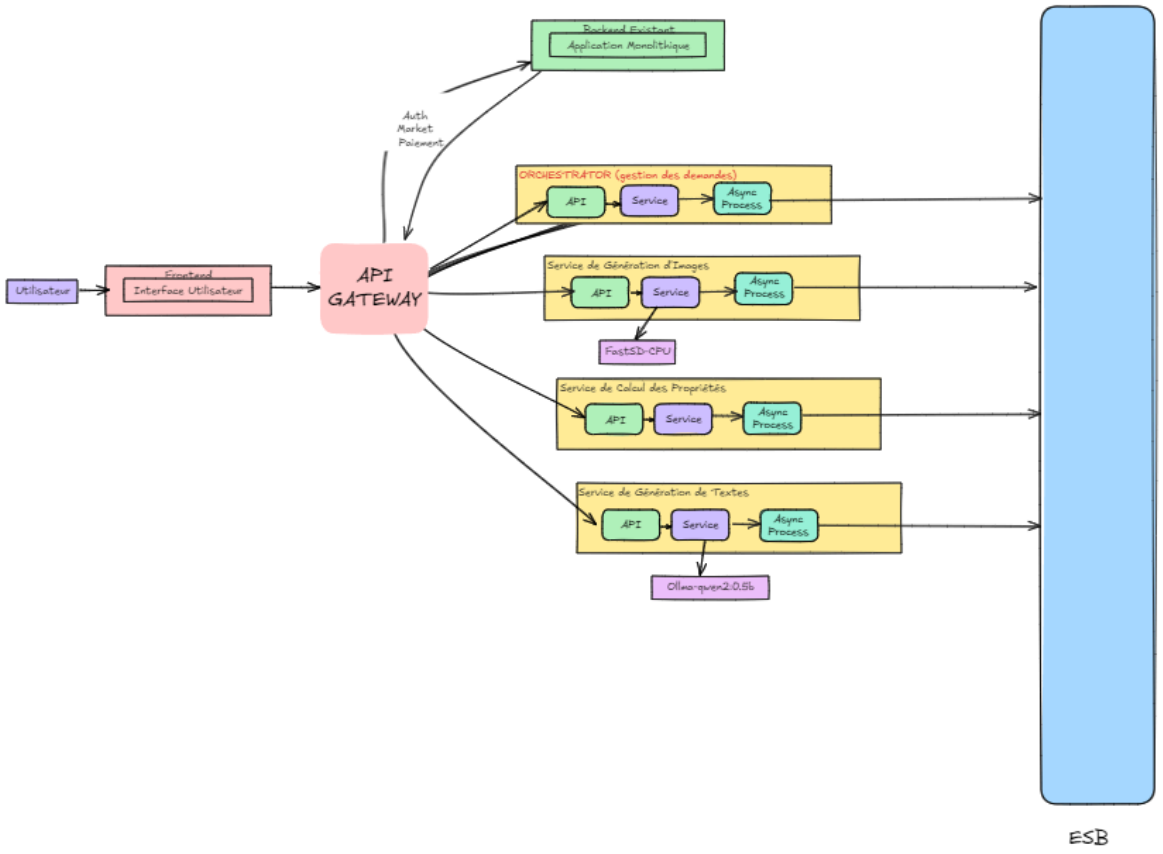
1.2. Réaliser un tableau récapitulatif des bus de communication les plus répandus (avantages/inconvénients)

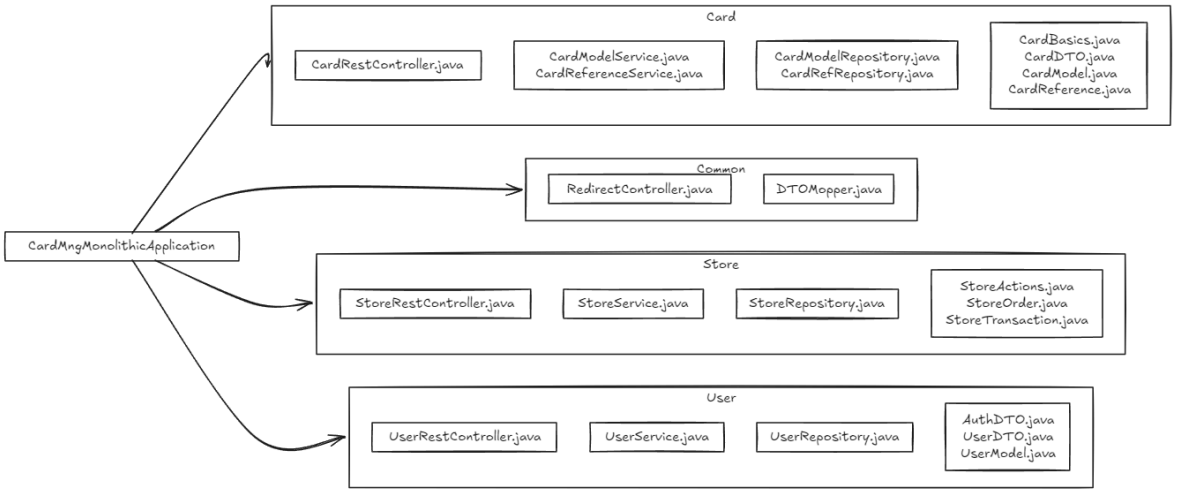
Nom du bus	Description brève	Avantages	Inconvénients
REST	Architecture basée sur HTTP permettant la communication entre applications via des services web légers.	Simplicité, ubiquité, utilise HTTP standard, facile à mettre en œuvre.	Pas adapté pour les communications en temps réel ou bidirectionnelles, manque de standardisation stricte.
SOAP	Protocole basé sur XML pour l'échange de données structurées entre applications via HTTP, SMTP, etc.	Standardisé, extensible, supporte WS-Security, transactions.	Complexité, surcharge due à XML, performances moindres par rapport à REST.
AMQP	Protocole de messagerie orienté messages pour les communications asynchrones entre applications.	Fiabilité, gestion des files d'attente, transactions, sécurité intégrée.	Complexe à mettre en œuvre, nécessite un broker compatible AMQP (ex : RabbitMQ).
MQTT	Protocole léger de messagerie publish/subscribe pour les réseaux contraints et les appareils IoT.	Faible encombrement, efficace pour les appareils à ressources limitées, supporte les communications en temps réel.	Moins sécurisé par défaut, fonctionnalités limitées par rapport à d'autres protocoles.
JMS	API Java pour la communication asynchrone basée sur les messages entre applications Java.	Intégration native avec Java, supporte les modèles point-à-point et publish/subscribe.	Limité à l'écosystème Java, nécessite un broker JMS.
Apache Kafka	Plateforme de streaming distribuée pour la gestion des flux de données en temps réel.	Haute performance, évolutivité, persistance des messages, traitement en temps réel.	Complexité de déploiement et de gestion, courbe d'apprentissage élevée.
RabbitMQ	Broker de messages open-source implémentant plusieurs protocoles de messagerie (AMQP, MQTT, STOMP).	Flexible, supporte de multiples protocoles, fiable, bonnes performances.	Peut être complexe à configurer, nécessite une gestion adéquate des files et des échanges.

WebSocket	Protocole de communication bidirectionnelle sur TCP permettant des interactions en temps réel entre un client et un serveur.	Communication en temps réel, faible latence, réduit l'encombrement réseau en évitant les requêtes HTTP répétées.	Pas adapté pour les communications stateless, support variable selon les environnements.
gRPC	Framework RPC open-source initialement développé par Google, utilisant Protocol Buffers pour la sérialisation de données.	Haute performance, sérialisation efficace, supporte les communications bidirectionnelles en streaming.	Moins convivial pour le débogage, support limité pour certains langages ou plateformes, complexité d'implémentation.
ZeroMQ	Bibliothèque de messagerie asynchrone haute performance, offrant une file d'attente de messages pour la communication entre processus ou sur le réseau.	Très rapide, flexible, sans broker intermédiaire, supporte plusieurs modèles de communication.	Ne garantit pas la fiabilité des messages, nécessite une gestion manuelle de certaines fonctionnalités (sécurité, persistance).
CoAP	Protocole web léger conçu pour les appareils contraints, basé sur le modèle REST et utilisant UDP.	Optimisé pour les appareils à faible puissance, supporte les interactions RESTful, faible surcharge.	Fonctionnalités limitées, moins fiables en raison de l'utilisation de UDP, moins répandu.
OPC UA	Protocole standard pour la communication industrielle et l'échange de données sécurisées dans les systèmes d'automatisation.	Sécurité intégrée, interopérabilité, extensibilité, adapté pour l'IoT industriel.	Complexité, nécessite une infrastructure adéquate, surcharge possible pour les petits systèmes.

Tableau comparatif des framework frontend

Nom	**React**	**Angular**	**Vue JS**	**Svelte**
Points forts	Grande communauté Flexibilité Nombreuses librairies	Structure stricte Outils intégrés	Simple à apprendre Polyvalent Documentation claire	Performance Code minimal Pas de Virtual DOM
Points faible	Difficile d'apprentissage	Complexe Verbeux Lourd	Écosystème plus petit Moins de ressources	Communauté plus petite Moins de ressources


Backend Existant
Application Monolithique
Auth
Market
Paiement
ORCHESTRATOR (gestion des demandes)
API
Service
Async Process
Utilisateur
Frontend
Interface Utilisateur
API GATEWAY
Service de Génération d'Images
API
Service
Async Process
FastSD-CPU
Service de Calcul des Propriétés
API
Service
Async Process
Service de Génération de Textes
API
Service
Async Process
Ollma-qwen2:0.5b
ESB


CardMngMonolithicApplication
Card
CardRestController.java
CardModelService.java
CardReferenceService.java
CardModelRepository.java
CardRefRepository.java
CardBasics.java
CardDTO.java
CardModel.java
CardReference.java
Common
RedirectController.java
DTOMapper.java
Store
StoreRestController.java
StoreService.java
StoreRepository.java
StoreActions.java
StoreOrder.java
StoreTransaction.java
User
UserRestController.java
UserService.java
UserRepository.java
AuthDTO.java
UserDTO.java
UserModel.java