Projet Jakarta EE – Frameworks & Composants

1. Objectif du projet

En groupe de 4 à 5 étudiants, vous devrez concevoir et développer une application Jakarta EE 8 autour d'un domaine libre. Le projet doit démontrer votre capacité à construire une architecture complète intégrant plusieurs composants Jakarta EE, des design patterns qui ont été présentés, ainsi qu'un système obligatoire de gestion des utilisateurs avec production d'événements JMS.

À la fin du projet, vous devrez rendre :
- Le code source complet (repo GitHub)
- Un diagramme de composants
- Une documentation
- Une démonstration (vidéo ou présentation)

2. Exigences techniques Jakarta EE

- API REST complète (CRUD) via JAX-RS
- Persistance via JPA (minimum 3 entités + relations)
- Messagerie asynchrone via JMS (producteur + consommateur)

3. Module obligatoire : Gestion des utilisateurs + Événement JMS

Chaque projet doit obligatoirement intégrer un module de gestion des utilisateurs comprenant :
- Une entité User persistée via JPA
- Un CRUD REST minimal : POST /users, GET /users, GET /users/{id}
- Un message JMS publié dans une queue lors de la création d'un utilisateur

Le message JMS doit contenir au minimum : l'ID de l'utilisateur créé, un timestamp, et éventuellement des informations supplémentaires (email, nom…).

4. Système de consommation en Queue ou Topic

Chaque projet doit inclure au moins un consommateur JMS. Deux options sont possibles :
- Consommateur basé sur une Queue (point-à-point)
- Consommateur basé sur un Topic (publish-subscribe)

Le consommateur doit réagir au message « création utilisateur » et effectuer un traitement libre (notification interne, journalisation, mise à jour d'un autre module, etc.).

5. Design Patterns – Contraintes et liberté

Vous devez intégrer au minimum 3 design patterns parmi ceux présentés en cours. Le choix est libre.
Exemples : Content-based Router, Message Filter, Translator, Enricher, Aggregator, Routing Slip, Lazy Load, Embedded Value.
Un pattern supplémentaire pourra être implémenté pour obtenir un bonus.

6. Modélisation attendue

Vous devez fournir un diagramme de composants global représentant :
- Les modules principaux (REST, Business, JPA, Messaging)
- Le flux général des données entre les composants
- Les patterns choisis

7. Documentation attendue

Votre documentation doit inclure :
- Présentation du problème et du fonctionnement de l'application

- Architecture globale expliquée
- Patterns choisis + justification claire
- Un scénario d'utilisation complet illustrant le flux REST → EJB → JPA → JMS → consommateur
- Un résumé des contributions individuelles du groupe

8. Exemples de domaines possibles

- Plateforme de réservation
- Système de gestion d'événements
- Gestionnaire de stock ou logistique
- Suivi sportif ou nutritionnel
- Gestion de missions spatiales
- Gestionnaire de tournoi e-sport

10. Critères d'évaluation (20 pts)

- Gestion des utilisateurs: 2 pts
- Émission JMS lors de la création utilisateur : 2 pts
- Consommateur JMS (Queue ou Topic) : 2 pts
- Autres composants Jakarta EE : 5 pts
- Patterns utilisés et pertinence : 4 pts
- Qualité de la documentation : 2 pts
- Qualité du diagramme de composants : 1 pts
- Organisation du code + démonstration : 2 pts
- Bonus technique éventuel : +1 pts

Annexe – Structure recommandée pour la documentation

1. Présentation du fonctionnement global de l'application

2. Diagramme de composants

3. Description technique par couche (REST, EJB, JPA, JMS, CDI)

4. Description du module utilisateur

5. Patterns choisis + justification

6. Résumé des contributions du groupe

Annexe – Design Patterns à utiliser

1. Messaging Patterns – Messages

Pattern	Description courte
Command Message	Encapsule une action à réaliser dans un message.
Document Message	Contient un document complet structuré.
Event Message	Transmet un événement survenu dans le système.
Request-Reply	Communication synchrone basé sur requête / réponse.
Return Address	Indique où envoyer une réponse.
Correlation Identifier	Permet de relier requêtes et réponses.
Message Sequence	Découpe un message en une suite ordonnée.
Message Expiration	Définit une durée de validité des messages.

Format Indicator	Indique le format d’un message (XML/JSON…).

2. Routing Patterns – Routage & Distribution

Pattern	Description
Pipes-and-Filters	Pipeline où chaque filtre transforme le message.
Message Router	Route les messages selon une règle.
Content-based Router	Route selon le contenu du message.
Message Filter	Filtre des messages selon critères.
Dynamic Router	Routeur dont la stratégie peut changer dynamiquement.
Recipient List	Envoie à une liste d’acteurs dynamique.
Splitter	Divise un message en plusieurs fragments.
Aggregator	Regroupe plusieurs messages en un seul.
Resequencer	Réordonne des messages reçus dans le désordre.
Composed Message Processor	Combine Splitter + Filters + Aggregator.
Scatter-Gather	Diffuse à plusieurs consommateurs, puis regroupe les réponses.
Routing Slip	Le message contient sa propre route.

3. Transformation Patterns – Transformation & Enrichissement

Pattern	Description
Message Translator	Convertit un message d’un format vers un autre.
Content Enricher	Ajoute des données supplémentaires au message.
Content Filter	Retire les données inutiles d’un message.
Normalizer	Uniformise les messages entrants.
Canonical Data Model	Format de données standard pour le système.

4. Endpoint Patterns – Intégration & Communication

Pattern	Description
Message Endpoint	Point d’entrée/sortie des messages.
Messaging Gateway	Interface unifiée pour envoyer/recevoir des messages.
Messaging Mapper	Convertit objets métier ↔ messages.
Transactional Client	Gère transactions sur réception de messages.
Polling Consumer	Récupère les messages par interrogation.
Event-driven Consumer	Réagit automatiquement à l’arrivée d’un message.
Competing Consumers	Plusieurs consommateurs en concurrence.
Message Dispatcher	Répartit les messages entre handlers.
Selective Consumer	Ne consomme que certains messages.

5. Channel Patterns – Canaux

Pattern	Description
Message Channel	Canal de communication.
Point-to-Point Channel	Un seul consommateur par message.
Publish-Subscribe Channel	Diffusion à plusieurs consommateurs.
Datatype Channel	Canal spécialisé par type de message.
Invalid Message Channel	Canal des messages incorrects.
Dead Letter Channel	Messages non livrables.
Guaranteed Delivery	Livraisons garanties via persistance.
Channel Adapter	Connecte système externe ↔ canal.
Messaging Bridge	Relie deux systèmes de messaging.
Message Bus	Infrastructure commune de messaging.

6. Patterns liés à la persistence (JPA)

Pattern	Description
Unit of Work	Coordonne les opérations d’écriture atomiques.
Lazy Load	Charge différé des relations.
Embedded Value	Valeur embarquée dans une entité.
Single Table Inheritance	Une table unique pour toute la hiérarchie.
Class Table Inheritance	Une table par classe abstraite.
Concrete Table Inheritance	Une table par classe concrète.
Gateway Pattern	API dédiée pour accéder à un sous-système.