Il faut noter que dans ce projet, on s'en fout complètement de la sécurité, ne jamais prendre l'aspect sécurité en compte, on veut juste un prototype d'api qui fonctionne en utilisant MySQL via XAMPP et Spring Boot, et juste écrire le code du strict nécessaire. Et le strict nécessaire des tests pour vérifier que l'API fonctionne.

🔷 Présentation du Projet : Plateforme de Réseau Social Immobilier au Cameroun

Ce projet consiste en la création d'un **réseau social immobilier** dédié au marché camerounais. Il a pour but de **connecter des acquéreurs et des annonceurs** de logements (chambres, appartements, studios) à **louer ou à vendre**, tout en permettant aux utilisateurs d'interagir via des publications, des notifications, des messages et des transactions sécurisées. Chaque **annonce** sur la plateforme peut contenir **jusqu'à 5 photos ou vidéos**, permettant ainsi aux acquéreurs de se faire une idée claire du bien proposé.

🔹 Les 4 grands acteurs du système

1. 👤 Acquéreur

C'est une personne qui arrive sur le site dans le but de consulter des offres immobilières. Il peut :

- Naviguer librement parmi les annonces.
- Demander à visiter un logement.
- Faire une **demande de réservation** ou un **achat**.
- Éventuellement devenir lui-même **annonceur** (voir plus bas).
- Enregistrer des annonces dans ses **favoris**.
- Recevoir des notifications ou échanger par message.

---

2. 📢 Annonceur

L'annonceur est un **utilisateur vérifié et abonné** qui peut :

- Publier ses propres annonces de location ou de vente.
- Ajouter des photos/vidéos, une description, un prix et un statut à chaque bien.
- Consulter les **statistiques de ses annonces** (nombre de vues, favoris, interactions).
- Être notifié lorsqu'un acquéreur souhaite **réserver, visiter ou acheter** un bien.
- Répondre aux messages des acquéreurs.
- Gérer son calendrier de **disponibilité pour les visites**.
- Modifier ou supprimer ses annonces à tout moment.

---

3. 🛠️ Administrateur

L'administrateur agit comme **intermédiaire de confiance**. Il est chargé de :

- **Valider ou refuser** les annonces avant publication.
- **Gérer les comptes utilisateurs** (création, suspension, suppression).

- **Gérer les paiements** : il reçoit les paiements des acquéreurs, puis reverse les sommes dues aux annonceurs.
- Transmettre les décisions importantes (validation, refus, calendrier) par email ou appel.
- Agir un peu comme un **notaire ou avocat numérique**, assurant que tout se passe selon les règles.

---

4. 📄 Le Système

C'est la **plateforme web elle-même**, qui automatise :

- L'affichage des annonces.
- L'envoi de formulaires.
- La génération de contrats et de documents.
- La gestion des profils, des paiements et des notifications.

🔷 Étapes pour devenir annonceur

Un acquéreur peut à tout moment devenir un annonceur grâce à ce processus :

1. Il clique sur **"Devenir annonceur"**.
2. Le **système lui envoie un formulaire** avec les détails et **le tarif de l'abonnement**.
3. L'acquéreur **remplit le formulaire**, effectue le **paiement** et le renvoie.
4. L'**administrateur vérifie** les informations fournies.
5. S'il valide, le **système crée un compte annonceur** et une interface personnalisée.
6. L'ancien acquéreur reçoit un **message d'autorisation** confirmant qu'il peut désormais publier des annonces.

🔷 Étapes pour publier une annonce

Une fois devenu annonceur, voici comment publier une annonce :

1. L'annonceur **demande à créer une annonce** via le système.
2. Le système lui renvoie un **formulaire à remplir** (titre, description, photos, prix...).
3. Il remplit le formulaire et l'envoie.
4. L'**administrateur reçoit la demande** :
    - Il peut **refuser** l'annonce en indiquant un **motif** clair.
    - Ou **l'accepter** et la publier.
5. En cas d'acceptation, le système **notifie l'annonceur** que l'annonce est désormais visible.

🔷 Étapes d'un achat ou d'une réservation

Voici comment un acquéreur procède pour réserver ou acheter un bien :

1. L'acquéreur **consulte les annonces** proposées par le système.
2. Il **demande une visite**.

3. Le **système contacte l'administrateur**, qui :
    - **Informe l'annonceur** de la demande.
    - L'annonceur propose un jour, ou renvoie d'autres dates disponibles.
4. L'administrateur **confirme la date au client** par email ou appel.

Si l'acquéreur est convaincu :

5. Il envoie une **demande d'achat ou de réservation**.
6. Le système lui retourne un **contrat et un formulaire** à remplir.
7. L'acquéreur **valide le formulaire et joint une preuve de paiement** (photo ou capture).
8. Le tout est **envoyé à l'administrateur**.

Deux scénarios possibles :

- **Paiement refusé** (preuves manquantes, erreurs…) → L'acquéreur reçoit un **message de refus**.
- **Paiement validé** → L'administrateur :
    - Reverse l'argent à l'annonceur.
    - Le système **envoie une confirmation** de paiement à l'acquéreur.
    - Le **statut de l'annonce passe à "VENDU" ou "RÉSERVÉ"** automatiquement.

Conclusion

Ce projet est une **plateforme complète et interactive** qui combine les fonctionnalités classiques d'un site immobilier avec celles d'un **réseau social intelligent** : messagerie, notifications, tableaux de bord, avis, évaluations et publications multimédias.
Grâce à une **interface simple et structurée**, chaque utilisateur peut :

- Trouver un logement facilement.
- Proposer ses biens à la location/vente.
- Gérer ses interactions et paiements en toute sécurité.

Diagramme de séquence :

Diagramme de séquence 1 : Achat ou réservation

Acteurs / objets :

- **Acquéreur**
- **Système**
- **Administrateur**
- **Annonceur**

Interactions :

1. **Acquéreur → Système** : *consulter les annonces*

2. **Système → Acquéreur** : *présentation des annonces*
3. **Acquéreur → Système** : *demande visite*
4. **Système → Administrateur** : *annoncer jour visite*
5. **Administrateur → Annonceur** : *accepter ou renvoyer les jours disponibles*
6. **Système → Acquéreur** : *contacter par mail ou appel*
7. **Acquéreur → Système** : *demande d'achat ou réservation*
8. **Système → Acquéreur** : *renvoie du contrat + formulaire d'achat ou de réservation*
9. **Acquéreur → Système** : *validation du formulaire avec capture de paiement*

Bloc `alt` (alternative) :

- **Condition : achat refusé**
  - **Système → Acquéreur** : *message de refus de paiement*
- **Condition : achat validé**
  - **Système → Administrateur** : *confirmation de paiement*
  - **Administrateur → Annonceur** : *versement de son dû pour la vente*
  - **Système → Acquéreur** : *message de confirmation de paiement*

🔷 Diagramme de séquence 2 : Devenir Annonceur

Acteurs / objets :

- **Acquéreur**
- **Système**
- **Administrateur**
- **Annonceur**

Interactions :

1. **Acquéreur → Système** : *demande devenir annonceur*
2. **Système → Acquéreur** : *envoie du formulaire et tarif d'abonnement*
3. **Acquéreur → Système** : *payer abonnement et renvoyer formulaire*
4. **Système → Administrateur** : *paiement reçu*
5. **Système → Acquéreur** : *création nouvelle interface annonceur*
6. **Système → Acquéreur** : *message d'autorisation de publier*
7. **Administrateur → Annonceur** : *(création d'un annonceur, noté `<<create>>`)*

🔷 Diagramme de séquence 3 : Publier une Annonce

Acteurs / objets :

- **Acquéreur**
- **Système**
- **Administrateur**
- **Annonceur**

Interactions :

1. **Annonceur → Administrateur** : *demander à faire une annonce*
2. **Administrateur → Annonceur** : *renvoie du formulaire à remplir*
3. **Annonceur → Administrateur** : *proposer annonce*

Bloc `alt` (alternative) :

- **Condition : annonce rejetée**
  - **Administrateur → Annonceur** : *rejeter annonce avec motif*
- **Condition : annonce acceptée**
  - **Administrateur → Système** : *publier annonce*
  - **Système → Annonceur** : *message d'annonce publiée*

Diagramme de classes :

🔷 Classes principales et leurs relations

✅ Classe Utilisateur (classe abstraite)

Attributs :

- `idUtilisateur : int`
- `nom : String`
- `email : String`
- `motDePasse : String`
- `role : Enum {ADMIN, ANNONCEUR, ACQUEREUR}`
- `numPhone : int`
- `formatNumPays : int`
- `photoProfil : BufferedImage`

Méthodes :

- `+sInscrire()`
- `+seConnecter()`
- `+modifierProfil()`
- `+supprimerCompte()`

✅ Classe Acquéreur *(hérite de Utilisateur)*

Attributs :

- `favoris : List`
- `historiqueRecherches : List`

Méthodes :

- `+signerContratVente(contrat : Contrat)`
- `+signerPromesseVente(contrat : Contrat)`
- `+rechercherAnnonce(filtres : Map<String, String>)`
- `+consulterListeContratNonCouru() : List<Contrat>`
- `+devenirAcquereur()`
- `+sePlaindre(date : String) : Plainte`

- `+effectuerPaiement(paiement : Paiement, photo : BufferedImage) : boolean`
- `+consulterListePaiement() : List<Paiement>`
- `+ecrireMessage(message : String, destinataire : Utilisateur)`

✓ Classe Annonceur *(hérite de Utilisateur)*

Attributs :
- `numeroMTN_MoMo : int`
- `numeroOrangeMoney : int`
- `numeroUBA : int`

Méthodes :
- `+consulterStatistiques(annonce : Annonce) : Statistiques`
- `+redigerDocumentJuridique(type : String, params : Map<String, Object>) : Document`
- `+consulterSesAnnonces() : List<Annonce>`
- `+consulterBilanPromessesEtVente() : Rapport`
- `+consulterCalendrierPourVisites(dates : List<Date>)`
- `+annulerAnnonce(idAnnonce : int)`
- `+consulterListeContratAvecOuValide(decision : boolean) : List<Contrat>`
- `+consulterTousSesPaiements() : List<Paiement>`

✓ Classe Administrateur

Attributs :
- `numeroMTN_MoMo : int`
- `numeroOrangeMoney : int`
- `numeroUBA : int`

Méthodes :
- `+gererUtilisateurs(utilisateur : Utilisateur, action : String)`
- `+gererAnnonces(annonce : Annonce, action : String)`
- `+analyserStatistiques() : Rapport`
- `+envoyerEmail(utilisateur : Utilisateur, message : String)`
- `+consulterContrat() : List<Contrat>`
- `+consulterPlainte() : List<Plainte>`
- `+consulterTousLesRecuPaiement() : List<Paiement>`
- `+validerPaiement(paiement : Paiement)`
- `+virementVersAnnonceur(paiement : Paiement, annonceur : Annonceur)`

✓ Classe Annonce

Attributs :
- `idAnnonce : int`
- `titre : String`
- `description : String`
- `prixMensuel : float`

- localisation : String
- photos : List
- status : Enum {DISPONIBLE, RÉSERVÉ, VENDU}
- datePublication : DateTime
- annonceur : Annonceur
- tarifReservation : int
- contrat : Contrat
- nombreVue : int
- nombreFavori : int
- nombreVisite : int

Méthodes :

- +changerStatut(nouveauStatut : Enum)
- +planifierVisite(acquereur : Acquereur, date : Date)
- +contacterAnnonceur(acquereur : Acquereur, message : String)
- +ajouterFavori(acquereur : Acquereur)
- +modifierAnnonce(params : Map<String,Object>)
- +supprimerAnnonce()
- +notifierConvoite(acheteurOuReserveur : String, acquereur : Acquereur)

✓ Classe Contrat

Attributs :

- idContrat : int
- dateSignature : Date
- type : Enum {PROMESSE_VENTE, VENTE}
- dateDebutContrat : Date
- dateFinContrat : Date
- annonceContrat : Annonce
- precisionEnSurplus : String
- signataire : List
- decisionAcquereur : boolean

Méthodes :

- +ajouterSignataire(utilisateur : Utilisateur)
- +validerContrat() : boolean
- +telechargerContrat() : Document
- +refuserContrat(motif : String) : boolean

✓ Classe Paiement

Attributs :

- idPaiement : int
- montant : float
- datePaiement : DateTime
- moyenPaiement : Enum {CARTE_UBA, OM, MoMo}
- statut : Enum {PAYÉ, EN_ATTENTE, DESISTÉ}
- typeContrat : Enum {PROMESSE_VENTE, VENTE}
- annonceConvoite : Annonce
- payerAcquereur : Acquereur

- `receveurPayee : Annonceur`
- `capturePhotoPreuvePaiement : BufferedImage`
- `lu : boolean`

Méthodes :

- `+genererRecuPaiement() : Document`
- `+mettreÀJourDuStatut() : String`

✓ Classe Notifications

Attributs :

- `idNotification : int`
- `dateCreation : DateTime`
- `type : Enum {CONTRAT, PAIEMENT, CONVOITISE, VISITE}`
- `contenu : String`
- `lu : boolean`
- `expediteur : Utilisateur`
- `destinataire : Utilisateur`

Méthodes :

- `+marquerCommeLu()`
- `+supprimerNotification()`

✓ Classe Message

Attributs :

- `idMessage : int`
- `expediteur : Utilisateur`
- `destinataire : Utilisateur`
- `contenu : String`
- `dateEnvoi : DateTime`
- `lu : boolean`

Méthodes :

- `+envoyerMessage(expediteur : Acquereur, destinataire : Annonceur)`
- `+marquerCommeLu() : void`
- `+supprimerMessage() : void`

🔗 Associations principales :

- `Annonce` ↔ `Contrat` : 1 → 1
- `Annonceur` ↔ `Annonce` : 1 → 0..*
- `Contrat` ↔ `Acquéreur` : via `signataire : List`
- `Contrat` ↔ `Annonce` : 1 → 1
- `Paiement` ↔ `Annonceur` et `Acquéreur` : 1 → 1
- `Notifications`, `Message` : entre objets `Utilisateur`
- `Administrateur` peut gérer ou valider des objets via méthodes.

DESCRIPTION DE LA BASE DE DONNEES

```
-- Table utilisateur (table abstraite, implémentée via ses sous-classes)
CREATE TABLE utilisateur (
   id_utilisateur INT AUTO_INCREMENT PRIMARY KEY,
   nom VARCHAR(40) NOT NULL,
   email VARCHAR(40) NOT NULL UNIQUE,
   mot_de_passe VARCHAR(40) NOT NULL,
   role ENUM('ADMIN', 'ANNONCEUR', 'ACQUEREUR') NOT NULL,
   num_phone INT NOT NULL,
   format_num_pays INT NOT NULL,
   photo_profil LONGBLOB
);

-- Table acquéreur (hérite de utilisateur)
CREATE TABLE acquereur (
   id_acquereur INTEGER PRIMARY KEY,
   historique_recherches JSON, -- Stocké en JSON pour la liste
   FOREIGN KEY (id_acquereur) REFERENCES utilisateur(id_utilisateur) ON DELETE CASCADE
);

-- Table annonceur (hérite de utilisateur)
CREATE TABLE annonceur (
   id_annonceur INTEGER PRIMARY KEY,
   numero_mtn_momo INT,
   numero_orange_money INT,
   numero_uba INT,
   FOREIGN KEY (id_annonceur) REFERENCES utilisateur(id_utilisateur) ON DELETE CASCADE
);

-- Table administrateur (hérite de utilisateur)
CREATE TABLE administrateur (
   id_admin INTEGER PRIMARY KEY,
   numero_mtn_momo INT,
   numero_orange_money INT,
   numero_uba INT,
   FOREIGN KEY (id_admin) REFERENCES utilisateur(id_utilisateur) ON DELETE CASCADE
);

-- Table annonce
CREATE TABLE annonce (
   id_annonce INT AUTO_INCREMENT PRIMARY KEY,
   titre TEXT NOT NULL,
```

```
  description TEXT NOT NULL,
  prix_mensuel INTEGER NOT NULL,
  localisation TEXT NOT NULL,
  status TEXT NOT NULL CHECK (status IN ('DISPONIBLE', 'RÉSERVÉ', 'VENDU')),
  date_publication DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  tarif_reservation INTEGER,
  nombre_vue INTEGER DEFAULT 0,
  nombre_favori INTEGER DEFAULT 0,
  nombre_visite INTEGER DEFAULT 0,
  id_annonceur INTEGER NOT NULL,
  FOREIGN KEY (id_annonceur) REFERENCES annonceur(id_annonceur) ON DELETE
CASCADE
);

-- Table photos pour les photos des annonces (relation 1 à plusieurs)
CREATE TABLE photo (
  id_photo INTEGER AUTO_INCREMENT PRIMARY KEY,
  id_annonce INTEGER NOT NULL,
  photo LONGBLOB NOT NULL, -- Stockage de l'image
  est_video BOOLEAN, -- FALSE pour photo, TRUE pour vidéo
  FOREIGN KEY (id_annonce) REFERENCES annonce(id_annonce) ON DELETE CASCADE
);

-- Table contrat
CREATE TABLE contrat (
  id_contrat INTEGER AUTO_INCREMENT PRIMARY KEY,
  date_signature DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  type TEXT NOT NULL CHECK (type IN ('PROMESSE_VENTE', 'VENTE')),
  date_debut_contrat DATETIME,
  date_fin_contrat DATETIME,
  precision_en_surplus TEXT,
  decision_acquereur BOOLEAN DEFAULT FALSE, -- FALSE pour non, TRUE pour oui
  id_annonce INTEGER NOT NULL, -- Relation 1-1 avec Annonce
  FOREIGN KEY (id_annonce) REFERENCES annonce(id_annonce) ON DELETE CASCADE
);

-- Table de relation pour les signataires des contrats (relation plusieurs à plusieurs)
CREATE TABLE signataire (
  id_contrat INTEGER NOT NULL,
  id_utilisateur INTEGER NOT NULL,
  date_signature DATETIME,
  PRIMARY KEY (id_contrat, id_utilisateur),
  FOREIGN KEY (id_contrat) REFERENCES contrat(id_contrat) ON DELETE CASCADE,
  FOREIGN KEY (id_utilisateur) REFERENCES utilisateur(id_utilisateur) ON DELETE CASCADE
);
```

```
-- Table paiement
CREATE TABLE paiement (
   id_paiement INTEGER AUTO_INCREMENT PRIMARY KEY,
   montant INT NOT NULL,
   date_paiement DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   moyen_paiement TEXT NOT NULL CHECK (moyen_paiement IN ('CARTE_UBA', 'OM',
'MoMo')),
   statut TEXT NOT NULL CHECK (statut IN ('CONFIRMEE', 'EN_ATTENTE')),
   type_contrat TEXT NOT NULL CHECK (type_contrat IN ('PROMESSE_VENTE', 'VENTE')),
   capture_photo_preuve_paiement LONGBLOB NOT NULL,
   lu BOOLEAN DEFAULT FALSE, -- FALSE pour non lu, TRUE pour lu
   id_annonce INTEGER NOT NULL,
   id_acquereur INTEGER NOT NULL,
   id_annonceur INTEGER NOT NULL,
   FOREIGN KEY (id_annonce) REFERENCES annonce(id_annonce) ON DELETE CASCADE,
   FOREIGN KEY (id_acquereur) REFERENCES acquereur(id_acquereur) ON DELETE CASCADE,
   FOREIGN KEY (id_annonceur) REFERENCES annonceur(id_annonceur) ON DELETE
CASCADE
);

-- Table notification
CREATE TABLE notification (
   id_notification INTEGER AUTO_INCREMENT PRIMARY KEY,
   date_creation DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   type TEXT NOT NULL CHECK (type IN ('CONTRAT', 'PAIEMENT', 'VISITE', 'MESSAGE')),
   lu BOOLEAN DEFAULT FALSE, -- FALSE pour non lu, TRUE pour lu
   id_expediteur INTEGER NOT NULL,
   id_destinataire INTEGER NOT NULL,
   FOREIGN KEY (id_expediteur) REFERENCES utilisateur(id_utilisateur) ON DELETE CASCADE,
   FOREIGN KEY (id_destinataire) REFERENCES utilisateur(id_utilisateur) ON DELETE
CASCADE
);

-- Table message
CREATE TABLE message (
   id_message INTEGER AUTO_INCREMENT PRIMARY KEY,
   contenu TEXT NOT NULL,
   date_envoi DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   lu BOOLEAN DEFAULT 0, -- 0 pour non lu, 1 pour lu
   id_expediteur INTEGER NOT NULL,
   id_destinataire INTEGER NOT NULL,
   FOREIGN KEY (id_expediteur) REFERENCES utilisateur(id_utilisateur) ON DELETE CASCADE,
   FOREIGN KEY (id_destinataire) REFERENCES utilisateur(id_utilisateur) ON DELETE
CASCADE
```

```
);

-- Table favoris (relation plusieurs à plusieurs entre acquéreur et annonce)
CREATE TABLE favoris (
   id_acquereur INTEGER NOT NULL,
   id_annonce INTEGER NOT NULL,
   date_ajout DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (id_acquereur, id_annonce),
   FOREIGN KEY (id_acquereur) REFERENCES acquereur(id_acquereur) ON DELETE CASCADE,
   FOREIGN KEY (id_annonce) REFERENCES annonce(id_annonce) ON DELETE CASCADE
);

-- Table visite pour planifier les visites
CREATE TABLE visite (
   id_visite INTEGER AUTO_INCREMENT PRIMARY KEY,
   date_visite DATE NOT NULL,
   statut TEXT NOT NULL CHECK (statut IN ('DEMANDEE', 'CONFIRMEE', 'ANNULEE',
'EFFECTUÉE')),
   id_annonce INTEGER NOT NULL,
   id_acquereur INTEGER NOT NULL,
   FOREIGN KEY (id_annonce) REFERENCES annonce(id_annonce) ON DELETE CASCADE,
   FOREIGN KEY (id_acquereur) REFERENCES acquereur(id_acquereur) ON DELETE CASCADE
);

-- Table plainte
CREATE TABLE plainte (
   id_plainte INTEGER AUTO_INCREMENT PRIMARY KEY,
   contenu TEXT NOT NULL,
   date_plainte DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   statut TEXT NOT NULL CHECK (statut IN ('NOUVELLE',  'LU')),
   id_acquereur INTEGER NOT NULL,
   FOREIGN KEY (id_acquereur) REFERENCES acquereur(id_acquereur) ON DELETE CASCADE
);

-- utilisateur
INSERT INTO utilisateur (nom, email, mot_de_passe, role, num_phone, format_num_pays,
photo_profil)
VALUES
('Alice Mbarga', 'alice@example.com', 'pass123', 'ACQUEREUR', 690112233, 237, NULL),
('Bruno Nji', 'bruno@example.com', 'pass456', 'ANNONCEUR', 677445566, 237, NULL),
('Clara Douala', 'clara@example.com', 'adminpass', 'ADMIN', 699998877, 237, NULL);

-- acquereur
INSERT INTO acquereur (id_acquereur, historique_recherches)
VALUES
```

```
(1, JSON_ARRAY('studio', 'Douala')),
(2, JSON_ARRAY('appartement', 'Yaoundé')),
(3, JSON_ARRAY('villa', 'Bafoussam'));

-- annonceur
INSERT INTO annonceur (id_annonceur, numero_mtn_momo, numero_orange_money,
numero_uba)
VALUES
(2, 675000001, 690000002, 123456789),
(1, 699000003, 678000004, 987654321),
(3, 670000005, 673000006, 111222333);

-- administrateur
INSERT INTO administrateur (id_admin, numero_mtn_momo, numero_orange_money,
numero_uba)
VALUES
(3, 699000010, 690000020, 456123789),
(1, 677000030, 688000040, 123789456),
(2, 676000050, 689000060, 321654987);

-- annonce
INSERT INTO annonce (titre, description, prix_mensuel, localisation, status,
tarif_reservation, id_annonceur)
VALUES
('Studio à Bonamoussadi', 'Studio moderne et bien équipé', 100000, 'Douala', 'DISPONIBLE',
10000, 2),
('Appartement à Bastos', 'Grand appartement 3 chambres', 250000, 'Yaoundé', 'RESERVÉ',
15000, 2),
('Chambre à Makepe', 'Petite chambre pas chère', 50000, 'Douala', 'VENDU', 5000, 2);


-- contrat
INSERT INTO contrat (type, date_debut_contrat, date_fin_contrat, precision_en_surplus,
decision_acquereur, id_annonce)
VALUES
('VENTE', NOW(), DATE_ADD(NOW(), INTERVAL 6 MONTH), 'Paiement complet', TRUE, 1),
('PROMESSE_VENTE', NOW(), DATE_ADD(NOW(), INTERVAL 3 MONTH), 'Acompte versé',
FALSE, 2),
('VENTE', NOW(), DATE_ADD(NOW(), INTERVAL 12 MONTH), 'Contrat notarié', TRUE, 3);

-- signataire
INSERT INTO signataire (id_contrat, id_utilisateur, date_signature)
VALUES
(1, 1, NOW()),
(2, 2, NOW()),
```

```
(3, 1, NOW());

-- notification
INSERT INTO notification (type, lu, id_expediteur, id_destinataire)
VALUES
('PAIEMENT', FALSE, 2, 1),
('VISITE', TRUE, 3, 1),
('CONTRAT', FALSE, 1, 2);

-- message
INSERT INTO message (contenu, lu, id_expediteur, id_destinataire)
VALUES
('Bonjour, je suis intéressé par votre annonce.', FALSE, 1, 2),
('Merci pour votre message.', TRUE, 2, 1),
('Quand puis-je visiter ?', FALSE, 1, 2);

-- favoris
INSERT INTO favoris (id_acquereur, id_annonce)
VALUES
(1, 1),
(1, 2),
(1, 3);

-- visite
INSERT INTO visite (date_visite, statut, id_annonce, id_acquereur)
VALUES
(CURDATE(), 'DEMANDEE', 1, 1),
(CURDATE(), 'CONFIRMEE', 2, 1),
(CURDATE(), 'ANNULEE', 3, 1);

-- plainte
INSERT INTO plainte (contenu, statut, id_acquereur)
VALUES
('Le vendeur ne répond pas.', 'NOUVELLE', 1),
('Annonce frauduleuse.', 'LU', 1),
('Le bien est déjà vendu mais encore affiché.', 'NOUVELLE', 1);
```

🔧 Structure de base des API REST à créer

◆ 1. /api/utilisateurs

Gérer l'inscription, la consultation des profils, la connexion simplifiée (pas d'authentification réelle).

Endpoints :

- `POST /api/utilisateurs` → Créer un utilisateur (acquéreur par défaut)
- `GET /api/utilisateurs/{id}` → Récupérer un utilisateur
- `PUT /api/utilisateurs/{id}` → Modifier son profil
- `DELETE /api/utilisateurs/{id}` → Supprimer son compte
- POST /api/utilisateurs/login → Connexion simple (email/password)

◆ 2. /api/annonceurs

Pour qu'un acquéreur devienne annonceur.

Endpoints :

- `POST /api/annonceurs` → Convertir un utilisateur en annonceur (simulateur de "paiement")
- `GET /api/annonceurs/{id}/annonces` → Voir toutes ses annonces

◆ 3. /api/annonces

Créer, voir, modifier ou supprimer les annonces immobilières.

Endpoints :

- `POST /api/annonces` → Créer une annonce
- `GET /api/annonces` → Voir toutes les annonces
- `GET /api/annonces/{id}` → Détails d'une annonce
- `PUT /api/annonces/{id}` → Modifier une annonce
- `DELETE /api/annonces/{id}` → Supprimer une annonce
- `PUT /api/annonces/{id}/statut` : Changer le statut d'une annonce (disponible/réservé/vendu)

◆ 4. /api/reservations

Permettre à un acquéreur de réserver ou acheter un bien.

Endpoints :

- `POST /api/reservations` → Créer une demande de réservation/achat
- `GET /api/reservations/{id}` → Voir les détails d'une réservation
- `PUT /api/reservations/{id}` → Valider ou refuser (simulé par admin)

◆ 5. /api/paiements

Gérer les paiements fictifs entre acquéreur et annonceur (pas de vraie transaction).

Endpoints :

- POST /api/acquereur/{id}/paiements → Effectuer un paiement avec capture de preuve
- `GET /api/paiements/annonceur/{id}` → Paiements reçus
- `GET /api/paiements/acquereur/{id}` → Paiements effectués
- `PUT /api/paiements/{id}/valider` → Validation manuelle par l'admin

◆ 6. /api/contrats

Créer, lire, valider ou refuser les contrats associés à un achat.

Endpoints :

- `POST /api/contrats` → Générer un contrat pour une annonce
- `PUT /api/contrats/{id}/valider` → Le valider
- `PUT /api/contrats/{id}/refuser` → Le refuser
- GET /api/acquereur/{id}/contrats → Récupérer tous les contrats de l'acquéreur
- GET /api/acquereur/{id}/contrats/non-signes → Récupérer les contrats non signés (en attente)

- GET /api/acquereur/{id}/contrats/{idContrat} → Récupérer les détails d'un contrat spécifique

---

◆ 7. /api/messages

Système simple de messagerie entre utilisateurs.

Endpoints :

- `POST /api/messages` → Envoyer un message
- `GET /api/messages/conversation/{id1}/{id2}` : Récupérer une conversation entre deux utilisateurs
- `GET /api/messages/utilisateur/{id}` : Récupérer tous les messages d'un utilisateur
- `GET /api/messages/{destinataireId}` → Voir les messages reçus

◆ 8. /api/notifications

Notifier l'utilisateur d'un événement important.

Endpoints :

- `POST /api/notifications` → Créer une notification
- `GET /api/notifications/{utilisateurId}` → Voir ses notifications
- `PUT /api/notifications/{id}/lu` : Marquer une notification comme lue

◆ 9. /api/favoris

- `POST /api/favoris` → Ajouter une annonce aux favoris
- `GET /api/favoris/{acquereurId}` → Voir ses annonces en favoris
- `DELETE /api/favoris/{idAcquereur}/{idAnnonce}` : Supprimer des favoris

◆ 10. /api/Photos

- `POST /api/photos` : Ajouter une photo à une annonce
- `DELETE /api/photos/{id}` : Supprimer une photo

◆ 11. /api/Visites

- `POST /api/visites` : Demander une visite
- `GET /api/visites/annonce/{id}` : Voir les visites pour une annonce
- `GET /api/visites/acquereur/{id}` : Voir les visites demandées par un acquéreur
- `PUT /api/visites/{id}/statut` : Modifier le statut d'une visite (demandée/confirmée/annulée)

◆ 12. /api/Admin

- `GET /api/admin/annonces/validation` : Lister les annonces en attente de validation
- `PUT /api/admin/annonces/{id}/validation` : Valider ou refuser une annonce
- `GET /api/admin/utilisateurs` : Lister tous les utilisateurs
- PUT /api/admin/paiements/{id}/valider → Valider un paiement
- GET /api/admin/paiements → Tous les paiements en attente
- GET /api/admin/utilisateurs → Tous les utilisateurs

ARBORESCENCE DE L'API

```
reseau-social-immobilier/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/
│   │   │       └── reseauimmobilier/
│   │   │           ├── ReseauImmobilierApplication.java
│   │   │           ├── model/
│   │   │           │   ├── Utilisateur.java
│   │   │           │   ├── Acquereur.java
│   │   │           │   ├── Annonceur.java
│   │   │           │   ├── Administrateur.java
│   │   │           │   ├── Annonce.java
│   │   │           │   ├── Photo.java
│   │   │           │   ├── Contrat.java
│   │   │           │   ├── Paiement.java
│   │   │           │   ├── Notification.java
│   │   │           │   ├── Message.java
│   │   │           │   ├── Favoris.java
│   │   │           │   ├── Visite.java
│   │   │           │   └── Plainte.java
│   │   │           ├── repository/
│   │   │           │   ├── UtilisateurRepository.java
│   │   │           │   ├── AcquereurRepository.java
│   │   │           │   ├── AnnonceurRepository.java
│   │   │           │   ├── AdministrateurRepository.java
│   │   │           │   ├── AnnonceRepository.java
│   │   │           │   ├── PhotoRepository.java
│   │   │           │   ├── ContratRepository.java
│   │   │           │   ├── PaiementRepository.java
│   │   │           │   ├── NotificationRepository.java
│   │   │           │   ├── MessageRepository.java
│   │   │           │   ├── FavorisRepository.java
│   │   │           │   ├── VisiteRepository.java
│   │   │           │   └── PlainteRepository.java
│   │   │           ├── service/
```

```
│   │   │       │   ├── UtilisateurService.java
│   │   │       │   ├── AnnonceurService.java
│   │   │       │   ├── AnnonceService.java
│   │   │       │   ├── PaiementService.java
│   │   │       │   ├── ContratService.java
│   │   │       │   ├── MessageService.java
│   │   │       │   ├── NotificationService.java
│   │   │       │   ├── FavorisService.java
│   │   │       │   ├── VisiteService.java
│   │   │       │   └── AdminService.java
│   │   │       └── controller/
│   │   │           ├── UtilisateurController.java
│   │   │           ├── AnnonceurController.java
│   │   │           ├── AcquereurController.java
│   │   │           ├── AnnonceController.java
│   │   │           ├── PaiementController.java
│   │   │           ├── ContratController.java
│   │   │           ├── MessageController.java
│   │   │           ├── NotificationController.java
│   │   │           ├── FavorisController.java
│   │   │           ├── PhotoController.java
│   │   │           ├── VisiteController.java
│   │   │           └── AdminController.java
│   │   └── resources/
│   │       ├── application.properties
│   │       └── static/
│   └── test/
│       └── java/
│           └── com/
│               └── reseauimmobilier/
│                   ├── ReseauImmobilierApplicationTests.java
│                   └── controller/
│                       ├── UtilisateurControllerTest.java
│                       ├── AnnonceControllerTest.java
│                       └── PaiementControllerTest.java
├── pom.xml
└── README.md
```