Saint Louis, le 25 mai 2023

Dr Samba SIDIBE
Enseignant-chercheur
Ecole polytechnique de Thiès
ssidibe@ept.sn

Projet matière

Mise en place d'une plateforme de vente de vélos

Université Gaston Berger de Saint Louis

Institut Polytechnique de Thiès

ING 2

2022-2023

A rendre avant le 12 juin 2022

Resumé :

Ce projet est une continuité du projet précédent portant sur la conception et la réalisation du mapping objet-relationnel avec Jakarta Persistence API pour une application de vente de vélos. L'objectif principal est désormais de créer une plateforme de vente de vélos de bout en bout. Cette plateforme sera composée de plusieurs éléments :

— **Un service web :** développé à l'aide de JAX-RS, ce service doit respecter les normes du W3C en matière de services web RESTful. Le service web devra être documenté en utilisant OpenAPI (Swagger).

— **Une interface web JSF :** Pour la version web, l'interface sera créée en utilisant Jakarta Faces avec l'implémentation de base PrimeFaces.

La figure 1 illustre quelques étapes du développement de la plateforme et les dépendances entre les étapes.

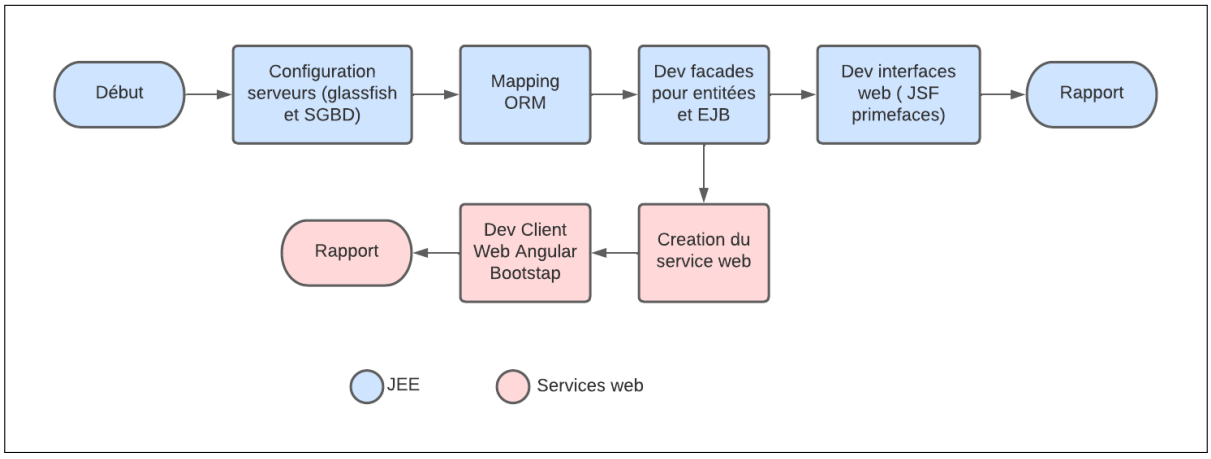

FIGURE 1 – Dépendances entre les taches globales

Livrables : Vous devez livrer un rapport écrit, le code source des applications, ainsi qu'une démonstration sous forme de captures vidéo.

Mots clés — Jakarta EE, EJB, JSF, services web, JAX-RS, Angular

Table des matières

Première partie

Jakarta EE *(100 pts)*

Dans le projet de mapping JPA donné précédemment, on avait demandé de proposer une solution JPA from scratch pour l'application de vente de vélo. On va retenir la solution proposée au niveau du diagramme des classes de la figure 2

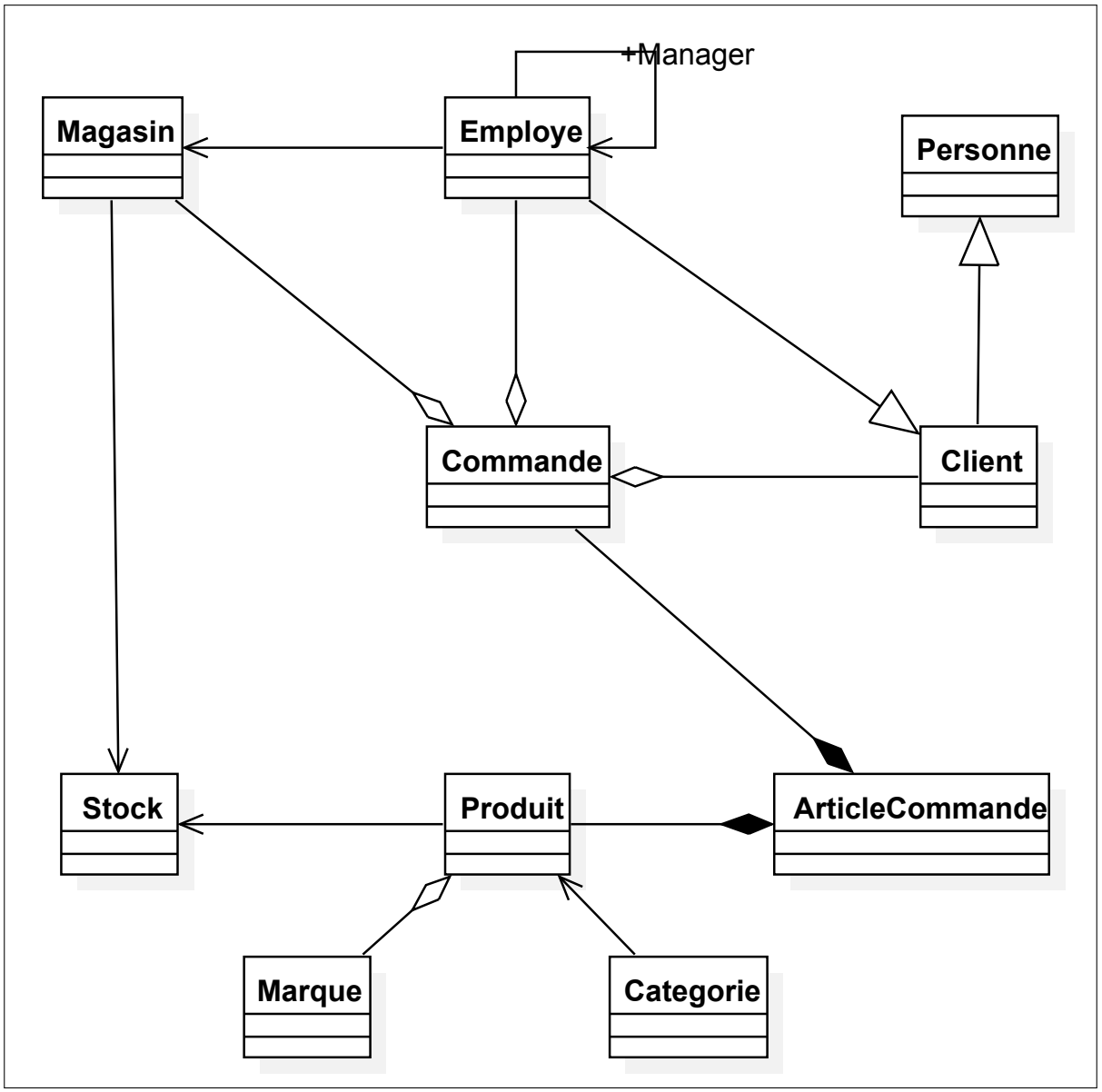

FIGURE 2 – Proposition de diagramme des classes pour les entités JPA

1 Configuration des serveurs *(5 pts)*

1.1 Installation serveur d'application *(1 pt)*

Installez un serveur d'application Jakarta EE de votre choix qui supporte la spécification Jakarta® Enterprise Beans (EJB) [1].

1.2 Installation SGBD *(1 pt)*

Installez un système de gestion de base de données relationnelles (SGBD) **différent** de MySQL.
NB : MySQL a été exclu pour vous permettre d'explorer la configuration de Jakarta Persistence (JPA) [2] avec d'autres SGBD.

1.3 Création source de données *(3 pts)*

Créez une source de données et la ressource associée pour permettre aux applications déployées d'accéder à la base de données à partir du serveur d'application JEE.

2 Mapping JPA *(5 point)*

2.1 Création nouveau projet *(1 pt)*

Créez un nouveau projet web et configurez-le pour qu'il utilise la source de données précédemment créée dans 1.3

2.2 Création des entités *(4 pts)*

Créez les entités correspondantes au diagramme de classes se trouvant sur la figure 2 et utilisez la stratégie **JOINED** pour l'héritage.

3 Développement des EJB et des façades *(15 points)*

3.1 Création des Facades *(1 pt)*

Créez les façades nécessaires pour l'enregistrement des entités.

3.2 Initialisation de la BD *(5 pts)*

Trouvez un moyen d'initialiser la base de données avec les données de test qui ont été fournies lors du TP précédent.

3.3 Session Bean pour l'envoi d'email *(4 pt)*

Créez une EJB Session Bean permettant d'envoyer un e-mail. Les paramètres d'accès au serveur SMTP seront donnés en pièce jointe.

3.4 Envoi d'alertes automatiques *(5 point)*

— Créez un EJB qui envoie un e-mail à chaque démarrage et arrêt de l'application.

— Créez un EJB qui envoie périodiquement par e-mail l'état des stocks des produits. Vous pouvez utiliser les Timers avec les annotations *@Schedules* et/ou *@Schedule* pour le faire (voir références [3, 4]).

4 Développement des interfaces web JSF *(70 points)*

4.1 CRUD sur les entités *(35 points)*

Créez les pages web nécessaires pour lister, ajouter, modifier et supprimer les entités. Si une entité n'a pas beaucoup de champs, vous pouvez utiliser le composant PrimeFaces *DataView* [5] pour les lister. Dans le cas contraire, utilisez les composants DataTable [6]

4.2 Creer un menu *(2 point)*

Choisissez l'un des composants de menu de *Primefaces*, puis gérez la navigation entre les pages de l'application.

4.3 Filtrage, triage et pagination *(18 points)*

Créez une page pour afficher la liste des produits. Utilisez le composant DataTable en mode *lazy*, avec une pagination et la possibilité de filtrer et de trier les différents produits. N'oubliez pas d'afficher les colonnes pour marque et catégorie.

4.4 Recherche avancée *(15 points)*

Créez une page permettant de faire des recherches avancées. Cette page doit utiliser le composant *DataView* pour afficher les résultats de manière *lazy-loading*.

5 Déploiement dans un environnement de production *(5 points)*

Dans cette section, vous devrez installer et déployer l'application dans un environnement Linux accessible via le cloud. Vous disposerez de 24 heures d'accès au serveur pour effectuer cette tâche. Il est recommandé de réaliser cette partie une fois que l'ensemble du projet est terminé.

5.1 Installation de l'environnement de production *(4 points)*

Installez les outils nécessaires au déploiement dans le serveur Linux.

5.2 Déployez l'application à la racine du serveur *(1 point)*

Deuxième partie

Service web restful *(100 pts)*

Dans cette partie, vous devez créer un service web *RESTful* qui supporte au moins la communication avec des messages au format XML et JSON. Ce service web est destiné aux clients mobiles tels qu'Android et aux clients web tels qu'Angular.

6 Définir les ressources *(20 pts)*

6.1 Identification des ressources *(5 pts)*

Identifiez les différentes ressources nécessaires au fonctionnement des applications mobiles et web.

6.2 Concevoir les endpoints *(15 pts)*

Pour chaque ressource, veuillez définir les différentes actions possibles ainsi que leurs URI et méthodes HTTP associées. Les choix doivent respecter les normes et règles définissant les services web RESTful et éventuellement le protocole HTTP.

7 Conception des services web *(40 pts)*

7.1 Développement des services web *(25 pts)*

Développez les services web correspondants aux solutions proposées à la question 6.2

7.2 Documentation des services web *(15 pts)*

Documentez les services web avec OpenAPI (Swagger). La documentation doit être détaillée afin de faciliter l'utilisation du service web par n'importe quel développeur.

8 Test des services web *(25 pts)*

Une fois la documentation terminée, vous pouvez normalement tester tous les services à l'aide de Swagger UI. L'ajout de nouveaux services ou la modification des entités peut entraîner des régressions sur les endpoints (fonctionnalités qui ne fonctionnent plus correctement). Il serait trop lourd de demander aux développeurs de tester manuellement tous les endpoints à chaque modification du code, d'où l'importance d'automatiser les tests.

Pour automatiser les tests unitaires, on peut les inclure dans le même projet que celui des services web. Ce type de test est lourd et lent, car il faut démarrer et arrêter le serveur pour chaque test unitaire. En général, on utilise la version embarquée du serveur "Embedded" qui dispose d'une API permettant de démarrer ou d'arrêter le serveur à l'aide de code Java.

Une autre option consiste à créer un nouveau programme et d'y inclure les tests. Il s'agira donc de créer des clients (dans les tests unitaires) qui enverront des requêtes aux services web. Pour valider un test unitaire, le client vérifiera si la réponse attendue correspond à ce que le serveur retourne.

Nous allons opter pour la solution précédente. Il faudra donc :

— Créer un projet

— Ecrire un test unitaire pour chaque fonctionnalité

— Vérifier que tous les tests passent

Troisième partie

Ecrire un rapport *(15 pts)*

Ecrivez un rapport détaillé et déposez le code source sur *Google Drive* ou *Github*

Références

[1] J. EE, "Jakarta®enterprise beans specification." [Online]. Available : https ://jakarta.ee/specifications/enterprise-beans/4.0/jakarta-enterprise-beans-spec-core-4.0.html

[2] ——, "Jakarta®persistence specification." [Online]. Available : https ://jakarta.ee/specifications/persistence/3.0/jakarta-persistence-spec-3.0.html

[3] ——, "Schedule (jakarta ee platform api)." [Online]. Available : https ://jakarta.ee/specifications/platform/10/apidocs/jakarta/ejb/schedule

[4] P. Sluiter, "Java ee timer scheduled events." [Online]. Available : https ://www.youtube.com/watch ?v=LrXBZJSRX3A

[5] PrimeFaces, "dataview." [Online]. Available : https ://www.primefaces.org/showcase/ui/data/dataview/basic.xhtml

[6] ——, "datatable." [Online]. Available : https ://www.primefaces.org/showcase/ui/data/datatable/basic.xhtml