TP étude de cas
Gestion Achat

UP ASI
Bureau E204

TP Gestion Magasin-Stock

- On désire créer une application de gestion des achats d'un magasin.
- Le magasin contient des produits ayant plusieurs catégories.
- Le stock de produit est géré de façon à assurer le suivi des mouvements des produits et détecter rapidement les stocks à alimenter.
- Des fournisseurs avec des catégories bien précises et un ou plusieurs domaines d'activités sont gérés au niveau de l'application.
- un ensemble de produits sous forme d'une facture affichant le montant total à payer est commandé par le magasin à des fournisseurs. Les détails de la facture sont également présentés(prix de chaque produit, quantité commandé, montant remise).

TP Gestion Magasin-Stock

- La facture est saisie par un opérateur.
- Un règlement partiel ou intégral peut être affecté à une facture fournisseur.
- L'application nous permet également d'afficher des métriques intéressantes tel que le chiffre d'affaire généré entre deux dates et le pourcentage de recouvrement.

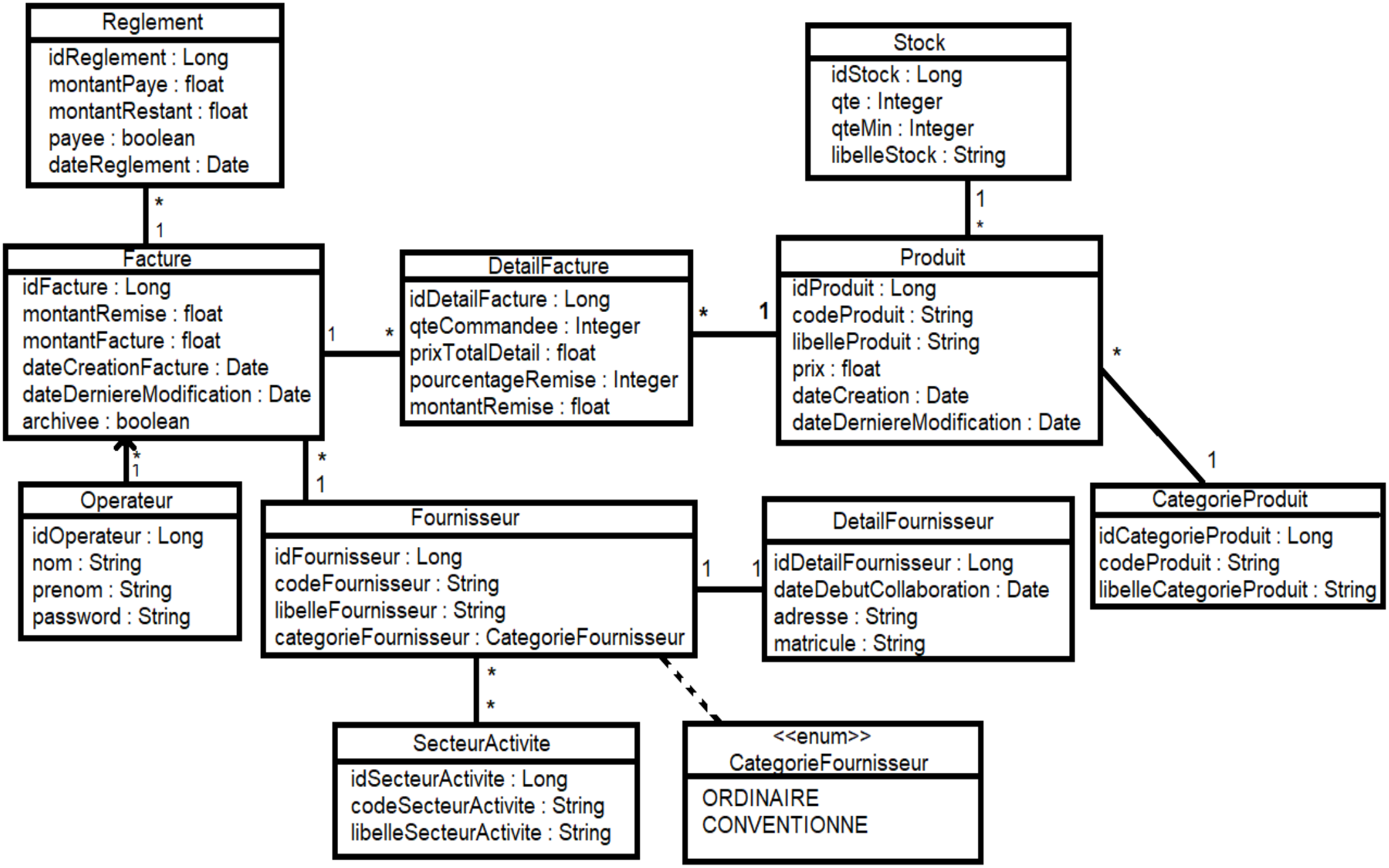

Figure 1 : Diagramme de classes gestion Achat

NB : la relation Facture-Operateur est une relation unidirectionnelle

Travail à faire

Partie 1 Spring Data JPA – Première entité

- Créer les entités se trouvant dans le diagramme des classes (sans les associations) et vérifier qu'ils ont été ajoutés avec succès dans la base de données.

Travail à faire

Partie 2 Spring Data JPA – Le mapping des différentes associations

- Supprimer les tables existantes dans la base de données.
- Créer les associations entre les différentes entités.
- Générer la base de donné de nouveau et vérifier que le nombre de tables crées est correct.

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

Entité Stock

List<Stock> retrieveAllStocks();

Stock addStock(Stock s);

Stock updateStock(Stock s);

Stock retrieveStock(Long id);

void removeStock(Long id);

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

Entité CategorieProduit

List<CategorieProduit> retrieveAllCategorieProduits();

CategorieProduit addCategorieProduit(CategorieProduit cp);

CategorieProduit updateCategorieProduit(CategorieProduit cp);

CategorieProduit retrieveCategorieProduit(Long id);

void removeCategorieProduit(Long id);

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

Entité Produit

List<Produit> retrieveAllProduits();

Produit addProduit (Produit p, Long idCategorieProduit, Long idStock);

Produit updateProduit (Produit p, Long idCategorieProduit, Long idStock);

Produit retrieveProduit(Long id);

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

Entité Operateur

List<Operateur> retrieveAllOperateurs();

Operateur addOperateur (Operateur o);

Operateur updateOperateur (Operateur o);

Operateur retrieveOperateur (Long id);

void removeOperateur (Long id);

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

Entité SecteurActivite

List<SecteurActivite> retrieveAllSecteurActivites();

SecteurActivite addSecteurActivite (SecteurActivite sa);

SecteurActivite updateSecteurActivite (SecteurActivite sa);

SecteurActivite retrieveSecteurActivite (Long id);

void removeSecteurActivite (Long id);

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Créer les CRUD des différentes **entités indiqués dans les slides suivants en respectant les signatures suivantes**

Entité Fournisseur

List<Fournisseur> retrieveAllFournisseurs();
Fournisseur addFournisseur (Fournisseur f);
Fournisseur updateFournisseur (Fournisseur f);
Fournisseur retrieveFournisseur (Long id);

NB: Pour l'ajout de fournisseur, il faut créer en même temps le détail fournisseur (entité fournisseur avec l'entité associé detailFournisseur)

Travail à faire

Partie 3 Spring Data JPA CRUD Repository–Le langage JPQL - JPA Repository

Entité Facture

List<Facture> retrieveAllFactures();
void cancelFacture(Long id);
Facture retrieveFacture(Long id);

NB : pour l'annulation de la facture, il faut que le champs archivee de la table facture soit mis à true.

Travail à faire

Partie 4 Spring MVC

Exposer les services implémentés dans la partie 3 avec Postman et/ou Swagger pour les tester.

Travail à faire

Partie 5 : Services avancés

On désire changer le stock d'un produit.
Créer un service permettant l'assignation d'un produit à un stock et exposer le en respectant la signature suivante :

void assignProduitToStock(Long idProduit, Long idStock);

Travail à faire

Partie 5 : Services avancés

On désire affecter un secteur d'activité à un fournisseur.
Créer un service permettant l'assignation d'un secteur d'activité à un fournisseur et exposer le en respectant la signature suivante :

public void assignSecteurActiviteToFournisseur(Long fournisseurId, Long secteurActiviteId) ;

Travail à faire

Partie 5 : Services avancés

On désire affecter un opérateur à une facture.
Créer un service permettant l'assignation d'un opérateur à une facture et exposer le en respectant la signature suivante :

public void assignOperateurToFacture(Long idOperateur, Long idFacture)

Travail à faire

Partie 5 : Services avancés

On souhaite récupérer les factures d'un fournisseur donné.
Créer le service adéquat en respectant la signature suivante :

List<Facture> getFacturesByFournisseur(Long idFournisseur);

Travail à faire

Partie 5 : Services avancés

On souhaite récupérer le ou les réglements d'une facture donnée.
Créer le service adéquat en respectant la signature suivante :

public List<Reglement> retrieveReglementByFacture(Long idFacture)

Travail à faire

Partie 5 : Services avancés

Nous venons d'acheter deux produits distincts d'un fournisseur.
Une facture doit etre saisie avec toutes les informations nécessaires.

Créer un service permettant de créer la facture avec les détails associés assignée au fournisseur concerné en respectant la signature suivante :

Facture addFacture(Facture f, Long idFournisseur);

PS : le calcul des montant factures et remise de l'entité **Facture** ainsi que le montant remise et le prix total de l'entité **detailFacture** doit etre fait convenablement.

Travail à faire

Partie 5 : Services avancés

Nous souhaitons effectuer un réglement partiel ou intégral d'une facture.

Créer un service permettant d'ajouter le réglement en respectant la signature suivante :

Reglement addReglement(Reglement r, Long idFacture)

PS : Le réglement d'une facture peut s'effectuer sur plusieurs fois.
Avant de sauvegarder le réglement dans la base de donnée, il faut récupèrer le montant des réglements déja effectué sur la facture et s'assurer que le montant à sauvegarder est inférieur au montantRestant.
Exemple
Facture : id=1 montantFacture = 1000
Reglement 1 sur la facture 1 = 350
Reglement 2 sur la facture 1 = 450
Reglement 3 sur la facture 1 <= 200 réglement acceptée >200 réglement refusé

Travail à faire

Partie 5 : Services avancés

Nous souhaitons calculer le chiffre d'affaire généré entre deux dates.
Créer un service permettant de faire le calcul en respectant la signature suivante :

public float getChiffreAffaireEntreDeuxDate(Date startDate, Date endDate)

PS : le chiffre d'affaire correspond à la somme des réglements entre deux dates correspondant à des factures non archivées.

Travail à faire

Partie 5 : Services avancés

Nous souhaitons calculer le pourcentage de recouvrement entre deux dates.
Créer un service permettant de faire le calcul en respectant la signature suivante :

float pourcentageRecouvrement(Date startDate, Date endDate);

PS : le pourcentage de recouvrement correspond à la somme des réglements divisés par la somme des factures non archivées entre deux dates.

Travail à faire

Partie 6 : Spring Scheduler

Nous souhaitons créer un service schedulé (programmé automatiquement) permettant d'avertir le responsable magasin des stocks dont la quantité disponible est inférieure à la quantité min toléré.

Créer un service nous permettant d'afficher les stocks concernés tous les jours à 22h en respectant la signature suivante :

String retrieveStatusStock();

NB: Pour des raisons de test, vous pouvez modifier l'horaire selon l'heure affiché sur votre machine. Le message sera affiché simplement sur console.

TP Gestion Magasin-Stock

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP Architectures des Systèmes d'Information

Bureau E204