

# **Architecture Logicielle "VendreFacile"**

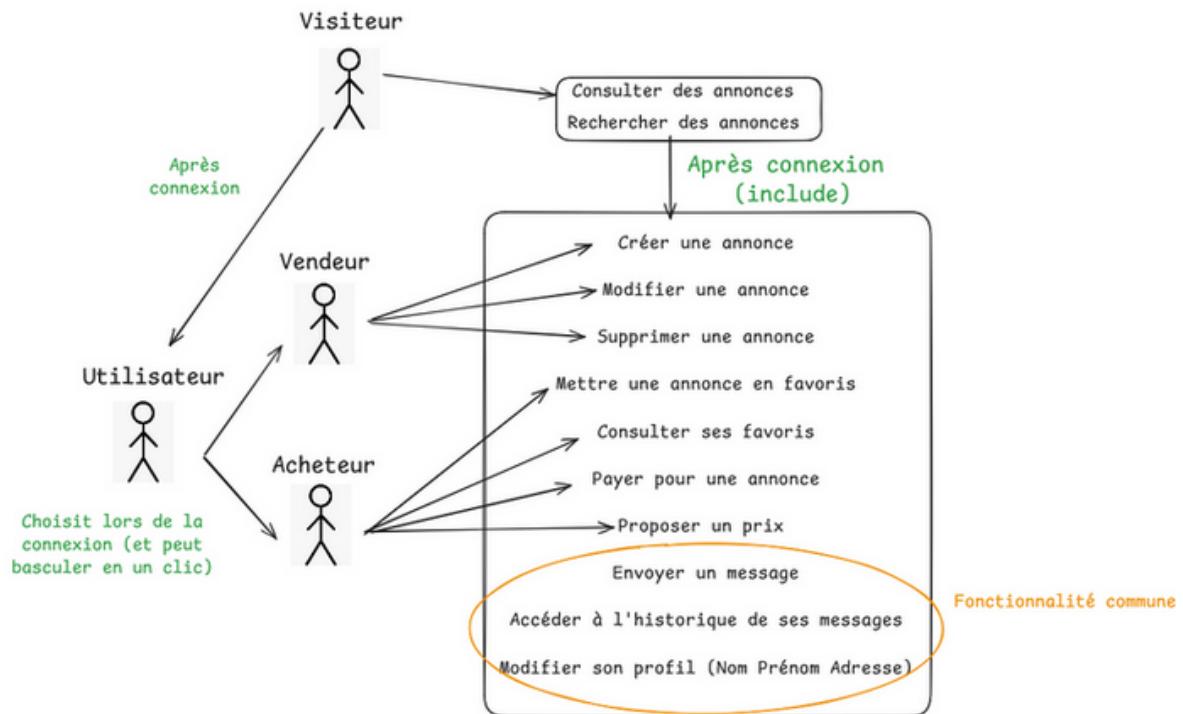
Cahier des charges client – Projet

**VendreFacile**

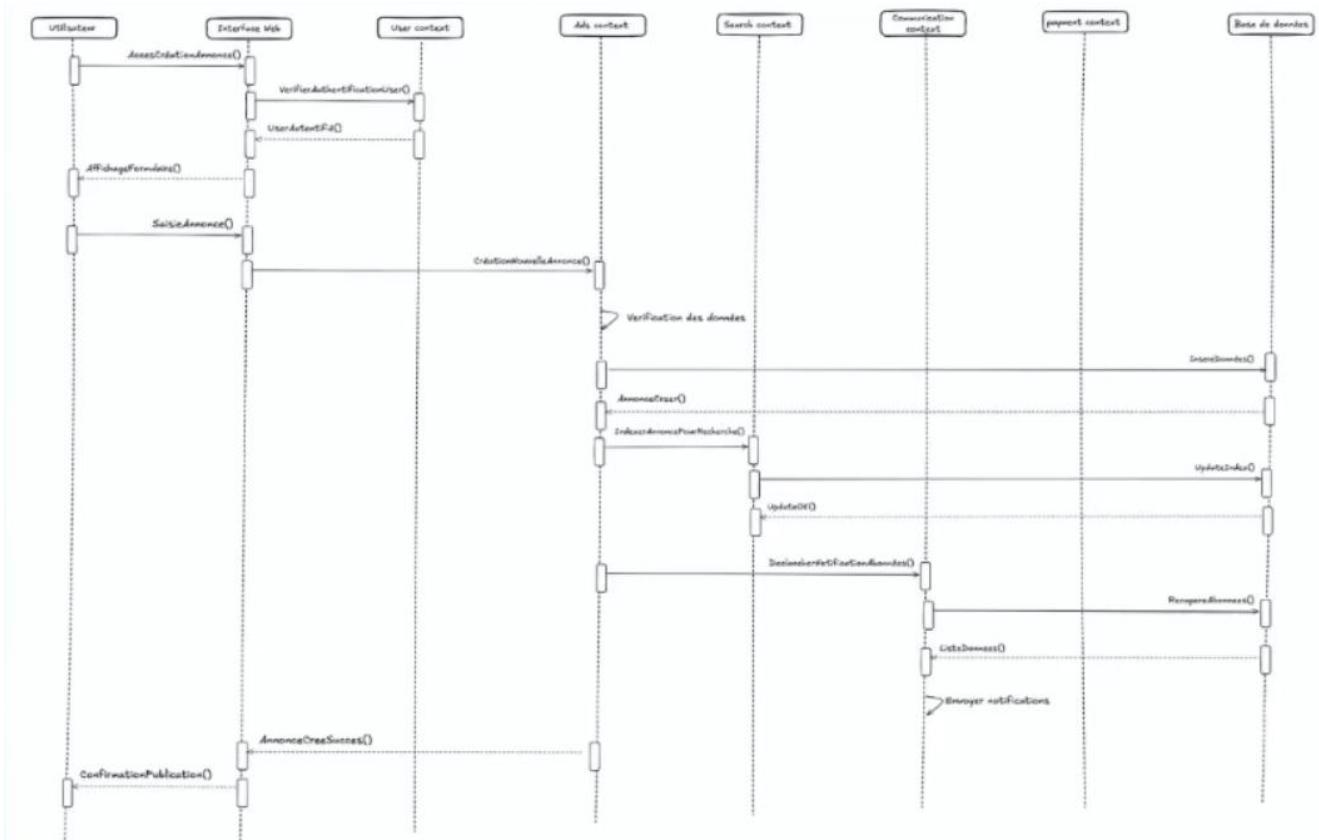


# Vu d'ensemble du projet

## Diagramme de cas d'usage – User case



## Diagramme de séquence



# Analyse DDD

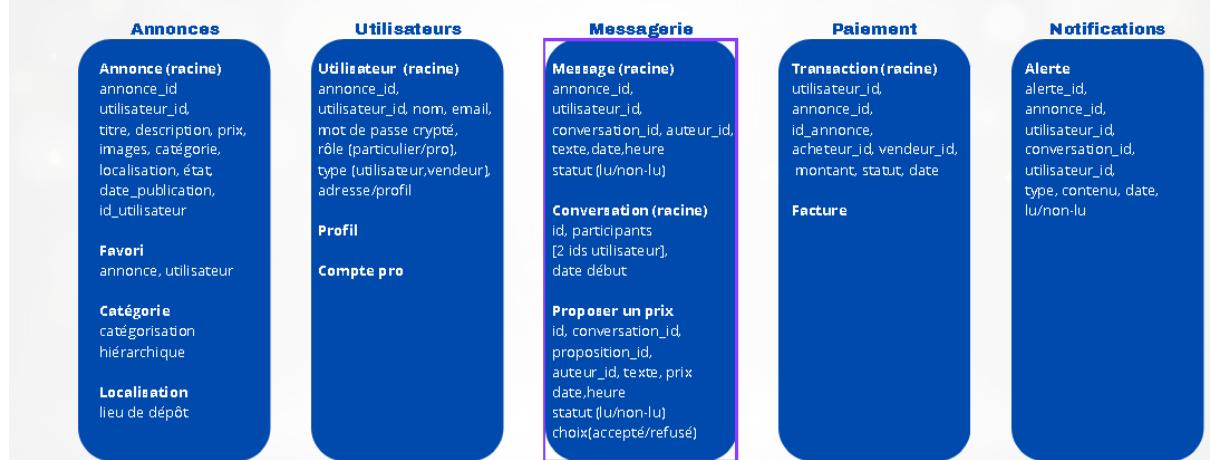
## Bounded-contexts

<b>annonces</b>	Création, gestion, recherche, affichage des annonces ; géolocalisation ; favoris
<b>utilisateur</b>	Gestion des comptes utilisateurs (inscription, profil, authentification, comptes pro)
<b>Messagerie</b>	Gestion de la messagerie privée entre acheteurs et vendeurs
<b>Paiement</b>	Gestion des paiements sécurisés et process associés
<b>Notifications</b>	Envoi d'alertes par email/notification

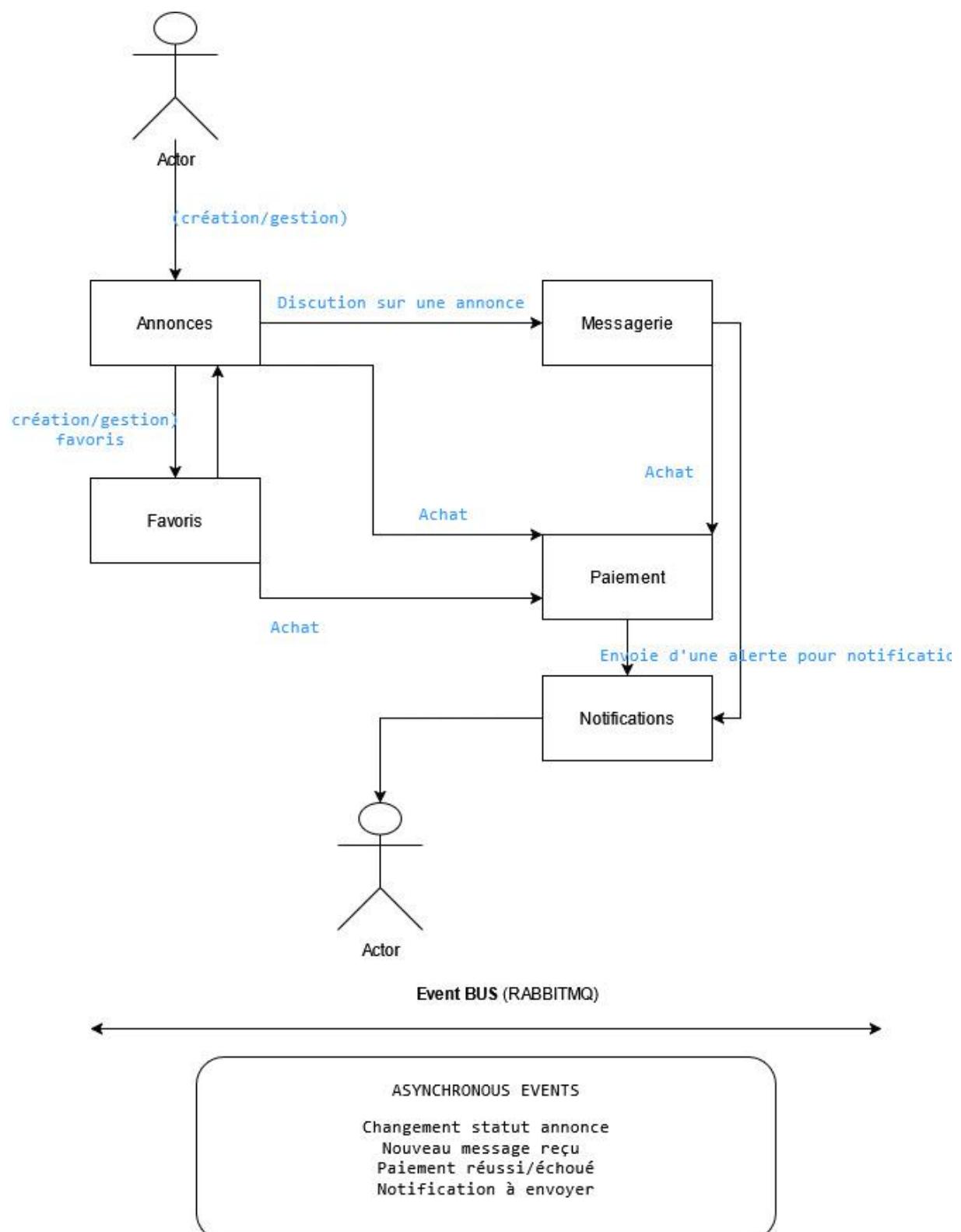
## Vocabulaire ubiquitaire

<b>Announce</b>	Offre publiée par un utilisateur, contenant un bien ou service à vendre ou à donner
<b>Utilisateur</b>	Personne ou entreprise qui possède un compte sur la plateforme
<b>Visiteur</b>	Personne ou entreprise qui ne possède pas un compte sur la plateforme
<b>Favori</b>	Statut permettant de marquer une annonce
<b>Conversation</b>	Fil de discussion privé entre deux utilisateurs portant sur une ou plusieurs annonces
<b>Message</b>	Unité de communication échangée dans une conversation
<b>Transaction</b>	Processus de paiement associant un acheteur à un vendeur et une annonce donnée

## Agrégats



## Context map



## Liste des fonctions par pages

### 1. Page d'Accueil / Home/Recherche

- chargementAnnonces()
- rechercheEffectuee()
- filtreApplique()
- ajoutFavori()

### 4. Page Profil Utilisateur

- inscriptionUtilisateur()
- connexionUtilisateur()
- deconnexionUtilisateur()
- modificationProfil()
- suppressionCompte()

### 2. Page Annonce

- ajoutFavori()
- retraitFavori()

### 5. Page Messagerie / Conversation

- creationConversation()
- envoiMessage()
- lectureMessage()
- suppressionConversation()
- PropositionPrix()

### 3. Page gestion d'Annonce

- RédactionAnnonce()
- validationAnnonce()
- ConsultationAnnonce()
- ModificationAnnonce()
- SuppressionAnnonce()

### 6. Page Favoris

- affichageFavoris()
- retraitFavori()
- ajoutFavori()

### 7. Page Paiement (option souhaitée)

- initialisationPaiement()
- paiementEffectue()
- paiementEchoeue()
- transactionConfirmee()

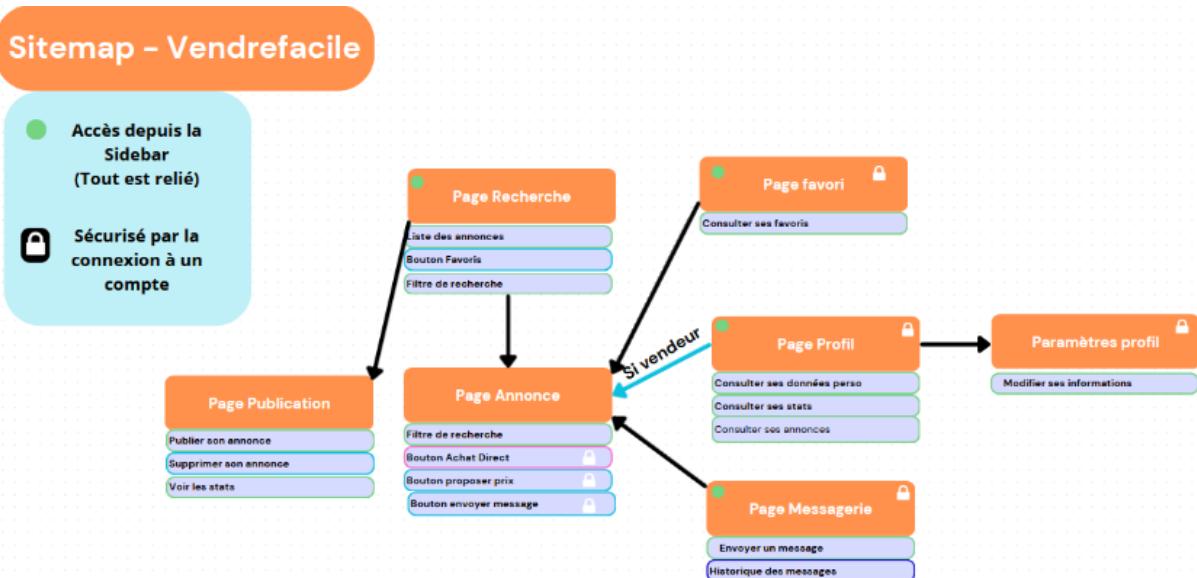


# Architecture frontend et backend

## Frontend Maquette

```
VendreFacile-mockup/
├── index.html          # Page d'accueil avec recherche
├── annonce.html        # Page détail d'une annonce
├── publication.html    # Page de publication d'annonce
├── profil.html         # Page profil utilisateur
├── messagerie.html     # Page messagerie/conversation
├── favoris.html         # Page favoris
├── paiement.html        # Page paiement sécurisé
└── css/
    └── style.css          # Styles principaux
└── js/
    └── script.js          # Fonctions JavaScript
README.md
```

## Sitemap





# Cahier des charges du développement final

## Fonctionnalités souhaitées

### Page d'Accueil / Recherche

- chargementAnnonces() - Chargement et affichage des annonces
- rechercheEffectuee() - Recherche par mots-clés, catégorie et localisation
- filtreApplique() - Filtrage par prix min/max
- Affichage en grille responsive
- Pagination simulée
- Catégories prédéfinies

### Page Annonce Détailnée

- ajoutAnnonce() - Redirection vers publication
- modificationAnnonce() - Modification d'annonce existante
- suppressionAnnonce() - Suppression avec confirmation
- ajoutFavori() - Ajout aux favoris avec feedback visuel
- retraitFavori() - Retrait des favoris
- Galerie d'images interactive
- Informations détaillées du vendeur
- Géolocalisation simulée

### Page Publication d'Annonce

- redactionAnnonce() - Formulaire de création complet
- validationAnnonce() - Validation et publication
- modificationAnnonce() - Modification de brouillon
- suppressionAnnonce() - Suppression de brouillon
- Upload d'images simulé
- Prévisualisation avant publication
- Sauvegarde automatique en brouillon

### Page Profil Utilisateur

- inscriptionUtilisateur() - Formulaire d'inscription
- connexionUtilisateur() - Formulaire de connexion
- deconnexionUtilisateur() - Déconnexion sécurisée
- modificationProfil() - Modification des informations
- suppressionCompte() - Suppression avec confirmation
- Statistiques utilisateur
- Historique des annonces
- Gestion des paramètres de confidentialité

## Page Messagerie / Conversation

- creationConversation() - Nouvelle conversation
- envoiMessage() - Envoi de message en temps réel
- lectureMessage() - Marquage comme lu
- suppressionConversation() - Suppression avec confirmation
- propositionPrix() - Proposition de prix intégrée
- Interface de chat temps réel
- Réponses automatiques simulées
- Notifications de messages

## Page Favoris

- affichageFavoris() - Affichage avec tri et filtre
- retraitFavori() - Retrait avec animation
- ajoutFavori() - Ajout depuis autres pages
- Tri par date, prix, localisation
- Mode liste/grille
- Actions groupées
- État vide avec call-to-action

## Page Paiement (Fonctionnalité Premium)

- initialisationPaiement() - Initialisation sécurisée
- paiementEffectue() - Confirmation de paiement
- paiementEchoue() - Gestion des erreurs
- transactionConfirmee() - Confirmation finale
- Résumé de commande détaillé
- Multiples moyens de paiement
- Adresse de facturation
- Badges de sécurité

## Fonctionnalités transversales

### Système de notifications

Notifications toast pour toutes les actions

Types : succès, erreur, information

Animation d'apparition/disparition

## Design responsive

Adaptation mobile complète

Navigation hamburger sur mobile

Grilles adaptatives

Touch-friendly sur tous les éléments

 Interface utilisateur

Design moderne et épuré

Animations CSS fluides

États hover/focus bien définis

Accessibilité de base

# Évolutions possibles

 Améliorations court terme

Interface utilisateur

Mode sombre - Thème dark avec switch

Animations avancées - Micro-interactions, transitions de page

Skeleton loading - Indicateurs de chargement élégants

Drag & drop - Upload d'images par glisser-déposer

## ## Fonctionnalités utilisateur

Système de notation - Évaluation vendeurs/acheteurs

Comparateur d'annonces - Comparaison côté à côté

Historique de navigation - Annonces récemment consultées

## ## Recherche et filtres

Filtres avancés - Date, état, livraison, etc.

Suggestions automatiques - Autocomplétion intelligente

Géolocalisation avancée - Recherche par rayon, carte interactive

## ## 🚀 Évolutions moyen terme

### ## Architecture technique

Migration vers React/Vue.js - Framework moderne

Progressive Web App (PWA) - Installation, notifications push

Service Workers - Cache avancé, synchronisation hors ligne

WebSockets - Messagerie temps réel

## ## Fonctionnalités avancées

Détection de fraude - Algorithmes de sécurité

Système d'enchères - Ventes aux enchères

Livraison intégrée - Partenariats transporteurs

## ## Expérience utilisateur

Gamification - Points, badges, niveaux

Analytics utilisateur - Statistiques détaillées

## ## 🌟 Évolutions long terme

Intégrations externes

API Google Maps - Géolocalisation réelle

Stripe/PayPal - Paiements réels

API SMS - Vérification par SMS

API Email - Envoi d'emails automatisés

API Réseaux sociaux - Connexion Facebook/Google  
API Reconnaissance d'images - Classification automatique  
API GOOGLE - Création et connexion à un compte simple

## **## Fonctionnalités business**

Comptes professionnels - Fonctionnalités entreprise  
Abonnements premium - Modèle freemium  
Publicités ciblées - Monétisation  
Analyses business - Dashboard administrateur  
Multi-langues - Internationalisation i18n  
Multi-devises - Support international

## **## Technologies émergentes**

Assistant vocal - Intégration Alexa/Google Assistant  
Blockchain - Vérification d'authenticité  
IoT - Objets connectés  
Machine Learning - Prédictions de prix  
Reconnaissance faciale - Vérification d'identité

## **## Évolutions backend (pour intégration future)**

Infrastructure

Microservices - Architecture distribuée  
RABBITMQ - Gestion des relations entre les microservices  
Load balancing - Répartition de charge  
Base de données distribuée - Scalabilité MariaDB  
Containerisation - Docker/Kubernetes

## **## Sécurité**

Authentification 2FA - Double authentification  
OAuth 2.0 - Authentification sécurisée  
Chiffrement E2E - Messages chiffrés  
Audit de sécurité - Conformité RGPD  
Rate limiting - Protection contre le spam

# Evolution et migration vers nodeJS

## VendreFacile - Structure Simplifiée du Projet

### 📁 Structure Globale

```
vendrefacile-microservices/
├── README.md
├── docker-compose.yml
├── .env.example
├── scripts/
├── docs/
└── infrastructure/
    ├── shared/
    └── services/
        └── frontend/
```



### 🚀 Services (Microservices)

```
services/
├── api-gateway/      # Point d'entrée unique
├── auth-service/     # Authentification & autorisation
├── user-service/     # Gestion des utilisateurs
├── announcement-service/ # Annonces et catégories
├── message-service/   # Messages et conversations
├── payment-service/   # Paiements et transactions
├── notification-service/ # Notifications (email, push, SMS)
├── analytics-service/ # Analytics et rapports
└── file-service/      # Gestion des fichiers
```



### 🏗 Infrastructure

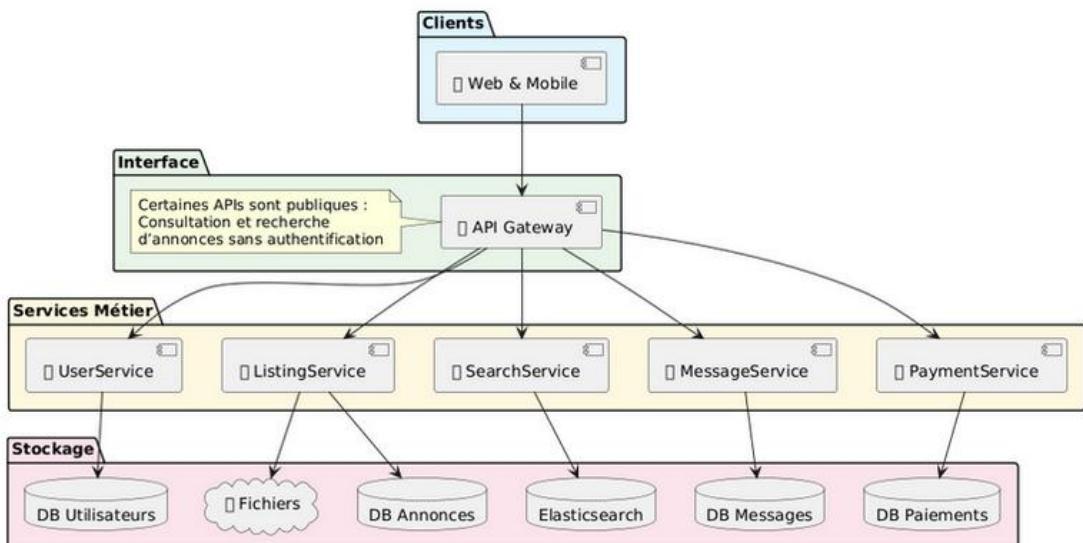
```
infrastructure/
└── docker/
    ├── nginx/          # Reverse proxy
    ├── mariadb-galera/  # Cluster base de données
    ├── redis/           # Cache et sessions
    ├── rabbitmq/        # Message broker
    └── elasticsearch/  # Moteur de recherche
└── k8s/               # Kubernetes
└── terraform/         # Infrastructure as code
└── monitoring/        # Prometheus, Grafana
└── ci-cd/             # Pipeline CI/CD
```



## Structure Type d'un Service

```
auth-service/
├── package.json
├── Dockerfile
└── src/
    ├── controllers/      # Contrôleurs API
    ├── services/         # Logique métier
    ├── repositories/     # Accès aux données
    ├── models/           # Modèles de données
    ├── routes/           # Routes API
    ├── events/           # Gestionnaires d'événements
    ├── config/           # Configuration
    └── utils/            # Utilitaires
    tests/               # Tests unitaires
    docs/                # Documentation
```

## Architecture



## **Avantages de RabbitMQ avec Exemples**

### **1. Découplage des services**

*Exemple :*

Quand un utilisateur publie une annonce, le service « Annonces » envoie un message. Le service « Notifications » reçoit ce message et envoie automatiquement un e-mail, sans que « Annonces » ait besoin de connaître « Notifications ».

### **2. Gestion des pics de trafic**

*Exemple :*

Si 1000 utilisateurs valident un paiement en même temps, RabbitMQ garde les messages en file d'attente. Le service « Paiement » les traite au fur et à mesure sans perdre d'informations.

### **3. Résilience**

*Exemple :*

Si le service « Notifications » est en panne, les messages attendent dans RabbitMQ et seront traités dès qu'il redémarre. Aucun événement n'est perdu.



## **Points clés mis en avant :**

### **Structure claire**

- **4 couches** bien définies avec leurs responsabilités
- **Flux de données** explicites entre les composants
- **Séparation des préoccupations** respectée

### **Valeur business**

- **Scalabilité** pour supporter la croissance
- **Résilience** face aux pannes
- **Évolutivité** progressive (monolithe → microservices)

### **Aspects techniques**

- **APIs publiques** pour consultation sans auth
- **Microservices** spécialisés et indépendants
- **Données adaptées** aux besoins (SQL + Elasticsearch)

### **Vision d'évolution**

- **Approche progressive** rassurante
- **Pas de sur-ingénierie** dès le début
- **Adaptation** selon les besoins réels

## **Méthode de management / développement**

### **MÉTHODE AGILE ET KISS**

#### **POURQUOI AGILE ?**

##### **Contexte VendreFacile**

- **Marché concurrentiel** donc besoin **adaptabilité rapide**
- **Besoins évolutifs** : fonctionnalités à ajuster selon utilisateurs

##### **Bénéfices Agile**

- **Livraison fréquentes**
- **Feedback réguliers (sprint)**
- **Adaptation**
- **Collaboration**

##### **POURQUOI "KISS" ?**

## **Objectifs**

- **Démarrage rapide et économique**
- **Permettra moins de bugs**
- **Une maintenance plus simple à mettre en place**

## **Application**

- **Code simple**
- **Éviter le "surdév"**
- **Déploiement simple**

## **PLAN D'APPLICATION**

### **Phase 1 : Démarrage Simple (KISS)**

- Application monolithique
- PostgreSQL/MariaDB seul
- Fonctionnalités essentielles uniquement
- Déploiement simple

### **Phase 2 : Itérations Agile**

- Sprints de 2 semaines
- Feedback utilisateurs réguliers
- Ajustements fonctionnels
- Tests et validation continue

### **Phase 3 : Évolution Progressive**

- Ajout de complexité selon les besoins mesurés
- Scalabilité progressive
- Optimisations ciblées

## Notre différence

### **NOTRE DIFFÉRENCE**

**Filtres poussés : Filtre exclusif (Exclure un endroit), Filtre logistique (Permifié), Distance maximum**

**Modération très présente (Eviter achat revente Aliexpress, fausses annonces...)**

**Recommandation sur les achats => Achat d'un vélo, proposition d'un antivol**

**Possibilité d'enregistrer des tags**

