Correction timesheet
CRUD
Cascade
Fetch

05/03/2017

Walid YAICH
walid.yaich@esprit.tn
Bureau E204

Plan

- **Ajouter entreprise**
- **Ajouter département**
- **Affecter département a entreprise**
- **Ajouter Département ET affecter l'entreprise associé**
- **Ajouter Entreprise avec sa liste de départements**
- **Ajouter Entreprise avec sa liste de départements(Cascade)**
- **Supprimer Entreprise**
- **Supprimer Entreprise (supprimer tous ses départements)**
- **Supprimer Entreprise (Cascade)**
- **Les types de cascades**
- **Récupérer la liste des départements (par défaut LAZY)**
- **Récupérer la liste des départements (EAGER)**
- **Default fetching policy**
- **Supprimer Employé**
- **Mettre a jour email d'un employé**
- **Mettre a jour email d'un employé - JPQL**
- **Supprimer tous les contrats - JPQL**
- **Chercher le salaire d'un employé (join)**
- **Ajouter un timesheet**
- **Valider un timesheet**
- **Récupérer toutes les missions d'un employé (join)**
- **Manipuler les dates**

Ajouter entreprise

Projet Serveur

```java
@Override
public int ajouterEntreprise(Entreprise entreprise) {
    em.persist(entreprise);
    return entreprise.getId();
}
```

Projet Client

```java
Entreprise ssiiConsulting = new Entreprise("SSII Consulting", "Cite El Ghazela");
int ssiiConsultingId = entrepriseServiceRemote.ajouterEntreprise(ssiiConsulting);
```

Ajouter département

Projet Serveur

```java
@Override
public int ajouterDepartement(Departement dep) {
    em.persist(dep);
    return dep.getId();
}
```

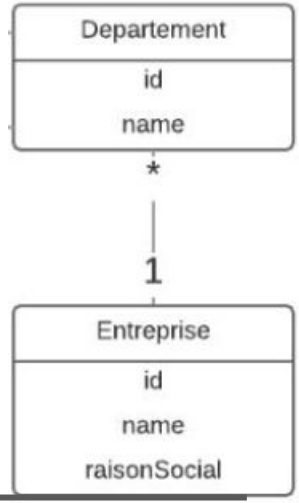
Projet Client

```java
Departement depRH = new Departement("RH");
Departement depTelecom = new Departement("Telecom");

int depTelecomId = entrepriseServiceRemote.ajouterDepartement(depTelecom);
int depRhId = entrepriseServiceRemote.ajouterDepartement(depRH);
```

Affecter département a entreprise

Projet Serveur

```java
@Override
public void affecterDepartementAEntreprise(int depId, int entrepriseId) {
    Entreprise entrepriseManagedEntity = em.find(Entreprise.class, entrepriseId);
    Departement depManagedEntity = em.find(Departement.class, depId);

    entrepriseManagedEntity.getDepartements().add(depManagedEntity);
    //ceci ne met pas a jour la relation !
}
```



BD

id	name	entreprise_id
1	Telecom	NULL
2	RH	NULL
NULL	NULL	NULL

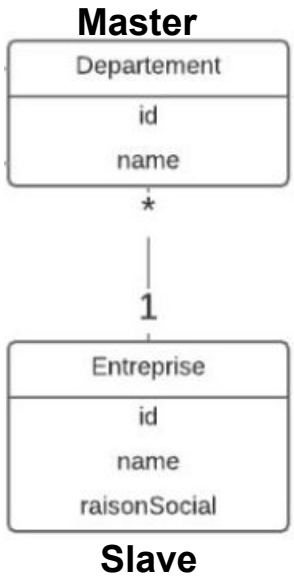
Projet Client

```java
entrepriseServiceRemote.affecterDepartementAEntreprise(depTelecomId, ssiiConsultingId);
entrepriseServiceRemote.affecterDepartementAEntreprise(depRhId, ssiiConsultingId);
```

Affecter département a entreprise

Projet Serveur

```java
@Override
public void affecterDepartementAEntreprise(int depId, int entrepriseId) {
    //Le bout Master de cette relation N:1 est departement
    //donc il faut rajouter l'entreprise a departement
    // ==> c'est l'objet departement(le master) qui va mettre a jour l'association
    //Rappel : la classe qui contient mappedBy represente le bout Slave
    //Rappel : Dans une relation oneToMany le mappedBy doit etre du cote one.
    Entreprise entrepriseManagedEntity = em.find(Entreprise.class, entrepriseId);
    Departement depManagedEntity = em.find(Departement.class, depId);

    depManagedEntity.setEntreprise(entrepriseManagedEntity);
    //inutile de faire : em.merge(depManagedEntity);
    //Dans ce cas, le merge n'est pas utile parce que depManagedEntity est
    //dans l'état managed, autrement dit,
    //depManagedEntity existe dans le persistence context.
    //n'importe quel entité dans le persistence context sera automatiquement
    //synchronisé avec la base de donnees
}
```



BD

id	name	entreprise_id
1	Telecom	1
2	RH	1

Projet Client

```java
entrepriseServiceRemote.affecterDepartementAEntreprise(depTelecomId, ssiiConsultingId);
entrepriseServiceRemote.affecterDepartementAEntreprise(depRhId, ssiiConsultingId);
```

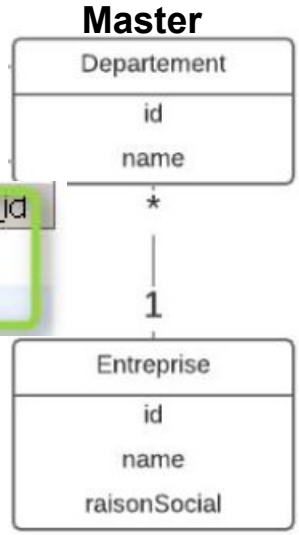
Ajouter Département ET affecter l'entreprise associé

Projet Serveur

```java
@Override
public int ajouterDepartement(Departement dep) {
    em.persist(dep);
    return dep.getId();
}
```

BD

id	name	entreprise_id
1	Telecom	1
2	RH	1



Projet Client

```java
Entreprise ssiiConsulting = new Entreprise("SSII Consulting", "Cite El Ghazela");
int ssiiConsultingId = entrepriseServiceRemote.ajouterEntreprise(ssiiConsulting);
ssiiConsulting.setId(ssiiConsultingId);

Departement depTelecom = new Departement("Telecom");
depTelecom.setEntreprise(ssiiConsulting);//En ajoutant cette ligne, on n'a plus besoin
//d'appeler la methode affecterDepartementAEntreprise(depTelecomId, ssiiConsultingId)
int depTelecomId = entrepriseServiceRemote.ajouterDepartement(depTelecom);

Departement depRH = new Departement("RH");
depRH.setEntreprise(ssiiConsulting);//En ajoutant cette ligne, on n'a plus besoin
//d'appeler la methode affecterDepartementAEntreprise(depTelecomId, ssiiConsultingId)
int depRhId = entrepriseServiceRemote.ajouterDepartement(depRH);
```

Il faut mettre l'ID !

Ajouter Entreprise avec sa liste de dépar

Projet Serveur

```
@Override
public int ajouterEntreprise(Entreprise entreprise) {
    em.persist(entreprise);
    return entreprise.getId();
}
```

BD

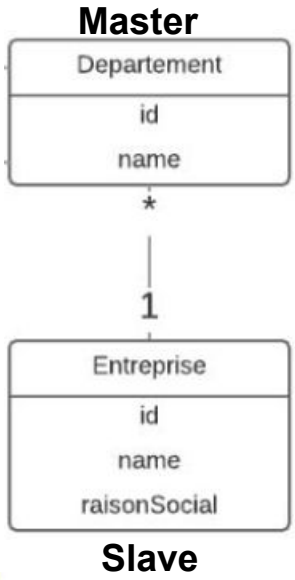
Entreprise

id	name	raisonSocial
1	SSII Consulting	Cite El Ghazela

Departement

id	name	entreprise_id
NULL	NULL	NULL

Defaults to no operations being cascaded



Projet Client

```
Entreprise ssiiConsulting = new Entreprise("SSII Consulting", "Cite El Ghazela");

Departement depTelecom = new Departement("Telecom");
Departement depRH = new Departement("RH");

List<Departement> depts = new ArrayList<>();
depts.add(depRH);
depts.add(depTelecom);

ssiiConsulting.setDepartements(depts);
```

Ajouter Entreprise avec sa liste de départements(Cascade)

Projet Serveur

```
@Entity
public class Entreprise implements Serializable{
    @OneToMany(mappedBy="entreprise", cascade = CascadeType.PERSIST)
    private List<Departement> departements = new ArrayList<>();
```

BD

Entreprise

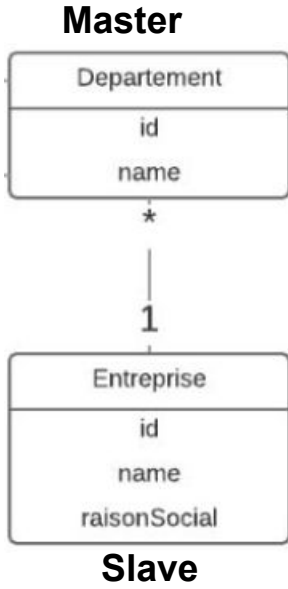
id	name	raisonSocial
1	SSII Consulting	Cite El Ghazela

Departement

id	name	entreprise_id
1	RH	NULL
2	Telecom	NULL

Projet Client

```
Entreprise ssiiConsulting = new Entreprise("SSII Consulting", "Cite El Ghazela");

Departement depTelecom = new Departement("Telecom");
Departement depRH = new Departement("RH");

List<Departement> depts = new ArrayList<>();
depts.add(depRH);
depts.add(depTelecom);

ssiiConsulting.setDepartements(depts);
```



Ajouter Entreprise avec sa liste de départements(Cascade)

Projet Serveur

```
@Entity
public class Entreprise implements Serializable{
        public void addDepartement(Departement departement){
            departement.setEntreprise(this);
            this.departements.add(departement);
        }
```

BD

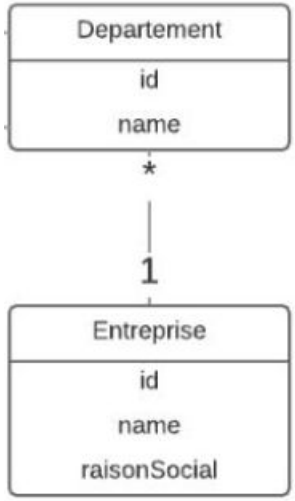
Entreprise

id	name	raisonSocial
1	SSII Consulting	Cite El Ghazela

Departement

id	name	entreprise_id
1	RH	1
2	Telecom	1

Master



Slave

Projet Client

```
Entreprise ssiiConsulting = new Entreprise("SSII Consulting", "Cite El Ghazela");

Departement depTelecom = new Departement("Telecom");
Departement depRH = new Departement("RH");

ssiiConsulting.addDepartement(depRH);
ssiiConsulting.addDepartement(depTelecom);

int ssiiConsultingId = entrepriseServiceRemote.ajouterEntreprise(ssiiConsulting);
```

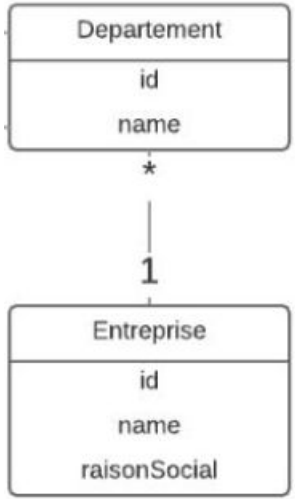
Supprimer Entreprise

Projet Serveur

```
@Override
public void deleteEntrepriseById(int entrepriseId){
    em.remove(em.find(Entreprise.class, entrepriseId));
}
```

```
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException:
Cannot delete or update a parent row: a foreign key constraint fails
(`imputation`.`departement`, CONSTRAINT `FK_3g9owgm3py9ek34qtssolyb7w` FOREIGN KEY
(`entreprise_id`) REFERENCES `entreprise` (`id`))
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
```

Master



Slave

BD

Entreprise

id	name	raisonSocial
1	SSII Consulting	Cite El Ghazela

Departement

id	name	entreprise_id
1	RH	1
2	Telecom	1

Projet Client

```
int ssiiConsultingId = 1;
entrepriseServiceRemote.deleteEntrepriseById(ssiiConsultingId);
```

Supprimer Entreprise (supprimer tous ses départements)

Projet Serveur

```java
@Override
public void deleteDepartementById(int depId){
    em.remove(em.find(Departement.class, depId));
}
```

```java
@Override
public void deleteEntrepriseById(int entrepriseId){
    em.remove(em.find(Entreprise.class, entrepriseId));
}
```

BD

Entreprise

id	name	raisonSocial
NULL	NULL	NULL

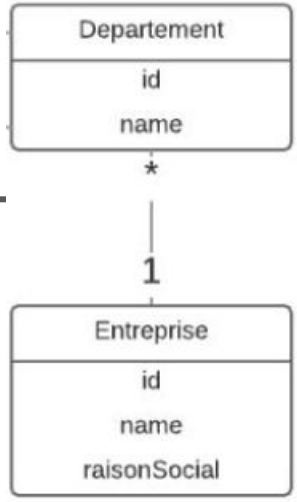
Departement

id	name	entreprise_id
NULL	NULL	NULL

Projet Client

```java
//Supprimer le Master(Departement) puis le Slave
//Le mappedBy est du coté Slave
int depRhId = 2;
int depTelecomId = 1;

entrepriseServiceRemote.deleteDepartementById(depTelecomId);
entrepriseServiceRemote.deleteDepartementById(depRhId);

int ssiiConsultingId = 1;
entrepriseServiceRemote.deleteEntrepriseById(ssiiConsultingId);
```

Master

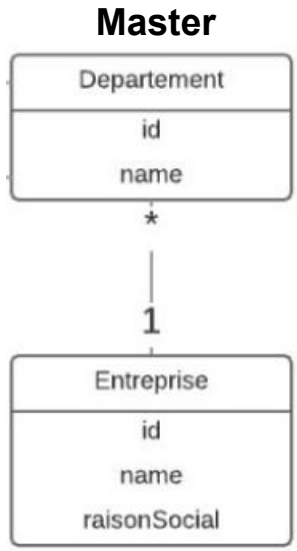


Slave

Supprimer Entreprise (Cascade)

Projet Serveur

```
@Entity
public class Entreprise implements Serializable{
  @OneToMany(mappedBy="entreprise", cascade = {CascadeType.PERSIST, CascadeType.REMOVE})
  private List<Departement> departements = new ArrayList<>();
```



BD

Entreprise

id	name	raisonSocial
NULL	NULL	NULL

Departement

id	name	entreprise_id
NULL	NULL	NULL

Projet Client

```
int ssiiConsultingId = 1;
entrepriseServiceRemote.deleteEntrepriseById(ssiiConsultingId);
```

Les types de cascades

- CascadeType.PERSIST
- CascadeType.MERGE
- CascadeType.REFRESH
- CascadeType.REMOVE
- CascadeType.DETACH
- CascadeType.ALL

Récupérer la liste des départements (par défaut LAZY)

Projet Serveur

```
@Override
public Entreprise getEntrepriseById(int entrepriseId) {
    return em.find(Entreprise.class, entrepriseId);
}
```

BD

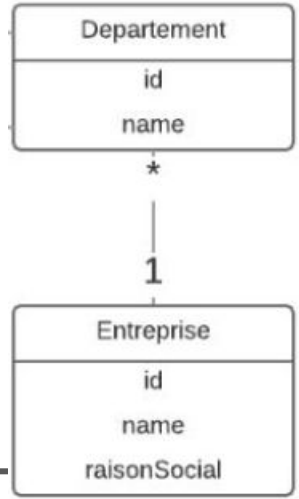
Entreprise

id	name	raisonSocial
1	SSII Consulting	Cite El Ghazela

Departement

id	name	entreprise_id
1	RH	1
2	Telecom	1

Master



Slave

Projet Client

```
Entreprise entreprise = entrepriseServiceRemote.getEntrepriseById(1);

for(Departement dep : entreprise.getDepartements()){
    System.out.println(dep.getName());
}
```

```
Exception in thread "main" org.hibernate.LazyInitializationException: failed to
lazily initialize a collection of role:
tn.esprit.timesheet.entities.Entreprise.departements, could not initialize proxy
```

Récupérer la liste des départements (EAGER)

Projet Serveur

```
@Entity
public class Entreprise implements Serializable{

    @OneToMany(mappedBy="entreprise",
            cascade = {CascadeType.PERSIST, CascadeType.REMOVE},
            fetch=FetchType.EAGER)
    private List<Departement> departements = new ArrayList<>();
```

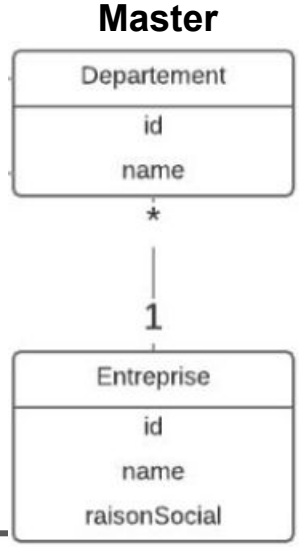
BD

Entreprise

id	name	raisonSocial
1	SSII Consulting	Cite El Ghazela

Departement

id	name	entreprise_id
1	RH	1
2	Telecom	1



Projet Client

```
Entreprise entreprise = entrepriseServiceRemote.getEntrepriseById(1);

for(Departement dep : entreprise.getDepartements()){
    System.out.println(dep.getName());
}
```

```
RH
Telecom
```

Default fetching policy

Association type	Default fetching policy
@OneTMany	LAZY
@ManyToMany	LAZY
@ManyToOne	EAGER
@OneToOne	EAGER

Supprimer Employé

Projet Serveur

```
@Override
public void deleteEmployeById(int employeId) {
    Employe employe = em.find(Employe.class, employeId);
    em.remove(employe);
}
```

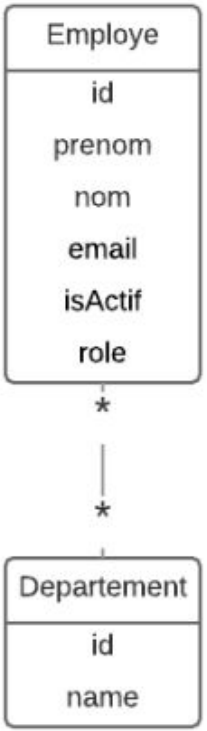
```
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException:
Cannot delete or update a parent row: a foreign key constraint fails
(`imputation`.`departement_employe`, CONSTRAINT `FK_fs8ophksl1ahctrfwo3o5ythx` FOREIGN KEY
(`employes_id`) REFERENCES `employe` (`id`))
```

BD

id	nom	prenom	role	email	isActif
1	kallel	khaled	ING...	Khaled.k...	1
2	zitouni	mohamed	TEC...	mohame...	1
3	ouali	aymen	ING...	aymen.o...	1
4	bouzid	bochra	CHE...	bochra.b...	1
5	arbi	yosra	CHE...	yosra.ar...	1

departements_id	employes_id
2	1
2	2
2	4
1	1
1	2
1	3
1	5

Master



Slave

Projet Client

```
employeServiceremote.deleteEmployeById(aymenOualiId);
```

Supprimer Employé

Projet Serveur

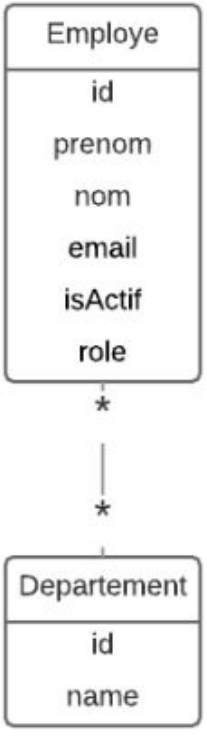
```
@Override
public void deleteEmployeById(int employeId) {
    Employe employe = em.find(Employe.class, employeId);

    //Desaffecter l'employe de tous les departements
    //c'est le bout master qui permet de mettre a jour
    //la table d'association
    for(Departement dep : employe.getDepartements()){
        dep.getEmployes().remove(employe);
    }

    em.remove(employe);
}
```

BD

id	nom	prenom	email	isActif	role
1	kallel	khaled	Khaled.k...	1	ING...
2	zitouni	mohamed	mohame...	1	TEC...
4	bouzid	bochra	bochra.b...	1	CHE...
5	arbi	yosra	yosra.ar...	1	CHE...

departements_id	employes_id
2	1
2	2
2	4
1	1
1	2
1	5

Master



Slave

Projet Client

```
employeServiceremote.deleteEmployeById(aymenQualiId);
```

Mettre a jour email d'un employé

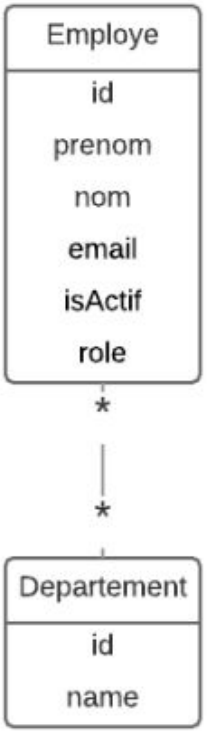
Projet Serveur

```
@Override
public void mettreAjourEmailByEmployeId(String email, int employeId) {
    Employe employe = em.find(Employe.class, employeId);
    employe.setEmail(email);
}
```

BD

id	email	isActif	nom	prenom	role
1	Khaled.kallel@ssiiconsult...	1	kallel	khaled	ING...
2	mohamed.zitouni@ssiic...	1	zitouni	mohamed	TEC...
3	aymen.ouali@ssiiconsul...	1	ouali	aymen	ING...
4	newEmail@email.tn	1	bouzid	bochra	CHE...
5	yosra.arbi@ssiiconsultin...	1	arbi	yosra	CHE...

Master



Slave

Projet Client

```
employeServiceremote.mettreAjourEmailByEmployeId("newEmail@email.tn", bochraBouzidId);
```

Mettre a jour email d'un employé - JPQL

Projet Serveur

```java
@Override
public void mettreAjourEmailByEmployeIdJPQL(String email, int employeId){
    Query query = em.createQuery("update Employe e set email=:email where e.id=:employeId");
    query.setParameter("email", email);
    query.setParameter("employeId", employeId);
    int modified = query.executeUpdate();
    if(modified == 1){
        System.out.println("successfully updated");
    }else{
        System.out.println("failed to update");
    }
}
```

BD

Projet Client

```java
employeServiceremote.mettreAjourEmailByEmployeId("newEmail@email.tn", bochraBouzidId);
```

Supprimer tous les contrats - JPQL

Projet Serveur

```java
@Override
public void deleteAllContratJPQL() {
    int modified = em.createQuery("delete from Contrat").executeUpdate();
    if(modified > 1){
        System.out.println("successfully deleted");
    }else{
        System.out.println("failed to delete");
    }
}
```

BD

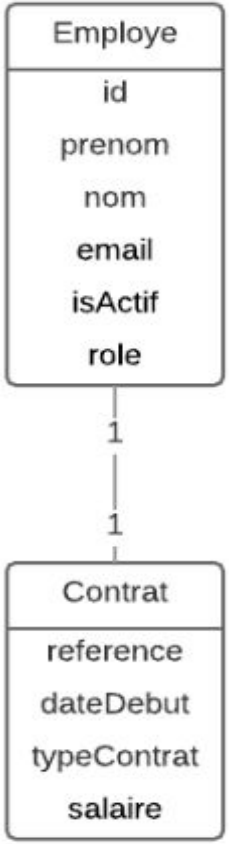
Projet Client

```java
employeServiceremote.deleteAllContratJPQL();
```

Chercher le salaire d'un employé (join)

Projet Serveur

```
@Override
public float getSalaireByEmployeIdJPQL(int employeId) {
  TypedQuery<Float> query = em.createQuery(
  "select c.salaire from Contrat c join c.employe e where e.id=:employeId",
  Float.class);
  query.setParameter("employeId", employeId);
  return query.getSingleResult();
}
```



BD

reference	dateDebut	salaire	typeContrat	employe_id
1	2015-02-01	1600	CDI	1
2	2010-03-01	2600	CDI	5
3	2013-05-15	900	CDI	2
4	2014-05-10	2000	CDI	3

Projet Client

```
float salaire = employeServiceremote.getSalaireByEmployeIdJPQL(aymenOualiId);
System.out.println("Le salaire de l'employe dont l'id est : " + aymenOualiId + " est : " + salaire);
```

Ajouter un timesheet

Projet Serveur

```
@Override
public void ajouterTimesheet(int missionId, int employeId, Date dateDebut, Date dateFin) {
    TimesheetPK timesheetPK = new TimesheetPK();
    timesheetPK.setDateDebut(dateDebut);
    timesheetPK.setDateFin(dateFin);
    timesheetPK.setIdEmploye(employeId);
    timesheetPK.setIdMission(missionId);

    Timesheet timesheet = new Timesheet();
    timesheet.setTimesheetPK(timesheetPK);
    timesheet.setValide(false); //par defaut non valide
    entityManager.persist(timesheet);
}
```

BD

dateDebut	dateFin	idEmploye	idMission	isValide
2016-01-01	2016-03-16	2	1	0
2016-01-01	2016-06-15	1	2	0
2016-01-01	2016-12-31	3	1	0

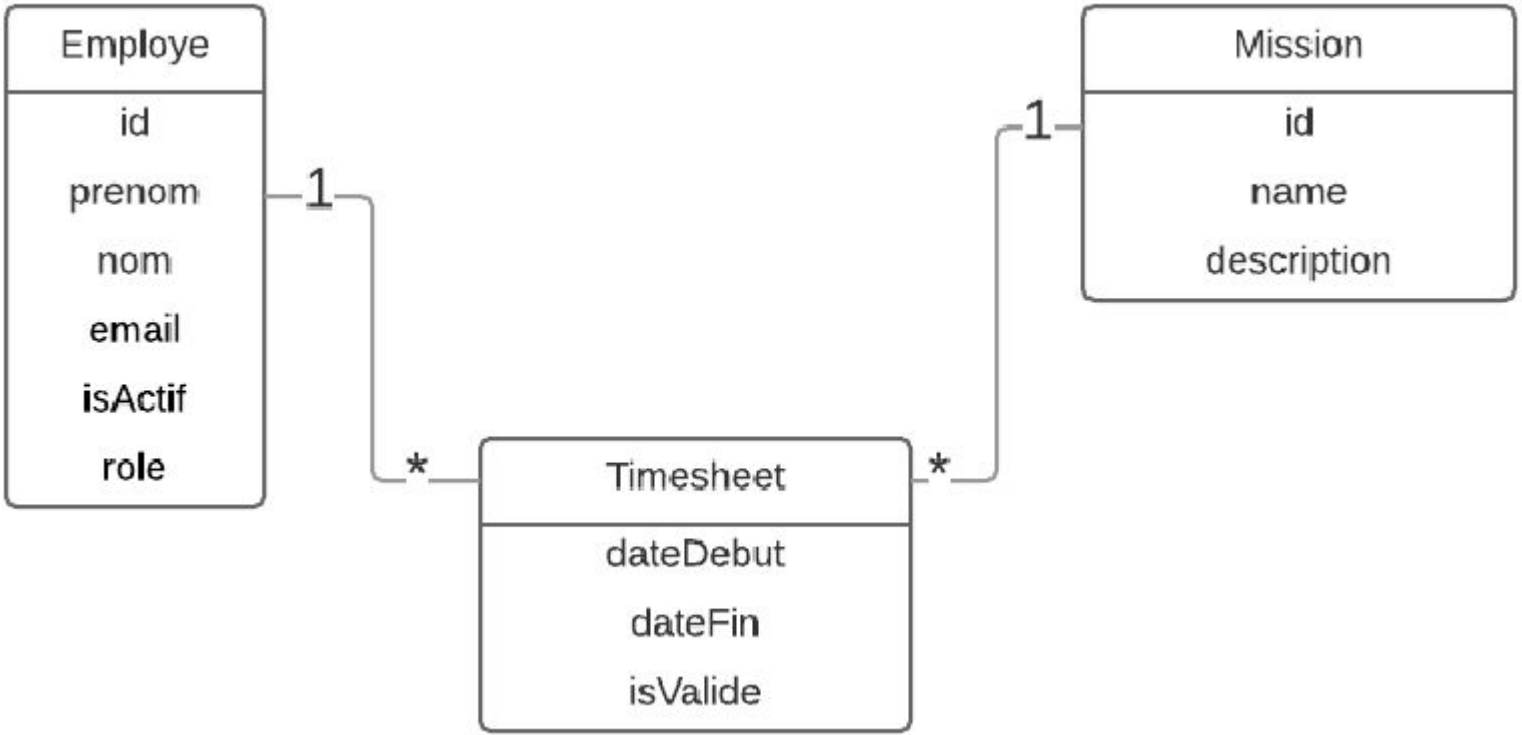
Projet Client

```
SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
//ajouter un timesheet pour Aymen
Date dateDebutAymenOuali4G = dateFormat.parse("01/01/2016");
Date dateFinAymenOuali4G = dateFormat.parse("31/12/2016");
int aymenOualiId = 3;
timesheetServiceRemote.ajouterTimesheet(miseEnPlace4GId, aymenOualiId,
                                        dateDebutAymenOuali4G, dateFinAymenOuali4G);
```

Valider un timesheet

```java
@Override
public void validerTimesheet(int missionId, int employeId, Date dateDebut, Date dateFin, int validateurId) {
    Employe validateur = entityManager.find(Employe.class, validateurId);
    Mission mission = entityManager.find(Mission.class, missionId);

    //verifier s'il est un chef de departement (interet des enum)
    if(!validateur.getRole().equals(Role.CHEF_DEPARTEMENT)){
        System.out.println("l'employe doit etre chef de departement pour valider une feuille de temps !");
        return;
    }
    //verifier s'il est le chef de departement de la mission en question
    boolean chefDeLaMission = false;
    for(Departement dep : validateur.getDepartements()){
        if(dep.getId() == mission.getDepartement().getId()){
            chefDeLaMission = true;
            break;
        }
    }
    if(!chefDeLaMission){
        System.out.println("l'employe doit etre chef de departement de la mission en question");
        return;
    }

    TimesheetPK timesheetPK = new TimesheetPK(missionId, employeId, dateDebut, dateFin);
    Timesheet timesheet = entityManager.find(Timesheet.class, timesheetPK);
    timesheet.setValide(true);
}
```

Récupérer toutes les missions d’un employé



Récupérer toutes les missions d'un employé (join)

```
@Override
public List<Mission> findAllMissionByEmployeJPQL(int employeId) {
    TypedQuery<Mission> query= entityManager.createQuery(
    "select DISTINCT m from Mission m join m.timesheets t join t.employe e where e.id=:employeId",
    Mission.class);
    query.setParameter("employeId", employeId);
    return query.getResultList();
}
```

Manipuler les dates

Déclarer une date

```
//Choisir le TemporalType selon le besoin metier
@Temporal(TemporalType.DATE)
private Date dateDebut;
```

Stocker une date

```
SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
//ajouter un timesheet pour Aymen
Date dateDebutAymenOuali4G = dateFormat.parse("01/01/2016");
Date dateFinAymenOuali4G = dateFormat.parse("31/12/2016");
```

Lire une date

```
SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
System.out.println("dateDebut : " + dateFormat.format(timesheet.getTimesheetPK().getDateDebut()));
```