# Table 1 : EmployeSimple

**Description**

La table EmployeSimple regroupe l'ensemble des salariés réels de l'entreprise. Elle contient toutes les informations complètes indispensables à la gestion des employés et de la paie (données personnelles, professionnelles et contractuelles).

**Paramètres (Champs)**

- **id_employe (PK)** : Identifiant unique du salarié.
- **nom** : Nom de famille.
- **prenom** : Prénom.
- **nom de l'utilisateur** : Identifiant pour se connecter =>mettre dans une classe Rôle de laquelle doit hériter (Employe, RH, admin, configurateur)
- **cin** : Carte d'identité nationale.
- **email_pro** : Adresse email professionnelle (unique).
- **email_perso** : Adresse email personnelle.
- **adresse** : Adresse résidentielle.
- **Ville** : Ville de résidence
- **telephone** : Numéro de téléphone.
- ~~**rib** : Coordonnées bancaires.=>classe rémunération~~
- ~~**poste_occupe** : Poste ou fonction occupée.=>Contrat~~
- **date_naissance** : Date de naissance.
- **lieu_naissance** : Lieu de naissance.
- **genre** : Sexe (M/F/Autre).
- **nationalite** : Nationalité.
- **situation_familiale** : Situation familiale.
- **enfants_a_charge** : Nombre d'enfants à charge.
- **permis_conduire** : Booléen indiquant si le salarié possède un permis.
- **mobilite_geographique** : Indique si le salarié est mobile géographiquement.
- **langues_parlees** : Langues que le salarié maîtrise.

- **contact_urgence** : Coordonnées d'une personne à contacter en cas d'urgence.
- **mot_de_passe** : Mot de passe crypté pour l'authentification.
- **etat_compte** : Statut du compte (actif, inactif, suspendu).
- **date_creation** : Date de création du compte.

  mettre dans une classe Rôle de laquelle doit hériter (Employe, RH, admin, configurateur)
- **date_modification** : Date de dernière mise à jour des informations.

**Méthodes (Opérations)**

- **createEmployeSimple(data)**
- **updateEmployeSimple(id, data)**
- **deleteEmployeSimple(id)**
- **getEmployeSimpleById(id)**
- **listEmployeSimples()**
- **authenticate(email, mot_de_passe)**
- **resetPassword(id)**
- **changePassword(id, ancienMotDePasse, nouveauMotDePasse)**

  mettre dans une classe Rôle de la quelle doit hériter (Employe, RH, admin, configurateur)

**Relation**

**EmployeSimple → Contrat**

- **Explication :**
  Un salarié (EmployeSimple) peut avoir plusieurs contrats au cours de sa carrière (pour des évolutions ou renouvellements d'emploi).
- **Cardinalité :**
  - EmployeSimple (1) ↔ Contrat (N)
    (Un salarié peut posséder N contrats, mais chaque contrat se rapporte à un seul salarié.)

**EmployeSimple → Remuneration**

- **Explication :**
  Chaque salarié possède une fiche de rémunération unique, qui résume ses informations de salaire et de paiement.
- **Cardinalité :**
  - EmployeSimple (1) ↔ Remuneration (N)
    (Relation 1 : 1, car un salarié dispose d'une seule fiche de rémunération active.)

**EmployeSimple → FichePaie**

- **Explication :**
  Un salarié peut avoir plusieurs fiches de paie générées au fil des périodes de paie (mensuelles, bimensuelles, etc.).
- **Cardinalité :**
  - EmployeSimple (1) ↔ FichePaie (N)
    (Un salarié peut générer plusieurs bulletins de paie, chacun étant associé à une période distincte.)

**EmployeSimple → Absence**

- **Explication :**
  Chaque absence enregistrée concerne un seul salarié. Un salarié peut cumuler plusieurs enregistrements d'absences (maladie, congés ponctuels, etc.).
- **Cardinalité :**
  - EmployeSimple (1) ↔ Absence (N)

**EmployeSimple → Conges**

- **Explication :**
  Un salarié peut déposer plusieurs demandes de congé au cours d'une année ou sur plusieurs années (et bénéficier de différents types de congés).
- **Cardinalité :**
  - EmployeSimple (1) ↔ Conges (N)

**EmployeSimple → DemandeDocument**

- **Explication :**
  Les demandes de documents (comme les attestations) émises par un salarié se rattachent à ce dernier, et un salarié peut faire plusieurs demandes au fil du temps.

- **Cardinalité :**
    - EmployeSimple (1) ↔ DemandeDocument (N)

**EmployeSimple → ElementVariablePaie**

- **Explication :**
  Un salarié peut avoir plusieurs éléments variables de paie (primes, retenues, indemnités), chacun correspondant à une ligne de calcul dans sa fiche de paie.
- **Cardinalité :**
    - EmployeSimple (1) ↔ ElementVariablePaie (N)

**EmployeSimple → SoldeConges**

- **Explication :**
  Chaque salarié dispose d’un solde de congés pour une année donnée. Si l'on stocke un enregistrement par année, un salarié pourra avoir plusieurs enregistrements (un par année).
- **Cardinalité :**
    - EmployeSimple (1) ↔ SoldeConges (N)
      (Typiquement 1 : N, si on enregistre le solde pour différentes années ou périodes.)

# Table 2 : Administrateur

## Description

La table *Administrateur* stocke les informations des utilisateurs ayant le rôle d'administrateur. Ces comptes, qui ne sont pas considérés comme salariés de l'entreprise, disposent d'un ensemble d'informations réduites par rapport aux employés (ex. pas de données personnelles complètes comme CIN ou adresse).

## Paramètres (Champs)

- **id_administrateur (PK)** : Identifiant unique de l'administrateur.
- **nom** : Nom de l'administrateur.
- **prenom** : Prénom.
- **nom de l'utilisateur** : Identifiant pour se connecter
- **email** : Adresse email professionnelle.
- **telephone** : Numéro de téléphone.
- **mot_de_passe** : Mot de passe crypté.
- **etat_compte** : Statut du compte (actif, inactif, suspendu).
- **date_creation** : Date de création du compte.
- **date_modification** : Date de dernière mise à jour.

## Méthodes (Opérations)

- **createAdministrateur(data)**
- **updateAdministrateur(id, data)**
- **deleteAdministrateur(id)**
- **getAdministrateurById(id)**
- **listAdministrateurs()**
- **authenticate(email, mot_de_passe)**
- **resetPassword(id)**
- **listRHs()**
- **listEmployeesSimples()**
- **listConfigurateurs()**
- **changePassword(id, ancienMotDePasse, nouveauMotDePasse)**

mettre dans une classe Rôle de la quelle doit hériter (Employe, RH, admin, configurateur)

**Relations**

- Admin ⇔ Société (N : N)
  Un administrateur est associé à plusieurs sociétés et inversement.
- ~~Admin ⇔ RH (N : N)~~
  ~~Un Administrateur peut être associé à plusieurs RHs, et un RH peut être suivi ou géré par plusieurs Administrateurs (On doit créer une table d'association)~~
- ~~Admin ⇔ Configurateur (N : N)~~
  ~~Un Administrateur peut être associé à plusieurs Configurateurs, et un Configurateur peut être suivi ou géré par plusieurs Administrateurs (On doit créer une table d'association)~~
- Admin ⇔ (N : N) RÔLE
  Un Administrateur gères les accès des différents profils (Employé, RH, Admin, configurateur)

-

# Table 3 : RH

**Description**

- La table *RH* (Responsables des Ressources Humaines) regroupe les utilisateurs chargés de la gestion des données et opérations RH. Ces comptes nécessitent un ensemble d'informations réduit par rapport aux salariés (pas de données liées à la paie ou au contrat).

**Paramètres (Champs)**

- **id_rh (PK)** : Identifiant unique du responsable RH.
- **nom** : Nom.
- **prenom** : Prénom.
- **nom de l'utilisateur** : Identifiant pour se connecter
- **email** : Adresse email professionnelle.
- **telephone** : Numéro de téléphone.
- **mot_de_passe** : Mot de passe crypté.
- **etat_compte** : Statut du compte.
- **date_creation** : Date de création.
- **date_modification** : Date de mise à jour.

**Méthodes (Opérations)**

- **createRH(data)**
- **updateRH(id, data)**
- **deleteRH(id)**
- **getRHById(id)**
- **listRHs()**
- **authenticate(email, mot_de_passe)**
- **changePassword(id, ancienMotDePasse, nouveauMotDePasse)**
- **resetPassword(id)**

**Relations**

- ~~Admin ⇔ RH (N : N)~~
  - ~~Un Administrateur peut être associé à plusieurs RHs, et un RH peut être~~

~~suivi ou géré par plusieurs Administrateurs (On doit créer une table d'association)~~

- RH ⇔ EmployéSimple (N : N)
  Un RH peut être associé à plusieurs employés simples, et un employé simple peut être suivi ou géré par plusieurs RH. (On doit créer une table d'association)
  RH - Société
  Rh - Congés
  Rh - Demande de document
  A vérifier et compléter

# Table 4 : Configurateur

**Description**

- La table *Configurateur* concerne les utilisateurs en charge de paramétrer le système (ex : mise à jour des barèmes, configuration du système, etc.). Ces comptes disposent d'un minimum d'informations essentielles (identification et authentification).

**Paramètres (Champs)**

- **id_configurateur (PK)** : Identifiant unique du configurateur.
- **nom** : Nom.
- **prenom** : Prénom.
- **email** : Adresse email professionnelle.
- **telephone** : Numéro de téléphone.
- **Nom d'utilisateur**
- **mot_de_passe** : Mot de passe crypté.
- **etat_compte** : Statut du compte.
- **date_creation** : Date de création.
- **date_modification** : Date de dernière mise à jour.

**Méthodes (Opérations)**

- **createConfigurateur(data)**
- **updateConfigurateur(id, data)**
- **deleteConfigurateur(id)**
- **getConfigurateurById(id)**
- **listConfigurateurs()**
- **authenticate(email, mot_de_passe)**
- **changePassword(id, ancienMotDePasse, nouveauMotDePasse)**
- **reset PWD()**

**Relations**

- ~~Admin ⇔ Configurateur (N : N)~~
  - ~~Un Administrateur peut être associé à plusieurs Configurateurs, et un~~

~~Configurateur peut être suivi ou géré par plusieurs Administrateurs (On doit créer une table d'association)~~

- Société ⇔ Configurateur (N : N)
  Une société peut être associée à plusieurs configurateurs et, réciproquement, un configurateur peut être rattaché à plusieurs sociétés.

# Table 5 : Societe.

**Description**

- La table *Societe* stocke les informations relatives aux sociétés avec lesquelles l'ERP RH interagit. Elle contient les données administratives et légales essentielles pour l'établissement des contrats et autres opérations financières ou déclaratives.

**Paramètres (Champs)**

- **id_societe (PK)** : Identifiant unique de la société.
- **nom_societe** : Nom complet de la société.
- **adresse** : Adresse postale de la société sans la ville.
- **ville** : Ville où est située la société.
- **identifiant_fiscal** : Numéro fiscal attribué par l'administration.
- **numero_cnss** : Numéro d'immatriculation à la CNSS.
- **numero_ice** : Numéro ICE de la société.
- **numero_rc** : Numéro du Registre du Commerce.
- **date_debut** : Date de début d'activité ou de validité des informations.
- **date_fin** : Date de fin de validité (si applicable).
- **nom :** Nom de la banque associée à la société.
- **rib :** Relevé d'Identité Bancaire attribué au compte de la société.
- **bic :** Code d'identification international (SWIFT) de la banque.

**Méthodes (Opérations)**

- **createSociete(data)**

  Crée une nouvelle société avec les informations fournies.
- **updateSociete(id,data)**

  Met à jour les informations d'une société existante.
- **deleteSociete(id)**

  Supprime la société identifiée par son ID.
- **getSocieteById(id)**

  Récupère les informations d'une société spécifique.
- **listSocietes()**

  Retourne la liste de toutes les sociétés enregistrées.

**Relations**

- **Societe → Contrat (1 : N)**

  *Une société peut être associée à plusieurs contrats.*

- **Société ⇔ RH (N : N)**

  *Une société peut être associée à plusieurs RHs et, réciproquement, un RH peut être rattaché à plusieurs sociétés.( On doit créer une table d'association )*

- **Société ⇔ Configurateur (N : N)**

  *Une société peut être associée à plusieurs configurateurs et, réciproquement, un configurateur peut être rattaché à plusieurs sociétés.( On doit créer une table d'association )*

- **Admin ⇔ Société (N : N)**

  *Un administrateur est associé à plusieurs sociétés et inversement.( On doit créer une table d'association )*

# Table 6 : Contrat

**Description**

- La table *Contrat* stocke toutes les informations relatives aux contrats de travail entre un employé et une société. Elle contient les détails de l'embauche, les dates d'ancienneté et de fin de contrat ainsi que les conditions spécifiques de l'emploi.

**Paramètres (Champs)**

- **id_contrat (PK)** : Identifiant unique du contrat.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé concerné par le contrat.
- **id_societe (FK → Societe)** : Référence à la société avec laquelle le contrat est établi.
- **Id_categoriesalariale (FK→ CatégorieSalariale )**: Référence à la catégorie du salarié correspondante au Contrat.
- **Id_ statutsalarial (FK→ StatutSalarial )**: Référence au statut de l'employé correspondant au Contrat
- **Id_unitéorganisationnelle (FK→ uniteOrganisationnelle)** : Référence à l'unité organisationnelle correspondante au contrat.
- **date_embauche** : Date d'embauche de l'employé.
- **date_anciennete** : Date marquant l'ancienneté de l'employé.
- **date_fin_contrat** : Date de fin du contrat (si applicable).
- **id_typecontrat (FK)** : Type de contrat (CDI, CDD, etc.).
- **fonction** : Fonction ou poste occupé.
- **departement** : Département de travail.
- **service** : Service ou unité d'affectation.
- **type_profil** : Type de profil (ex. Cadre, Ouvrier, etc.).
- **numero_CIMR** : Numéro CIMR (retraite) s'il existe.
- **numero_mutuelle** : Référence à la mutuelle.
- **responsable_hierarchique** : Identifiant du responsable hiérarchique.
- **mode_travail** : Mode de travail (présentiel, télétravail, etc.).
- **periode_essai** : Durée ou période d'essai du contrat.

- **conditions_specifiques** : Conditions particulières associées au contrat.

**Méthodes (Opérations)**

- **createContrat(data)**

  Crée un nouveau contrat avec les informations fournies.
- **updateContrat(id,data)**

  Met à jour les informations d'un contrat existant.
- **deleteContrat(id)**

  Supprime le contrat identifié par son ID.
- **getContratById(id)**

  Récupère les détails d'un contrat spécifique.
- **listContratsByEmploye(employeId)**

  Liste tous les contrats d'un employé.
- **listContratsBySociete(societeId)**

  Liste tous les contrats associés à une société.

**Relations**

- **Contrat → EmployeSimple (N : 1)**

  *Un contrat appartient à un seul employé.*
- **Contrat → Societe (N : 1)**

  *Un contrat appartient à une seule société.*
- **Contrat → TypeContrat (N : 1)**
  *Un contrat est associé à un seul type de contrat, tandis qu'un type de contrat peut regrouper plusieurs contrats.*

# Table 7 : Remuneration

**Description**

La table *Remuneration* enregistre les informations de rémunération liées à chaque employé. Chaque employé dispose d'une fiche de rémunération unique qui contient les détails de son salaire et de ses modalités de paiement.

**Paramètres (Champs)**

- **id_remuneration (PK)** : Identifiant unique de la fiche de rémunération.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé concerné.
- **rib** : Coordonnées bancaires pour le paiement.
- **bic :** Code d'identification international (SWIFT) de la banque.
- **banque** : Nom de la banque.
- **salaire_base** : Salaire de base de l'employé.
- **taux_activite** : Taux d'activité (par exemple, temps plein/partiel).
- **mode_paiement** : Mode de paiement (virement, chèque, espèces.).

**Méthodes (Opérations)**

- **createRemuneration(data)**

  Crée une fiche de rémunération pour un employé.
- **updateRemuneration(id,data)**

  Met à jour les informations de la rémunération.
- **getRemunerationByEmploye(employeId)**

  Récupère la fiche de rémunération d'un employé spécifique.

**Relations**

- **Remuneration → EmployeSimple (0 : 1)**

  *Chaque employé possède une fiche de rémunération active unique.*

# Table 8 : RubriquePaie

**Description**

La table *RubriquePaie* contient les rubriques ou catégories de paie. Elle définit les différents éléments de salaire tels que les gains, les primes, ou les retenues, qui pourront être associés aux éléments variables de paie et aux primes/indemnités.

**Paramètres (Champs)**

- **id_rubrique (PK)** : Identifiant unique de la rubrique.
- **designation** : Nom ou libellé de la rubrique (ex : Salaire de base, Prime de transport).
- **type_rubrique** : Type de rubrique (par exemple, "gain", "retenue", "cotisation"). A voir si on aura besoin de créer une table TypeRubriquePaie !

**Méthodes (Opérations)**

- **createRubrique(data)**
  Crée une nouvelle rubrique de paie avec les informations fournies.
- **updateRubrique(id,data)**
  Met à jour les détails d'une rubrique existante.
- **deleteRubrique(id)**
  Supprime une rubrique identifiée par son ID.
- **getRubriqueById(id)**
  Récupère les informations d'une rubrique spécifique.
- **listRubriques()**
  Liste toutes les rubriques de paie enregistrées.

**Relations**

- **RubriquePaie → ElementVariablePaie (0 : N)**
  *Une rubrique de paie peut être associée à plusieurs éléments variables de paie.*
- **RubriquePaie → PrimeIndemnite (0 : N)**
  *Une rubrique de paie peut être liée à plusieurs primes ou indemnités.*
- **RubriquePaie ⇔ Cotisation (N : N)**
  *Une rubrique paie peut-être liée à plusieurs cotisations et, inversement, une*

*cotisation peut concerner plusieurs rubriques paie. (On doit créer une table d'association ).*

- **RubriquePaie ⇔ Absence (N : N)**

  *Une rubrique paie peut s'appliquer à plusieurs types d'absences, tandis qu'un enregistrement d'absence peut être relié à plusieurs rubriques paie*

- **RubriquePaie ⇔ Rémunération (N : N)**

  *Une rubrique paie peut-être associée à plusieurs rémunérations, et une rémunération peut être constituée de plusieurs rubriques paie.*

# Table 9 : ElementVariablePaie

**Description**

La table *ElementVariablePaie* enregistre les éléments variables du traitement de la paie. Chaque enregistrement correspond à un ajustement ou une composante spécifique (comme une prime, une retenue ou une indemnité) qui est associée à un employé et classifiée par une rubrique de paie.

**Paramètres (Champs)**

- **id_element (PK)** : Identifiant unique de l'élément variable.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé concerné.
- **id_rubrique (FK → RubriquePaie)** : Référence à la rubrique de paie associée.
- **nombre** : Quantité ou nombre d'unités (si applicable).
- **montant** : Montant de l'élément variable.
- **periode** : Période de paie concernée (ex : mois et année).

**Méthodes (Opérations)**

- **createElementVariablePaie(data)**
  Crée un nouvel élément variable de paie pour un employé.
- **updateElementVariablePaie(id,data)**
  Met à jour les informations d'un élément variable existant.
- **deleteElementVariablePaie(id)**
  Supprime l'élément variable identifié par son ID.
- **listElementVariablePaieByEmploye(employeId)**
  Liste tous les éléments variables de paie pour un employé spécifique.

**Relations**

- **ElementVariablePaie → EmployeSimple (N : 1)**
  *Un élément variable de paie concerne un seul employé.*
- **ElementVariablePaie → RubriquePaie (N : 1)**
  *Un élément variable de paie est associé à une seule rubrique de paie.*
- **ElementVariablePaie → PériodePaie (N : 1)**
  *Un élément variable de paie est rattaché à une seule période de paie, tandis qu'une période de paie peut regrouper plusieurs éléments variables.*

# Table 10 : Absence

**Description**

La table *Absence* enregistre les périodes d'absence des employés. Chaque enregistrement correspond à une absence spécifique, qui est associée à un type d'absence pour clarifier la raison (congé, maladie, etc.).

**Paramètres (Champs)**

- **id_absence (PK)** : Identifiant unique de l'absence.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé concerné par l'absence.
- **id_type_absence (FK → TypeAbsence)** : Référence au type d'absence.
- **date_debut** : Date de début de l'absence.
- **date_fin** : Date de fin de l'absence.

**Méthodes (Opérations)**

- **createAbsence(data)**
  Crée un nouvel enregistrement d'absence.
- **updateAbsence(id,data)**
  Met à jour les détails d'une absence existante.
- **deleteAbsence(id)**
  Supprime un enregistrement d'absence par son ID.
- **listAbsencesByEmploye(employeId)**
  Liste toutes les absences pour un employé donné.

**Relations**

- **Absence → EmployeSimple (N : 1)**
  *Une absence concerne un seul employé.*
- **Absence → TypeAbsence (N : 1)**
  *Une absence correspond à un seul type d'absence.*
- **Absence → PériodePaie (N : 1)**
  *Un enregistrement d'absence est rattaché à une seule période de paie, tandis qu'une période de paie peut regrouper plusieurs absences.*

# Table 11 : Conges

**Description**

La table *Conges* gère les demandes de congé des employés. Elle stocke toutes les informations relatives aux congés demandés, y compris le nombre de jours, le motif et l'état de la demande, ainsi que l'identifiant du validateur (RH).

**Paramètres (Champs)**

- **id_conge (PK)** : Identifiant unique de la demande de congé.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé qui fait la demande.
- **type_conge** : Type de congé demandé (congé payé, récupération.).
- **date_debut** : Date de début du congé.
- **date_fin** : Date de fin du congé.
- **nb_jours** : Nombre de jours de congé calculés.
- **motif** : Motif ou raison du congé.
- **piece_jointe** : Lien ou référence à un document justificatif (si applicable).
- **etat_conge** : État de la demande (en attente, validé, refusé).
- **motif_refus** : Motif du refus (le cas échéant).
- **date_demande** : Date à laquelle le congé a été demandé.
- **date_traitement** : Date à laquelle le congé a été traité.
- **valide_par (FK → RH)** : Référence à (RH) qui valide le congé.

**Méthodes (Opérations)**

- **createConge(data)**

  Crée une nouvelle demande de congé.
- **updateConge(id,data)**

  Met à jour les informations d'une demande de congé existante.
- **deleteConge(id)**

  Supprime une demande de congé par son ID.
- **getCongeById(id)**

  Récupère les détails d'une demande de congé spécifique.
- **listCongesByEmploye(employeId)**

  Liste toutes les demandes de congé d'un employé.

- **validateConge(id,validatorId)**

  Valide une demande de congé en indiquant l'ID du validateur .

**Relations**

- **Conges → Employe (N : 1)**

  *Une demande de congé concerne un seul employé.*
- **Conges → RH (valide_par) (N : N)**

  *Une demande de congé est validée par plusieurs responsables RH ( un seul à la fois).*
- **Congés → PériodePaie (1 : N)**

  *Un congé peut concerner plusieurs périodes de paie, tandis qu'une période de paie peut être associée à un seul congé.*

# Table 12 : SoldeConges

**Description**

La table *SoldeConges* stocke le solde de congés de chaque employé pour une année donnée. Elle permet de suivre le nombre total de jours de congé, les jours déjà pris et le solde restant.

**Paramètres (Champs)**

- **id_solde (PK)** : Identifiant unique du solde de congés.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé concerné.
- **solde_total** : Nombre total de jours de congé attribués pour l'année.
- **solde_pris** : Nombre de jours de congé déjà utilisés.
- **solde_restant** : Nombre de jours de congé restants.
- **annee** : Année de référence pour le solde de congés.
- **moisReporte** : indique à partir de quel mois de l'année on effectue le report des congés non pris de l'année précédente vers l'année en cours.

**Méthodes (Opérations)**

- **updateSoldeConges(employeId,newSolde)**
  Met à jour le solde de congés pour un employé.
- **getNewSoldeConges(employeId, nombreAbsences)**
  Calcule le nouveau solde de congés d'un employé en prenant en compte le nombre d'absences.
- **getSoldeCongesByEmploye(employeId)**
  Récupère le solde de congés d'un employé spécifique.

**Relations**

- **SoldeConges → EmployeSimple (N : 1)**
  *Un solde de congés appartient à un seul employé.*

# Table 13 : FichePaie

**Description**

La table *FichePaie* enregistre les bulletins de paie générés pour chaque employé. Elle contient les informations relatives à la période de paie, le lien vers le document PDF du bulletin et la date de génération.

**Paramètres (Champs)**

- **id_fiche (PK)** : Identifiant unique de la fiche de paie.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé concerné.
- **periodeAnnee** : Période de paie concernée en année.
- **periodeMois** : Période de paie concernée en mois.
- **lien_pdf** : Lien ou chemin d'accès au fichier PDF du bulletin.
- **date_generation** : Date à laquelle la fiche de paie a été générée.

**Méthodes (Opérations)**

- **createFichePaie(data)**

  Crée une nouvelle fiche de paie pour un employé.
- **getFichePaieByEmploye(employeId)**

  Récupère la fiche de paie d'un employé spécifique.
- **listFichePaieByEmploye(employeId)**

  Liste toutes les fiches de paie d'un employé.

**Relations**

- **FichePaie → EmployeSimple (N : 1)**

  *Une fiche de paie appartient à un seul employé.*
- **FichePaie → PeriodePaie (N : 1)**

  *Une fiche de paie appartient à une seule période de paie.*
- **DemandeDocument → FichePaie (N : 1)**

  *Chaque demande de document est associée à une seule fiche de paie. Inversement, une fiche de paie peut être liée à plusieurs demandes de document.*

# Table 14 : DemandeDocument

**Description**

La table *DemandeDocument* gère les demandes faites par les employés pour obtenir des documents administratifs, tels que des attestations de travail ou de salaire. Elle enregistre le type de document, le motif de la demande, le format souhaité, et le lien de téléchargement une fois validée.

**Paramètres (Champs)**

- **id_demande (PK)** : Identifiant unique de la demande.
- **id_employe (FK → EmployeSimple)** : Référence à l'employé qui fait la demande.
- **id_attestation (FK → Attestation)** : Référence à l'attestation concernée (si applicable).
- **type_document** : Type de document demandé (ex : attestation de travail, attestation de salaire).
- **motif** : Motif de la demande.
- **format_document** : Format souhaité (PDF, Impression Papier).
- **etat_demande** : État de la demande (en attente, en cours, validée, refusée).
- **motif_refus** : Motif du refus, le cas échéant.
- **date_demande** : Date de la demande.
- **date_traitement** : Date de traitement de la demande.
- **lien_telechargement** : Lien vers le document généré.

**Méthodes (Opérations)**

- **createDemandeDocument(data)**
  Crée une nouvelle demande de document.
- **updateDemandeDocument(id,data)**
  Met à jour les informations d'une demande existante.
- **deleteDemandeDocument(id)**
  Supprime une demande par son ID.
- **getDemandeDocumentById(id)**
  Récupère les détails d'une demande spécifique.
- **listDemandeDocumentsByEmploye(employeId)**
  Liste toutes les demandes de documents d'un employé.

- **markAsProcessed(id)**

  Marque la demande comme traitée.

**Relations**

- **DemandeDocument → EmployeSimple (N : 1)**

  *Une demande de document est effectuée par un seul employé.*
- **DemandeDocument → Attestation (N : 1)**

  *Une demande de document concerne une seule attestation prédéfinie.*

# Table 15 : Attestation

**Description**

La table *Attestation* stocke les différentes attestations administratives disponibles dans le système (attestation de travail, attestation de salaire, etc.). Ces documents servent de modèle pour répondre aux demandes des employés.

**Paramètres (Champs)**

- **id_attestation (PK)** : Identifiant unique de l'attestation.
- **Id_type_attestation (FK) :** Référence au type d'attestation concernée
- **nom_attestation** : Nom ou libellé de l'attestation.

**Méthodes (Opérations)**

- **createAttestation(data)**

  Crée une nouvelle attestation dans le système.
- **updateAttestation(id,data)**

  Met à jour les informations d'une attestation existante.
- **deleteAttestation(id)**

  Supprime une attestation par son ID.
- **getAttestationById(id)**

  Récupère les détails d'une attestation spécifique.
- **listAttestations()**

  Liste toutes les attestations disponibles.

**Relations**

- **Attestation → DemandeDocument (1 : N)**

  *Une attestation peut être associée à plusieurs demandes de documents.*
- **TypeAttestation → Attestation (1 : N)**

  *Chaque type d'attestation peut être utilisé par plusieurs attestations concrètes dans la table* ***Attestation****. Ainsi, dans la table* ***Attestation****, le champ Id_type_attestation sera une clé étrangère (FK) référencée à partir de* ***TypeAttestation****.*

# Table 16 : TypeAttestation

**Description**

La table **TypeAttestation** stocke les différents types ou catégories d'attestations administratives qui pourront être utilisées en tant que modèles (par exemple, attestation de travail, attestation de salaire, etc.). Cette table permet de définir les caractéristiques générales de chaque type d'attestation et d'assurer une cohérence lors de leur utilisation dans le système.

**Paramètres (Champs)**

- **id_type_attestation (PK)** : Identifiant unique pour le type d'attestation.
- **nom_type_attestation** : Nom ou libellé du type d'attestation (par exemple, « Attestation de travail »).
- **description** : Description détaillée du type d'attestation, permettant d'expliquer son usage ou ses spécificités.

**Méthodes (Opérations)**

- **createTypeAttestation(data)**

  Crée un nouveau type d'attestation dans le système en enregistrant les informations fournies.
- **updateTypeAttestation(id,data)**

  Met à jour les informations d'un type d'attestation existant identifié par son id_type_attestation.
- **deleteTypeAttestation(id)**

  Supprime un type d'attestation du système à l'aide de son identifiant.
- **getTypeAttestationById(id)**

  Récupère les détails d'un type d'attestation spécifique en utilisant son identifiant.
- **listTypeAttestations()**

  Liste tous les types d'attestations disponibles dans le système.

**Relations**

- **TypeAttestation → Attestation (1 : N)**

  Chaque type d'attestation peut être utilisé par plusieurs attestations concrètes dans la table **Attestation**. Ainsi, dans la table **Attestation**, le champ

Id_type_attestation sera une clé étrangère (FK) référencée à partir de **TypeAttestation**.

# Table 17 : ParametresSysteme

**Description**

La table *ParametresSysteme* contient les réglages globaux de l'application. Elle définit les paramètres du système tels que le fuseau horaire, les formats de date et d'heure, l'adresse email système et l'état des notifications. Ce sont des paramètres globaux qui influent sur le fonctionnement et l'affichage de l'application.

**Paramètres (Champs)**

- **id_parametre (PK)** : Identifiant unique du paramètre système.
- **Id_configurateur (FK)** : Référence au configurateur qui a effectué la modification.
- **fuseau_horaire** : Fuseau horaire utilisé par le système.
- **format_date** : Format de date (ex : JJ/MM/AAAA).
    - **format_heure** : Format d'heure (ex : HH:MM).
- **email_systeme** : Adresse email utilisée par le système pour les notifications.
- **notifications_actives** : Booléen indiquant si les notifications système sont activées.
- **date_modification** : Date de la dernière mise à jour des paramètres.

**Méthodes (Opérations)**

- **createParametresSysteme(data)**

  Crée une nouvelle configuration de paramètres système (généralement lors de l'initialisation).
- **updateParametresSysteme(id, data)**

  Met à jour les paramètres existants.
- **getParametresSysteme()**

  Récupère la configuration actuelle des paramètres système.

**Relations**

- **ParametresSysteme → Configurateur (N : 1)**

  *Un paramétrage de système peut être effectuée par un configurateur.*

# Table 18 : SauvegardeBDD

**Description**

La table *SauvegardeBDD* enregistre les informations relatives aux sauvegardes de la base de données. Chaque enregistrement représente une sauvegarde effectuée par un utilisateur (typiquement un administrateur ou un RH), permettant de restaurer l'état de la base en cas de besoin.

**Paramètres (Champs)**

- **id_sauvegarde (PK)** : Identifiant unique de la sauvegarde.
- **nom_fichier** : Nom du fichier de sauvegarde.
- **emplacement** : Chemin ou emplacement de stockage de la sauvegarde.
- **date_sauvegarde** : Date à laquelle la sauvegarde a été effectuée.
- **cree_par (FK → Administrateur)** : Référence à l'administrateur qui a effectué la sauvegarde.

**Méthodes (Opérations)**

- **createSauvegardeBDD(data)**
  Enregistre une nouvelle sauvegarde de la base de données.
- **deleteSauvegardeBDD(id)**
  Supprime une sauvegarde en fonction de son identifiant.
- **listSauvegardes()**
  Liste toutes les sauvegardes existantes.

**Relations**

- **SauvegardeBDD → Administrateur (N : 1)**
  *Une sauvegarde est effectuée par un administrateur.*

# Table 19 : PeriodePaie

**Description**

La table *PeriodePaie* gère les périodes de paie. Chaque enregistrement correspond à une période de paie (généralement définie par le mois et l'année) et indique si la période est ouverte ou fermée, ce qui impacte la saisie et la génération des déclarations.

**Paramètres (Champs)**

- **id_periode (PK)** : Identifiant unique de la période de paie.
- **mois** : Mois de la période de paie.
- **annee** : Année de la période de paie.
- **etat_periode** : État de la période (ouverte, fermée).

**Méthodes (Opérations)**

- **createPeriodePaie(data)**

  Crée une nouvelle période de paie.
- **closePeriodePaie(id)**

  Ferme une période de paie pour bloquer les saisies et permettre la génération des déclarations.
- **openPeriodePaie(id)**

  Ouvre ou réouvre une période de paie.
- **getPeriodePaieById(id)**

  Récupère les détails d'une période de paie spécifique.
- **listPeriodePaie()**

  Liste toutes les périodes de paie existantes.

**Relations**

- **PeriodePaie → DeclarationSociale (1 : N)**

  *Une période de paie peut comporter plusieurs déclarations sociales associées.*
- **FichePaie → PeriodePaie (N : 1)**

  *Chaque fiche de paie est associée à une seule période de paie, tandis qu'une période de paie peut contenir plusieurs fiches de paie (relation 1 : N du côté de PeriodePaie).*

- **ElementVariablePaie → PériodePaie (N : 1)**
  *Un élément variable de paie est rattaché à une seule période de paie, tandis qu'une période de paie peut regrouper plusieurs éléments variables.*

- **Congés → PériodePaie (1 : N)**
  *Un congé peut concerner plusieurs périodes de paie, tandis qu'une période de paie peut être associée à un seul congé.*

- **Absence → PériodePaie (N : 1)**
  *Un enregistrement d'absence est rattaché à une seule période de paie, tandis qu'une période de paie peut regrouper plusieurs absences.*

-

# Table 20 : DeclarationSociale

**Description**

La table *DeclarationSociale* gère les déclarations sociales pour une période de paie donnée. Elle enregistre les déclarations concernant des organismes sociaux (comme la CNSS, l'IR, la CIMR, etc.), incluant l'état de génération et le fichier produit.

**Paramètres (Champs)**

- **id_declaration (PK)** : Identifiant unique de la déclaration sociale.
- **id_periode (FK → PeriodePaie)** : Référence à la période de paie concernée.
- **type_declaration** : Type de déclaration (ex : CNSS, IR, CIMR).
- **etat_generation** : État de la déclaration (en attente, générée, etc.).
- **date_generation** : Date à laquelle la déclaration a été générée.
- **fichier_genere** : Chemin ou référence au fichier généré.

**Méthodes (Opérations)**

- **createDeclarationSociale(data)**
  Crée une nouvelle déclaration sociale pour une période donnée.
- **updateDeclarationSociale(id, data)**
  Met à jour les informations d'une déclaration sociale existante.
- **getDeclarationSocialeByPeriode(periodeId)**
  Récupère la ou les déclarations sociales d'une période spécifique.
- **listDeclarationSocialeByPeriode(periodeId)**
  Liste toutes les déclarations sociales associées à une période.

**Relations**

- **DeclarationSociale → PeriodePaie (N : 1)**
  *Une déclaration sociale concerne une seule période de paie.*

# Table 21 : Cotisation

**Description**

La table *Cotisation* contient les informations sur les cotisations sociales et fiscales. Chaque enregistrement définit un type de cotisation, incluant ses taux à la charge de l'employé et de l'employeur, ainsi que les plafonds applicables.

**Paramètres (Champs)**

- **id_cotisation (PK)** : Identifiant unique de la cotisation.
- **type_cotisation** : Catégorie de la cotisation (ex : CNSS, CIMR, AMO, etc.).
- **id_type_cotisation (FK → TypeCotisation)** : Référence au type de cotisation.
- **taux_salarial** : Pourcentage appliqué à la charge du salarié.
- **taux_patronal** : Pourcentage appliqué à la charge de l'employeur.
- **plafond_salarial** : Montant plafond pour la part salariale.
- **plafond_patronal** : Montant plafond pour la part patronale.
- **date_debut** : Date d'entrée en vigueur de la cotisation.
- **date_fin** : Date de fin de validité (si applicable).
- **description** : Description ou informations complémentaires sur la cotisation.

**Méthodes (Opérations)**

- **createCotisation(data)**

  Crée une nouvelle cotisation avec les détails fournis.
- **updateCotisation(id,data)**

  Met à jour les informations d'une cotisation existante.
- **deleteCotisation(id)**

  Supprime une cotisation en fonction de son identifiant.
- **getCotisationById(id)**

  Récupère les détails d'une cotisation spécifique.
- **listCotisations()**

  Liste toutes les cotisations enregistrées.

**Relations**

**Cotisation → TypeCotisation : N : 1**

*Un enregistrement dans la table **Cotisation** appartient à un seul type de cotisation défini dans la table **TypeCotisation***

# Table 22 : TypeCotisation

## Description

La table *TypeCotisation* permet de définir les différents types de cotisations sociales et fiscales disponibles dans le système.
Elle est utilisée pour paramétrer les caractéristiques des cotisations (par exemple, description, période de validité) et peut être référencée par la table Cotisation pour associer chaque enregistrement à un type prédéfini.

## Paramètres (Champs)

- **id_type_cotisation (PK)** : Identifiant unique du type de cotisation.
- **nom_cotisation** : Libellé du type de cotisation (ex : CNSS, CIMR, AMO).
- **code_cotisation** : Code abrégé pour identifier le type de cotisation.
- **description** : Description détaillée du type de cotisation.
- **date_debut** : Date de début de validité du type de cotisation.
- **date_fin** : Date de fin de validité (facultatif, si applicable).

## Méthodes (Opérations)

- **createTypeCotisation(data)**
  Crée un nouveau type de cotisation avec les informations fournies.
- **updateTypeCotisation(id, data)**
  Met à jour les informations d'un type de cotisation existant.
- **deleteTypeCotisation(id)**
  Supprime un type de cotisation par son identifiant.
- **getTypeCotisationById(id)**
  Récupère les détails d'un type de cotisation spécifique.
- **listTypeCotisations()**
  Liste tous les types de cotisation enregistrés.

## Relations

- **TypeCotisation → Cotisation : 1 : N**
  *Un type de cotisation peut être associé à plusieurs enregistrements dans la table Cotisation.*

# Table 23 : TypeAbsence

## Description

La table *TypeAbsence* définit les différents types d'absences utilisés pour catégoriser les périodes d'absence des employés. Elle précise, par exemple, si une absence est rémunérée ou nécessite un justificatif.

## Paramètres (Champs)

- **id_type_absence (PK)** : Identifiant unique du type d'absence.
- **nom_absence** : Libellé du type d'absence (ex : Congé payé, Maladie, Absence injustifiée).
- **code_absence** : Code interne ou abrégé pour identifier le type d'absence.
- **justificatif_requis** : Indique si un justificatif est nécessaire (booléen ou liste).
- **absence_remuneree** : Indique si l'absence est rémunérée (booléen).
- **impact_solde_conge** : Précise si l'absence décompte du solde de congé (booléen).
- **date_debut** : Date d'entrée en vigueur du type d'absence.
- **date_fin** : Date de fin de validité (si applicable).

## Méthodes (Opérations)

- **createTypeAbsence(data)**
  Crée un nouveau type d'absence avec les informations fournies.
- **updateTypeAbsence(id, data)**
  Met à jour un type d'absence existant.
- **deleteTypeAbsence(id)**
  Supprime un type d'absence par son identifiant.
- **getTypeAbsenceById(id)**
  Récupère les détails d'un type d'absence spécifique.
- **listTypeAbsence()**
  Liste tous les types d'absence enregistrés.

## Relations

- **TypeAbsence → Absence (1 : N)**
  *Un type d'absence peut être utilisé pour plusieurs enregistrements d'absence.*

# Table 24 : PrimeIndemniteRetenue

## Description

La table *PrimeIndemniteRetenue* enregistre les primes et indemnités qui peuvent être appliquées aux salariés. Elle permet de paramétrer différentes formes de rémunération variable (montant fixe, pourcentage ou valeur unitaire) et de définir leur application selon des critères précis.

## Paramètres (Champs)

- **id_prime (PK)** : Identifiant unique de la prime ou indemnité.
- **id_rubrique (FK → RubriquePaie)** : Référence à la rubrique de paie à laquelle la prime est associée.
- **Id_typeprime (FK → TypePrimeIndemniteRetenue) :** Référence à la table TypePrimeIndemniteRetenue
- **nom_prime** : Nom ou libellé de la prime/indemnité.
- **code_prime** : Code interne utilisé pour identifier la prime.
- **rubrique_source** : Rubrique sur laquelle le pourcentage est basé (si applicable).
- **valeur_unitaire** : Montant par unité (utilisé si le type est "Nombre").
- **date_debut** : Date de début de validité de la prime.
- **date_fin** : Date de fin de validité (si applicable).

## Méthodes (Opérations)

- **createPrimeIndemniteRetenue(data)**
  Crée une nouvelle prime ou indemnité avec les paramètres définis.
- **updatePrimeIndemniteRetenue (id, data)**
  Met à jour les informations d'une prime existante.
- **deletePrimeIndemniteRetenue (id)**
  Supprime une prime ou indemnité par son identifiant.
- **getPrimeIndemniteRetenueById(id)**
  Récupère les détails d'une prime spécifique.
- **listPrimeIndemniteRetenueByRubrique(rubriqueId)**
  Liste toutes les primes associées à une rubrique donnée.

## Relations

- **PrimeIndemnite → RubriquePaie (N : 1)**
  *Une prime ou indemnité appartient à une seule rubrique de paie.*
- **PrimeIndemniteRetenue → TypePrimeIndemniteRetenue : N : 1**
- **PrimeIndemniteRetenue → EmployeSimple (N : 1)**
  *Chaque enregistrement dans la table PrimeIndemniteRetenue est associé à un seul employé dans la table EmployeSimple.*

- **PrimeIndemniteRetenue → PeriodePaie (N : 1)**

# Table 25 : TypePrimeIndemniteRetenue

**Description**

- *La table TypePrimeIndemniteRetenue sert à définir les différents types de primes, indemnités ou retenues dans le système. Elle permet de paramétrer le type (Prime, Indemnité, ou Retenue), le mode de calcul du montant (Nombre, Montant fixe ou Pourcentage) ainsi que les cotisations auxquelles ce type est soumis (CNSS, AMO, CIMR, IR).*

**Paramètres (Champs)**

- ***id_typeprime (PK)** : Identifiant unique du type de prime, indemnité ou retenue.*
- ***type** : Indique le type d'enregistrement, avec les valeurs possibles : "Prime", "Indemnité" ou "Retenue".*
- ***unité** : Type de l'unité utilisé (par exemple : "Nombre", "Montant fixe", "Pourcentage").*
- ***Nombre** : Quantité ou coefficient saisi lors du calcul si l'unité choisie est "Nombre".*
- ***montant_fixe** : Valeur fixe applicable si le type de montant est "Montant fixe".*
- ***taux_pourcentage** : Pourcentage appliqué si le type de montant est "Pourcentage".*
- ***soumisCNSS** : Booléen indiquant si le montant est soumis à la cotisation CNSS.*
- ***soumisAMO** : Booléen indiquant si le montant est soumis à la cotisation AMO.*
- ***soumisCIMR** : Booléen indiquant si le montant est soumis à la cotisation CIMR.*
- ***soumisIR** : Booléen indiquant si le montant est soumis à l'impôt sur le revenu (IR).*

**Méthodes (Opérations)**

- ***createTypePrimeIndemniteRetenue(data)***
  *Crée un nouveau type de prime, indemnité ou retenue avec les informations fournies.*
- ***updateTypePrimeIndemniteRetenue(id, data)***
  *Met à jour les informations d'un type de prime, indemnité ou retenue existant.*

- ***deleteTypePrimeIndemniteRetenue(id)***
  *Supprime un enregistrement de type prime, indemnité ou retenue par son identifiant.*
- ***getTypePrimeIndemniteRetenueById(id)***
  *Récupère les détails d'un type de prime, indemnité ou retenue spécifique.*
- ***listTypePrimeIndemniteRetenue()***
  *Liste tous les enregistrements de types de primes, indemnités et retenues enregistrés.*

**Relations**

- **TypePrimeIndemniteRetenue → PrimeIndemniteRetenue : 1 : N**

  *Chaque enregistrement dans **TypePrimeIndemniteRetenue** (par exemple, "Prime de transport", "Indemnité de panier", etc.) pourra être associé à plusieurs enregistrements dans **PrimeIndemniteRetenue** qui appliquent ce type de prime/indemnité/retenue aux salariés.*

# Table 26 : JourFerie

**Description**

La table *JourFerie* gère la liste des jours fériés officiels. Elle permet de définir les jours fériés ainsi que leur périodicité (répétition annuelle ou non) pour être utilisés lors du calcul des congés et de la planification du calendrier.

**Paramètres (Champs)**

- **id_jour_ferie (PK)** : Identifiant unique du jour férié.
- **Id_parametreur (FK)** : Référence au configurateur qui a effectué la modification
- **nom_jour** : Nom ou libellé du jour férié (ex : Aid El Fitr, 1er Mai).
- **date_debut** : Date de début du jour férié.
- **date_fin** : Date de fin du jour férié (si applicable, par exemple pour un jour férié sur plusieurs jours).
- **recurrence_annuelle** : Booléen indiquant si le jour férié se répète chaque année.

**Méthodes (Opérations)**

- **createJourFerie(data)**
  Crée un nouvel enregistrement pour un jour férié.
- **updateJourFerie(id, data)**
  Met à jour les informations d'un jour férié existant.
- **deleteJourFerie(id)**
  Supprime un jour férié par son identifiant.
- **getJourFerieById(id)**
  Récupère les détails d'un jour férié spécifique.
- **listJoursFeries()**
  Liste tous les jours fériés définis.

**Relations**

- **JourFerie → Configurateur (N : 1)**

  *Chaque jour férié est enregistré ou modifié par un seul paramétreur.*
  *Un paramétreur peut être à l'origine de plusieurs ajouts ou modifications de jours fériés.*

# Table 27 : Mesure

## Description

La table *Mesure* enregistre l'historique de vie d'un salarié au sein de l'entreprise, en tracant les événements importants tels que l'embauche, les changements de contrat, les promotions ou les sorties**.**

## Paramètres (Champs)

- **id_mesure (PK)** : Identifiant unique de l'événement.
- **id_employe (FK → EmployeSimple)** : Référence au salarié concerné.
- **id_typemesure (FK → TypeMesure ):** Référence au type de mesure
- **description** : Description complète de la mesure
- **date_debut** : Date de début de validité ou d'application de l'événement.
- **date_fin** : Date de fin de validité ou d'application de l'événement.

## Méthodes (Opérations)

- **createMesure(data)**
- **updateMesure(id, data)**
- **deleteMesure(id)**
- **getMesureById(id)**
- **listMesureByEmploye(employeId)**

## Relations

- **EmployeSimple → Mesure (1 : N)**
- **TypeMesure → Mesure (1 : N)**
  *Un type de mesure peut être associé à plusieurs enregistrements dans la table Mesure.*
  *Chaque mesure appliquée à un salarié est rattachée à un seul type défini dans TypeMesure.*
- **Mesure → MotifMesure (1 : N)**
  *Une mesure peut être associé à plusieurs enregistrements dans la table MotifMesure.*
  *Chaque motif de mesure est rattachée à un seul type défini dans la table Mesure.*

# Table 28 : TypeMesure

## Description

La table TypeMesure permet de définir les différents types de mesures RH appliquées aux employés, telles que l'embauche, la mutation, la suspension, ou la sortie. Elle sert de référence dans la gestion des événements liés au parcours administratif du salarié.

---

## Paramètres (Champs)

- id (PK) : Identifiant unique du type de mesure.
- code : Code court permettant d'identifier le type de mesure (ex : EMB pour Embauche, SUS pour Suspension).
- nom : Libellé complet du type de mesure (ex : Mutation interne, Embauche en CDI).
- embauche : Booléen (case à cocher) indiquant si ce type de mesure correspond à une embauche (true / false).

## Méthodes (Opérations)

- createTypeMesure(data)
  Crée un nouveau type de mesure RH dans le système.
- updateTypeMesure(id, data)
  Met à jour les informations d'un type de mesure existant.
- deleteTypeMesure(id)
  Supprime un type de mesure à partir de son identifiant.
- getTypeMesureById(id)
  Récupère les détails d'un type de mesure spécifique.
- listTypesMesure()
  Liste tous les types de mesures RH enregistrés.

## Relations

- **TypeMesure → Mesure (1 : N)**
  *Un type de mesure peut être associé à plusieurs enregistrements dans la table Mesure.*
  *Chaque mesure appliquée à un salarié est rattachée à un seul type défini dans TypeMesure.*

-

# Table 29 : MotifMesure

## Description

La table *MotifMesure* fournit des détails sur les motifs associés à un événement enregistré dans la table *Mesure*. Par exemple, pour une promotion, le motif peut indiquer s'il s'agit d'une augmentation générale ou individuelle ; pour une sortie, s'il s'agit d'une démission ou d'une fin de contrat (CDD).

## Paramètres (Champs)

- **id_motif (PK) :** Identifiant unique du motif.
- **id_mesure (FK → Mesure) :** Référence à l'événement auquel le motif est associé.
- **motif :** Libellé du motif (par exemple, "Augmentation individuelle", "Démission", "Fin de contrat").
- **id_typemotifmesure (FK → TypeMotifMesure ):** Référence au type de motif de la mesure
- **description :** Description complémentaire du motif (facultatif).
- **date_debut :** Date à partir de laquelle le motif s'applique.
- **date_fin** : Date jusqu'à laquelle le motif s'applique.

## Méthodes (Opérations)

- **createMotifMesure(data)**
- **updateMotifMesure(id, data)**
- **deleteMotifMesure(id)**
- **getMotifMesureById(id)**
- **listMotifMesureByMesure(id_mesure)**

## Relations

- **Mesure → MotifMesure :** 1 : N
- **TypeMotifMesure → MotifMesure (1 : N)**
  *Un type de motif peut être utilisé dans plusieurs enregistrements de la table* ***MotifMesure****.*
  *Chaque* ***MotifMesure*** *fait référence à un seul* ***TypeMotifMesure*** *pour indiquer la catégorie à laquelle il appartient.*

# Table 30 : TypeMotifMesure

## Description

- La table **TypeMotifMesure** définit les catégories générales de motifs applicables aux mesures RH. Elle sert de référentiel pour classifier les motifs enregistrés dans la table **MotifMesure** (par exemple : "Sortie", "Promotion", "Mutation", "Sanction", etc.). Cette catégorisation facilite le filtrage, l'analyse et le traitement administratif des différents types de décisions RH.

## Paramètres (Champs)

- **id_typemotifmesure (PK)** : Identifiant unique du type de motif.
- **code** : Code court ou abréviation du type de motif (ex : SORT, PROM, MUT, SANC).
- **libelle** : Libellé explicite du type de motif (ex : « Sortie », « Promotion », « Mutation », « Sanction disciplinaire »).
- **description** : Description complémentaire sur l'usage ou le contexte du type de motif (facultatif).

## Méthodes (Opérations)

- **createTypeMotifMesure(data)**
  Crée un nouveau type de motif dans le référentiel.
- **updateTypeMotifMesure(id,data)**
  Met à jour les informations d'un type de motif existant.
- **deleteTypeMotifMesure(id)**
  Supprime un type de motif à partir de son identifiant.
- **getTypeMotifMesureById(id)**
  Récupère les détails d'un type de motif spécifique.
- **listTypesMotifMesure()**
  Liste tous les types de motifs enregistrés.

## Relations

- **TypeMotifMesure → MotifMesure (1 : N)**
  *Un type de motif peut être utilisé dans plusieurs enregistrements de la table **MotifMesure**.*
  *Chaque **MotifMesure** fait référence à un seul **TypeMotifMesure** pour indiquer la catégorie à laquelle il appartient.*
-

# Table 31 : ProfilSalarial

## Description

La table *ProfilSalarial* contient les profils types des employés, définissant les caractéristiques standards de chaque catégorie (ex : Cadre, Technicien, Ouvrier). Ces profils servent à attribuer une grille salariale et à orienter les paramètres de rémunération.

## Paramètres (Champs)

- **id_profil (PK)** : Identifiant unique du profil salarial.
- **nom_profil** : Nom du profil (ex : Cadre commercial, Technicien).
- **Id_categorieSalariale ( FK → CategorieSalariale) :** Référence à la catégorie salariale.
- **Id_statuSalarial ( FK → StatutSalarial ) :** Référence au statut salarial.
- **fonction** : Description ou intitulé du poste type associé.
- **salaire_base** : Salaire de base standard pour le profil.
- **Id_prime ( FK → PrimeIndemniteRetenue )** : Référence à la prime associée
- **date_debut** : Date d'entrée en vigueur du profil.
- **date_fin** : Date de fin de validité du profil (si applicable).

## Méthodes (Opérations)

- **createProfilSalarial(data)**
  Crée un nouveau profil salarial.
- **updateProfilSalarial(id, data)**
  Met à jour un profil existant.
- **deleteProfilSalarial(id)**
  Supprime un profil par son identifiant.
- **getProfilSalarialById(id)**
  Récupère les informations d'un profil spécifique.
- **listProfilSalarial()**
  Liste tous les profils salariaux enregistrés.

## Relations

- **ProfilSalarial → PrimeIndemniteRetenue (1 : N)**

*Un profil salarial peut être associé à plusieurs primes, indemnités ou retenues prédéfinies.*
*Chaque enregistrement dans PrimeIndemniteRetenue peut être paramétré à l'avance pour un seul ProfilSalarial, facilitant l'automatisation de la rémunération selon le profil de l'employé.*

- **ProfilSalarial → CategorieSalariale (N : 1)**

  *Chaque **ProfilSalarial** est rattaché à une seule catégorie salariale.*
  *Une **CategorieSalariale** peut regrouper plusieurs profils salariaux similaires en termes de niveau ou de classification.*

- **ProfilSalarial → StatutSalarial (N : 1)**

  *Chaque **ProfilSalarial** correspond à un seul statut salarial (ex. : Cadre, Non Cadre, Stagiaire).*
  *Un **StatutSalarial** peut être partagé par plusieurs profils salariaux ayant des conditions de statut similaires.*

# Table 32 : GrilleSalariale

## Description

La table *GrilleSalariale* définit les grilles de salaires associées à un profil salarial. Elle détaille le niveau, l'échelon, l'ancienneté minimale et le salaire minimum applicable, permettant de structurer les évolutions de rémunération en fonction des parcours professionnels.

## Paramètres (Champs)

- **id_grille (PK)** : Identifiant unique de la grille salariale.
- **Id_echelon  (FK → Echelon ) :** Référence à l'échelon associé**.**
- **anciennete_min** : Ancienneté minimale requise pour atteindre ce niveau.
- **salaire_min** : Salaire minimum défini pour ce niveau/échelon.
- **date_debut** : Date d'entrée en vigueur de la grille.
- **date_fin** : Date de fin de validité (si applicable).

## Méthodes (Opérations)

- **createGrilleSalariale(data)**
  Crée une nouvelle grille salariale pour un profil donné.
- **updateGrilleSalariale(id, data)**
  Met à jour les informations d'une grille existante.
- **deleteGrilleSalariale(id)**
  Supprime une grille salariale par son identifiant.
- **getGrilleSalarialeByProfil(profilId)**
  Récupère la grille associée à un profil spécifique.
- **listGrilleSalarialeByProfil(profilId)**
  Liste toutes les grilles salariales associées à un profil donné.

## Relations

- **Echelon → GrilleSalariale (N : 1)**
  *Chaque échelon est défini dans le cadre d'une grille salariale.*
  *Une grille salariale peut contenir plusieurs échelons associés à différents niveaux.*

# Table 33 : Echelon

## Description

La table Echelon permet de définir les niveaux hiérarchiques ou de progression d'un salarié au sein d'une catégorie salariale, d'un statut salarial, et d'une grille salariale donnée. Elle sert à affiner la gestion des rémunérations, avancements ou paliers selon la structure de l'entreprise.

## Paramètres (Champs)

- **id (PK) :** Identifiant unique de l'échelon**.**
- **niveau :** Niveau hiérarchique global (ex : 1, 2, 3...), utilisé pour l'ordre ou la classification**.**
- **echelon :** Libellé ou code de l'échelon (ex : A1, B2...).
- **date_debut :** Date d'entrée en vigueur de l'échelon.
- **date_fin :** Date de fin de validité de l'échelon.
- **id_cat (FK) :** Référence à la CategorieSalariale à laquelle appartient l'échelon.
- **id_statut (FK) :** Référence au StatutSalarial associé à cet échelon.
- **id_grillesalariale (FK) :** Référence à la GrilleSalariale définissant les barèmes pour cet échelon.

## Méthodes (Opérations)

- **createEchelon(data)**
  Crée un nouvel échelon dans le système.
- **updateEchelon(id, data)**
  Met à jour les informations d'un échelon existant.
- **deleteEchelon(id)**
  Supprime un échelon par son identifiant.
- **getEchelonById(id)**
  Récupère les détails d'un échelon spécifique.
- **listEchelons()**
  Liste tous les échelons définis dans le système.

## Relations

- **Echelon → CategorieSalariale (N : 1)**
  *Un échelon appartient à une seule catégorie salariale.*
  *Une catégorie salariale peut contenir plusieurs échelons.*
- **Echelon → StatutSalarial (N : 1)**
  *Chaque échelon est associé à un seul statut salarial.*
  *Un statut salarial peut regrouper plusieurs échelons.*
- **Echelon → GrilleSalariale (N : 1)**
  *Chaque échelon est défini dans le cadre d'une grille salariale.*
  *Une grille salariale peut contenir plusieurs échelons associés à différents niveaux.*

# Table 34 : BaremeIR

**Description**

- La table *BaremeIR* sert à paramétrer le barème de l'impôt sur le revenu (IR) via l'interface du configurateur. Elle permet de gérer les différentes tranches d'imposition, en définissant les seuils (valeur minimale et maximale), le taux d'imposition applicable et le montant à déduire pour chaque tranche. Cette table est autonome et sera utilisée par le module de calcul **d'impôt**.

**Paramètres (Champs)**

- **id_tranche (PK)** : Identifiant unique de la tranche.
- **Minimum** : Valeur minimale (seuil inférieur) de la tranche.
- **maximum**: Valeur maximale (seuil supérieur) de la tranche.
- **taux_ir** : Taux d'imposition applicable pour la tranche (exprimé en pourcentage ou en décimal).
- **montant_deduction** : Somme à déduire pour cette tranche.
- **date_debut** : Date d'entrée en vigueur du barème.
- **date_fin** : Date de fin de validité.

**Méthodes (Opérations)**

- **createBaremeIR(data)**
  Crée une nouvelle tranche dans le barème de l'IR avec les informations fournies.
- **updateBaremeIR(id, data)**
  Met à jour les informations d'une tranche existante identifiée par son ID.
- **deleteBaremeIR(id)**
  Supprime une tranche du barème de l'IR en fonction de son identifiant.
- **getBaremeIRById(id)**
  Récupère les détails d'une tranche spécifique.
- **listBaremeIR()**
  Liste toutes les tranches définies dans le barème de l'IR.

**Relations**

- **BaremeIR → Configurateur (N : 1)**
  *Chaque tranche ou ligne du barème IR est définie ou ajustée par un seul paramétreur.*

*Un Configurateur peut paramétrer plusieurs lignes du Barème IR, ce qui permet de conserver un historique clair des modifications fiscales appliquées.*

# Table 35: Constantes

**Description**

- La table *Constantes* centralise les valeurs fixes utilisées dans le programme. Elle permet de définir des constantes avec une période de validité (date début et date fin), un identifiant unique, un code, un nom descriptif et une valeur associée. Cette table est utile pour gérer des paramètres globaux que le programme peut consulter ou modifier selon les besoins.

**Paramètres (Champs)**

- **id_const (PK)** : Identifiant unique de la constante.
- **code_const** : Code unique associé à la constante.
- **nom_const** : Nom ou libellé descriptif de la constante.
- **valeur** : Valeur de la constante (peut être numérique, textuelle, etc. en fonction de son usage).
- **date_debut** : Date de début d'application ou de validité de la constante.
- **date_fin** : Date de fin d'application ou de validité de la constante.

**Méthodes (Opérations)**

- **createConstante(data)**
  Crée une nouvelle constante avec les informations fournies.
- **updateConstante(id, data)**
  Met à jour les informations d'une constante existante identifiée par son **id_const**.
- **deleteConstante(id)**
  Supprime une constante en fonction de son **id_const**.
- **getConstanteById(id)**
  Récupère les détails d'une constante spécifique.
- **listConstantes()**
  Liste toutes les constantes enregistrées dans le système.

**Relations**

**• Constantes → Configurateur (N : 1)**

*Chaque constante du système est enregistrée ou modifiée par un seul paramétreur (configurateur).*
*Un **Configurateur** peut être à l'origine de plusieurs modifications ou ajouts de*

***constantes**, permettant ainsi de tracer l'historique des ajustements techniques du système.*

# Table 36: CategorieSalariale

**Description**

La table *CategorieSalariale* permet de paramétrer les catégories de salariés. Elle définit les codes, noms et descriptions associées à chaque catégorie (ex : Cadre, Employé, Ouvrier) ainsi que la période pendant laquelle la catégorie est valable.

**Paramètres (Champs)**

- id_categorie (PK) : Identifiant unique de la catégorie.
- code_categorie : Code de la catégorie (Champ de texte).
- nom_categorie : Nom de la catégorie (Champ de texte, visible).
- description_categorie : Description facultative de la catégorie (Zone de texte libre, non visible).
-
- date_debut : Date de début de validité (Date, visible).
- date_fin : Date de fin de validité (Date, visible).

**Méthodes (Opérations)**

- createCategorieSalariale(data)
  Crée une nouvelle catégorie de salarié.
- updateCategorieSalariale(id, data)
  Met à jour une catégorie existante.
- deleteCategorieSalariale(id)
  Supprime une catégorie par son identifiant.
- getCategorieSalarialeById(id)
  Récupère les détails d'une catégorie spécifique.
- listCategorieSalariale()
  Liste toutes les catégories de salariés.

**Relations**

- **Echelon → CategorieSalariale (N : 1)**
  *Un échelon appartient à une seule catégorie salariale.*
  *Une catégorie salariale peut contenir plusieurs échelons.*

- **ProfilSalarial → CategorieSalariale (N : 1)**

  *Chaque **ProfilSalarial** est rattaché à une seule catégorie salariale.*
  *Une **CategorieSalariale** peut regrouper plusieurs profils salariaux similaires en termes de niveau ou de classification.*

- **CategorieSalariale → StatutSalarial (N : 1)**

*Chaque catégorie salariale est rattachée à un seul statut salarial (ex. : Cadre, Non Cadre).*
*Un StatutSalarial peut regrouper plusieurs CategorieSalariale partageant les mêmes conditions de statut (avantages, obligations, réglementation RH).*

# Table 37: StatutSalarial

### Description

La table *StatutSalarial* permet de paramétrer les statuts des salariés (ex : Actif, Suspendu, En congé). Elle inclut un code, un nom, une description optionnelle, une raison d'inactivité (le cas échéant) et une période de validité pour chaque statut.

### Paramètres (Champs)

- id_statut (PK) : Identifiant unique du statut.
- code_statut : Code du statut (Champ de texte).
- Id_categorie (FK → CategorieSalariale) : Référence à la catégorie salariale associé.
- nom_statut : Nom du statut (Champ de texte, visible).
- description_statut : Description optionnelle du statut (Zone de texte libre, non visible).
- raison_inactivite : Raison d'inactivité, si applicable (Champ de texte, non visible).
- date_debut : Date de début de validité du statut (Date, visible).
- date_fin : Date de fin de validité du statut (Date, visible).

### Méthodes (Opérations)

- createStatutSalarial(data)
  Crée un nouveau statut pour un salarié.
- updateStatutSalarial(id, data)
  Met à jour un statut existant.
- deleteStatutSalarial(id)
  Supprime un statut par son identifiant.
- getStatutSalarialById(id)
  Récupère les détails d'un statut spécifique.
- listStatutSalarial()
  Liste tous les statuts définis.

### Relations

- **CategorieSalariale → StatutSalarial (N : 1)**

  *Chaque catégorie salariale est rattachée à un seul statut salarial (ex. : Cadre, Non Cadre).*
  *Un StatutSalarial peut regrouper plusieurs CategorieSalariale partageant les mêmes conditions de statut (avantages, obligations, réglementation RH).*

- **Echelon → StatutSalarial (N : 1)**
  *Chaque échelon est associé à un seul statut salarial.*
  *Un statut salarial peut regrouper plusieurs échelons.*

- **ProfilSalarial → StatutSalarial (N : 1)**

  *Chaque **ProfilSalarial** correspond à un seul statut salarial (ex. : Cadre, Non Cadre, Stagiaire).*
  *Un **StatutSalarial** peut être partagé par plusieurs profils salariaux ayant des conditions de statut similaires.*

# Table 38 : UniteOrganisationnelle

**Description**

La table *UniteOrganisationnelle* permet de paramétrer les unités organisationnelles de l'entreprise. Elle définit le code, le nom, le type d'unité (ex : Département, Service, Direction), ainsi que le rattachement hiérarchique (possibilité d'auto-référence), la description et le statut de l'unité, avec une période de validité.

**Paramètres (Champs)**

- **id_unite (PK)** : Identifiant unique de l'unité organisationnelle.
- **code_unite** : Code de l'unité (Champ de texte).
- **nom_unite** : Nom de l'unité (Champ de texte, visible).
- **type_unite** : Type d'unité, par exemple "Département", "Service", "Direction" (Liste déroulante, visible).
- **rattachement** : Clé étrangère auto-référencée (FK vers id_unite) pour déterminer l'unité parent, si applicable (Liste déroulante, visible).
- **description_unite** : Description de l'unité (Zone de texte libre, non visible).
- **date_debut** : Date de début de validité de l'unité (Date, visible).
- **date_fin** : Date de fin de validité de l'unité (Date, visible).

**Méthodes (Opérations)**

- **createUniteOrganisationnelle(data)**
  Crée une nouvelle unité organisationnelle.
- **updateUniteOrganisationnelle(id, data)**
  Met à jour les informations d'une unité existante.
- **deleteUniteOrganisationnelle(id)**
  Supprime une unité par son identifiant.
- **getUniteOrganisationnelleById(id)**
  Récupère les détails d'une unité spécifique.
- **listUniteOrganisationnelle()**
  Liste toutes les unités organisationnelles.

**Relations**

- La relation de rattachement est auto-référencée :
  **UniteOrganisationnelle → UniteOrganisationnelle (rattachement) : 1 : N**
  *(Une unité peut être le parent d'une ou plusieurs autres unités.)*

- **UniteOrganisationnelle → Societe (N : 1)**
  *Chaque unité organisationnelle appartient à une seule société.*
  *Une Societe peut comporter plusieurs UnitesOrganisationnelles, correspondant à ses départements, agences ou divisions internes.*

- **EmployeSimple → UniteOrganisationnelle (N : 1)**
  *Chaque employé est rattaché à une seule unité organisationnelle.*
  *Une UniteOrganisationnelle peut regrouper plusieurs EmployeSimple, selon leur affectation (département, service ou structure hiérarchique).*