire.

La table “Authentification”, composée d’un login et d’un mot de passe, permet de stocker les informations de connection des utilisateurs.

Chaque table “Ref_…” contient une type d’information avec une liste de possibilités. Exemple avec “Ref_Type_Conges”. Un congé doit être soit un congé payé, soit un congé formation, soit un RTT, mais cette liste doit pouvoir évoluer. C’est pour cela que nous avons créé une table, plutôt que de mettre une contrainte “CHECK” sur l’élément de la table “Conge”.

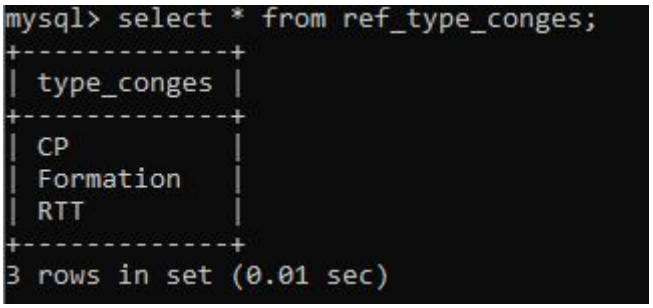
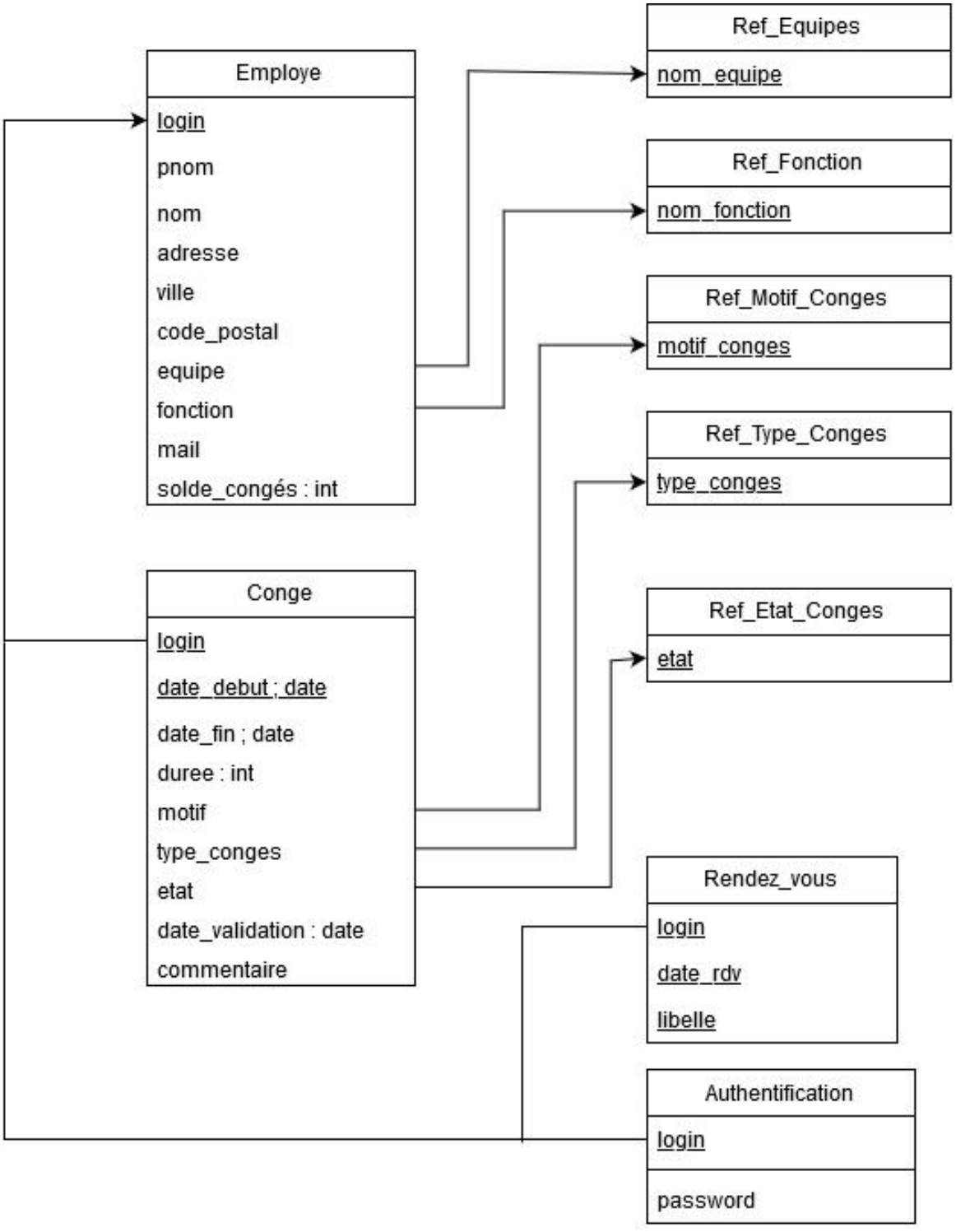


Schéma de BDD :



Implémentation technique

L'implémentation technique des différentes fonctionnalités s'est faite au sein des contrôleurs, chaque contrôleur traite uniquement les informations le concernant.

Le contrôleur "AuthController" permet l'authentification des utilisateurs. Lors de la réception d'un requête POST, les paramètres de la requête "inputUser" et "inputPassword" sont récupérés puis on vérifie qu'il existe une ligne dans la table "Authentification" contenant ces deux informations. Si c'est le cas, l'utilisateur est redirigé vers la page principale de l'application, sinon un message d'erreur est affiché sur la page de connection et l'utilisateur est invité à retenter une connection.

Pour chacun des contrôleurs suivant, lorsque l'utilisateur n'est pas authentifié, il est redirigé vers le contrôleur "AuthController".

Le contrôleur "LeavePersoController" permet d'afficher la liste des congés pris et en attente de validation. Il permet aussi de modifier ou supprimer un congé en cours de validation. Il donne aussi accès au bouton permettant de faire une nouvelle demande de congé. Lors de la réception d'un requête, le contrôleur récupère la liste des congés lié à l'employé, auprès de la bdd, et les passes dans les paramètres de la requête. Lorsque l'utilisateur clique sur le bouton de suppression d'un congé, une paramètre "action" avec la valeur "delete" est ajouté à la requête. Si ce paramètre est présent le contrôleur récupère la valeur de du paramètre "delBeginDate" et supprime le congé, lié à cette date et à l'utilisateur, en BDD.

Le contrôleur "LeaveCreateController" permet l'affichage de la page de création de congé ainsi que l'ajout de celui-ci en BDD. Lors de la réception d'une requête POST, les paramètres "bday", "eday", "motif" et "type" sont récupéré. Si ces paramètres sont valides et que le nombre de congé de l'employé est suffisant, le congé est créé et ajouté en BDD.

Le contrôleur "LeaveEditController" permet la modification d'une demande de congé et l'affichage de la page lié à cette modification. Lors de la réception d'un requête GET, le congé, lié à la date de début passé en paramètre de l'URL, est récupéré en base puis passé en paramètre de requête. Lors de la réception d'un paramètre POST, les paramètres "bday", "eday", "motif" et "type" sont récupéré. Si les informations sont toujours valide et le solde de congés le permet, le congé est modifié en BDD.

Le contrôleur "LeaveEmpController" permet l'affichage des congés à valider et pris des personnes sous la direction du RH ou du chef d'équipe authentifié. Lors

de la réception d'un requête GET, le contrôleur récupère la liste des employés sous la direction de l'utilisateur connecté. Il récupère ensuite la liste des congés à valider et la liste des congés acceptée pour pouvoir les afficher. Si la valeur du paramètre "action", stockée dans l'URL, est à "accept", les paramètres "upBday", "upLogin" et "upWording" sont également récupérés. Ces paramètre permettront d'identifier le congé accepté. Les mêmes actions sont effectuées pour le refus d'un congé, mais cette fois la paramètre "action" est à "decline". Il vérifie aussi qu'un commentaire à bien été laissé, grâce au paramètre "upWording". Si ce n'est pas le cas, une erreur est retournée.

Le contrôleur "StatisticsController" permet l'affichage de statistiques sur l'ensemble des données stockées dans la BDD. Lors de la réception d'une requête GET, l'ensemble des information à afficher dans le graphique est récupéré puis passé en paramètre de cette même requête.

Le contrôleur "EmployeeController" permet aux employés RH d'accéder à la liste de l'ensemble de employés. Il permet aussi de modifier ou supprimer une fiche employé. Il donne aussi accès au bouton permettant de créer une nouvelle fiche. Lors de la réception d'un requête, le contrôleur récupère la liste des employé, puis passe cette liste en paramètre de la requête. Si le paramètre "action" est à "delete", le paramètre "delLogin" est également récupéré et l'employé lié à ce login est supprimé de la base.

Le contrôleur "EmployeeCreateController" permet l'affichage de la page de création d'une fiche employé ainsi que l'ajout de celle-ci en BDD. Lors de la réception d'une requête POST, l'ensemble des informations transmises par le formulaire de création de fiche sont récupéré et sont utilisé pour créer l'employé et pour l'ajouter à la BDD.

Le contrôleur "EmployeeEditController" permet la modification d'une fiche d'employé et l'affichage des données lié à cette utilisateur. Lors de la réception d'un requête GET, la fiche, lié au login passé en paramètre de l'URL, est récupéré en base puis passé en paramètre de requête, pour l'affichage des informations. Lors de la réception d'un paramètre POST, l'ensemble des informations transmises par le formulaire de modification sont récupéré. Ces informations serviront à modifier la fiche stocké en BDD.

Les modifications en BDD, présenté plus haut, se font grâce aux classes “EmployeeRepoImp”, “LeaveRepoImp” et “AuthRepoImp”.

La classes “EmployeeRepoImp” permet de gérer la table “Employe”. Cette classe possède plusieurs fonctions.

La fonction “getEmploye()” prend en paramètre le login d’un utilisateur et envoi la requête "*SELECT * FROM Employe WHERE login* = " + le login de l’utilisateur. Il utilise les information récupérées dans la BDD pour créer un Employe qu’il retournera.

La fonction “actualizeRemainingBalance()” prend en paramètre un Employe et le nouveau nombre de congés restant à l’employé. Cette fonction met à jour la base de données grâce à la requête "*UPDATE Employe SET solde_conges = ? WHERE login = ?*".

La fonction “getEmployees()” récupère tous les éléments de la table “Employe” grâce à la requête “*SELECT * FROM Employe*”, puis transforme ces information en une liste d’objet Employee avant de les retourner.

La fonction “addEmployee()”, prenant en paramètre un Employe, récupère l’ensemble des information puis les ajoute en base grâce aux requètes "*INSERT INTO Employe(login, pnom, nom, adresse, ville, code_postal, equipe, fonction, mail, solde_conges) VALUES(?,?,?,?,?,?,?,?,?,?)*" et "*INSERT INTO Authentification(login, password) VALUES(?,?)*".

La fonction “updateEmployee()”, prenant en paramètre un Employe et un chaine de caractère “login”, permet de mettre à jour les informations d’un employé en modifiant les tables “Employe” et “Authentification” grâce au requêtes “*UPDATE Employe SET login = ?, pnom = ?, nom = ?, adresse = ?, ville = ?, code_postal = ?, equipe = ?, fonction = ?, mail = ?, solde_conges = ? WHERE login = ?*” et “*UPDATE Authentification SET login = ?, password = ? VALUES(?,?) WHERE login = ?*”

La fonction “countEmployeesByTeam()” permet de compter le nombre d’employé par équipe. Cette fonction prend en paramètre le nom de l’équipe. L’information est récupéré depuis la base “Employ” grâce à la requête suivante : “*SELECT count(*) as total FROM Employe WHERE equipe = ?*”.

La dernière fonction de “EmployeeRepoImpl” est la suppresion d’un utilisateur. Elle se nomme “deleteEmployee()” et prend en paramètre un objet Employe. La fonction récupète le login de l’utilisateur, puis supprime les ligne correspondante en BDD grâce aux commandes suivante : “*DELETE FROM Employe WHERE login = ?*”, “*DELETE FROM Authentification WHERE login = ?*”.

La classes “LeaveRepoImpl” permet de gérer la table “Conge”.

La fonction “getLeaves()” de cette classe, prendre un objet Employee en

paramètre, et permet de récupérer l'ensemble des congés d'un utilisateur. La fonction récupère le login de l'utilisateur puis l'intègre à la requête SQL suivante : "*SELECT * FROM Conge WHERE login = ?*". Les résultats obtenus sont transformés en objet Leave puis retourné sous la forme d'une liste.

La fonction "addLeave()" prend en paramètre un objet Leave et permet d'ajouter une ligne dans la table "Conge", contenant les informations de l'objet. L'insertion dans la table se fait grâce à la requête suivante : "*INSERT INTO Conge(login, date_debut, date_fin, duree, motif, type_conges, etat) VALUES(?,?,?,?,?,?,?)*".

La fonction "deleteLeave()", à contrariode la précédente, permet de supprimer un congé dans la table. Cette fonction prend un objet Leave en paramètre. Le login et la date du début de congé sont récupérés dans l'objet puis ajoutés à la requête de suppression "*DELETE FROM Conge WHERE login = ? and date_debut = ?*".

La fonction "getLeave()", prend en paramètre un login d'utilisateur et une date de début de congé, et permet de récupérer un congé depuis la base de données. Cette récupération se fait grâce à la requête "*SELECT * FROM Conge WHERE login = ? and date_debut = ?*".

La fonction "getLeavesByState()", proche de la précédente, permet de récupérer l'ensemble des congés, n'appartenant pas à l'utilisateur, selon leur état (en attente, validé, refusé). Pour cela, la fonction prend un objet Employee et une String state en paramètre. La récupération en base se fait grâce à la requête "*SELECT * FROM Conge WHERE login != ? AND etat = ?*".

La fonction "updateLeave()" permet de mettre à jour un congé dans la table "Conge". Elle prend en paramètre un objet Leave ainsi que l'ancienne date de début. L'ensemble des informations liées aux congés est récupéré puis inséré dans la requête de modification suivante "*UPDATE Conge SET date_debut = ?, date_fin = ?, duree = ?, motif = ?, type_conges = ?, etat = ?, date_validation = ?, commentaire = ? WHERE login = ? AND date_debut = ?*".

La fonction "countLeavesByState()" permet de compter le nombre de congé par type stocké dans la table "Conge". Pour cela, la fonction prend en paramètre une String state, définissant le type de congés à récupérer. L'information est ensuite récupérés grâce à la requête "*SELECT count(*) as total FROM Conge WHERE etat = ?*" puis est retournée.

La fonction "countLeavesByMonth()" permet de compter le nombre de congé par mois stocké dans la table "Conge". Pour cela, la fonction prend en paramètre un entier "month", définissant la date de début des congés à récupérer. L'information est ensuite récupérés grâce à la requête "*SELECT count(*) as total FROM Conge WHERE MONTH(date_debut) = ?*" puis est retournée.

La classe “AuthRepoImp” permet uniquement d’obtenir le mot de passe d’un utilisateur grâce à la fonction “getPassword()” prenant en paramètre le login de l’utilisateur et récupérant les informations grâce à la requête "SELECT * FROM Authentification".

Conclusion

La mise en oeuvre de ce projet nous a permis d’utiliser JEE dans un contexte professionnel au travers d’un projet concret. Dans la continuité de l’analyse UML nous avons dû mettre en oeuvre une architecture complexe avec l’utilisation de Tomcat et de MySQL : ce qui nous a permis de mettre en application nos acquis des modules de Base de données. C’est le fonctionnement dans cette architecture qui nous amené des complications mais qui nous a aussi permis de créer une application complexe et de progresser dans la conception et le développement JEE.

```
mysql> show tables;
+----------------------+
| Tables_in_projetjee |
+----------------------+
| authentification     |
| conge                |
| employe              |
| ref_equipes          |
| ref_etat_conges      |
| ref_fonction         |
| ref_motif_conges     |
| ref_type_conges      |
| rendez_vous          |
+----------------------+
9 rows in set (0.03 sec)
```

```
mysql> select * from authentification;
+----------+----------+
| login    | password |
+----------+----------+
| cyoudec  | admin    |
| lgagnant | admin    |
+----------+----------+
2 rows in set (0.00 sec)
```

```
mysql> select * from conge;
+----------+------------+------------+-------+----------------------+-------------+------------+-----------------+-------------+
| login    | date_debut | date_fin   | duree | motif                | type_conges | etat       | date_validation | commentaire |
+----------+------------+------------+-------+----------------------+-------------+------------+-----------------+-------------+
| cyoudec  | 2019-11-22 | 2019-11-24 |     2 | Maladie              | CP          | En attente | NULL            | NULL        |
| lgagnant | 2019-11-26 | 2019-11-28 |     2 | Raisons personnelles | RTT         | En attente | NULL            | NULL        |
| mboue    | 2019-11-15 | 2019-11-17 |     2 | Enfants malades      | CP          | En attente | NULL            | NULL        |
| nguzzo   | 2019-11-22 | 2019-11-24 |     2 | Maladie              | CP          | En attente | NULL            | NULL        |
| nguzzo   | 2019-12-22 | 2019-12-27 |     5 | Maladie              | CP          | En attente | NULL            | NULL        |
+----------+------------+------------+-------+----------------------+-------------+------------+-----------------+-------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from employe;
+---------+--------+--------+----------------+--------+-------------+-------------+----------------+--------------------+-------------+
| login   | pnom   | nom    | adresse        | ville  | code_postal | equipe      | fonction       | mail               | solde_conges |
+---------+--------+--------+----------------+--------+-------------+-------------+----------------+--------------------+-------------+
| cyoudec | Clement | YOUDEC | 4 rue de rouge  | Lannion | 22300       | RH          | Membre d'equipe | cyoudec@enssat.fr  |          20 |
| lgagnant | Laurre | GAGNANT | 4 rue de orange | Lannion | 22300       | Informatique | Membre d'equipe | lgagnant@enssat.fr |           1 |
| mboue   | Martin | BOUE   | 4 rue de violet | Lannion | 22300       | Comptabilite | Chef d'equipe  | mboue@enssat.fr    |           1 |
| nguzzo  | Nicolas | GUZZO | 4 rue de verte  | Lannion | 22300       | RH          | Chef d'equipe  | nguzzo@enssat.fr   |          10 |
| tlegoff | Thomas | LE GOFF | 4 rue de jaune | Lannion | 22300       | Comptabilite | Membre d'equipe | tlegoff@enssat.fr |          50 |
| yperrot | Yohan  | PERROT | 4 rue de bleu  | Lannion | 22300       | Informatique | Chef d'equipe  | yperrot@enssat.fr  |           3 |
+---------+--------+--------+----------------+--------+-------------+-------------+----------------+--------------------+-------------+
6 rows in set (0.00 sec)
```

```
mysql> select * from ref_equipes;
+--------------+
| nom_equipe   |
+--------------+
| Comptabilite |
| Informatique |
| RH           |
+--------------+
3 rows in set (0.01 sec)
```

```
mysql> select * from ref_etat_conges;
+------------+
| etat       |
+------------+
| En attente |
| Refuse     |
| Valide     |
+------------+
3 rows in set (0.01 sec)
```

```
mysql> select * from ref_fonction;
+-----------------+
| nom_fonction    |
+-----------------+
| Chef d'equipe   |
| Membre d'equipe |
+-----------------+
2 rows in set (0.01 sec)
```

```
mysql> select * from ref_motif_conges;
+----------------------+
| motif_conges         |
+----------------------+
| Enfants malades      |
| Famille              |
| Maladie              |
| Raisons personnelles |
+----------------------+
4 rows in set (0.01 sec)
```

```
mysql> select * from ref_type_conges;
+-------------+
| type_conges |
+-------------+
| CP          |
| Formation   |
| RTT         |
+-------------+
3 rows in set (0.01 sec)
```

```
mysql> select * from rendez_vous;
+---------+------------+-------------+
| login   | date_rdv   | libelle     |
+---------+------------+-------------+
| cyoudec | 2019-11-15 | revu projet |
+---------+------------+-------------+
1 row in set (0.00 sec)
```