@ Quatissimo

1. Fonctionnalités de l'application

1.1 Troc entre voisins

- Les utilisateurs peuvent publier des offres/demandes de troc (par exemple, échange de livres, d'équipements, etc.).
- Possibilité d'échanger des messages privés pour finaliser l'échange.

1.2 Services

- Les voisins peuvent proposer ou solliciter des services (récupération de colis, sortie de chien, courses pour une personne âgée, etc.).
- Système de disponibilité et de "planning" (possibilité de consulter les prochaines dates/horaires où un utilisateur est disponible).

1.3 Sorties / Activités

- Organisation d'une sortie (cinéma, restaurant, balade, etc.) avec un nombre maximum de participants.
- Les utilisateurs doivent s'inscrire. L'activité est annulée si le quota minimum n'est pas atteint (ou si le maximum est dépassé).

1.4 Surveillance en cas d'absence

- Les voisins peuvent déclarer leurs dates d'absence pour demander une surveillance de leur logement.
- Possibilité de s'inscrire comme "contact de confiance" pour un voisin.

1.5 Messagerie (incluant vidéo si possible)

- Discussions textuelles individuelles ou de groupe.
- Indication de l'état "connecté/hors-ligne" pour chaque utilisateur.
- (Optionnel) Intégration WebRTC pour la visioconférence (complexité plus élevée).

1.6 Organisation d'événements communautaires

- Nettoyage, collecte de déchets, fêtes de quartier, etc.
- Gestion d'une page/section listant les prochains événements, avec possibilité de s'inscrire.

1.7 Journal de quartier

- Pages d'articles, news ou annonces stockées dans MongoDB.
- Accès en lecture (public ou restreint) et possibilité de contribution (rédiger un article).

1.8 Mémorisation des interactions

- Exemples :
  - A a sorti le chien de B.
  - B est allé au cinéma avec C.
- Objectif : constituer un historique pour générer des suggestions de rencontres ou de participation (un moteur de suggestion peut recommander à A de proposer un autre service à B, etc.).

1.9 Jeux (optionnel, selon le temps)

- Exemples : quiz communautaire, petits jeux "casual" en ligne entre voisins.
- Vise à promouvoir l'interaction et la convivialité.

---

2. Architecture globale

2.1 Back-end (NodeJS + BDD)

- NodeJS (API REST).
- SGBD relationnel (PostgreSQL).
- MongoDB pour les pages du journal de quartier ou l'historique des interactions.
- JWT pour sécuriser les routes.

2.2 Front-end Web (React)

- Deux interfaces :
  1. **Interface utilisateur** : accessible à tous les habitants (pages troc, messagerie, services, etc.).
  2. **Back-office** : gestion avancée (administration, modération des contenus, statistiques d'usage).
- Messagerie en temps réel :
  - WebSocket (via Socket.io côté back NodeJS) pour la messagerie instantanée et le statut en ligne/hors-ligne.
  - Affichage d'une liste de contacts, avec indication de leur statut.
- Design :
  - Responsive (Bootstrap).
  - Systèmes de thèmes (clairs/sombres).

2.3 Application Desktop Java (JavaFX + WebScraping)

- **Objectif** : Récolter des données (événements locaux, actualités de la mairie, etc.) depuis d'autres sites web pour enrichir le journal ou proposer des suggestions.
- **Technologies** :
  - JavaFX pour l'interface.
  - Bibliothèque JSoup pour parser le HTML.
- **Interface** :
  - Configuration des sites à scraper (URL).
  - Configuration des catégories d'informations à extraire (annonces, articles, etc.).
  - Affichage des données extraites, logs d'exécution, etc.
- **Mécanismes avancés** :
  - Système de mise à jour du logiciel : vérification sur un serveur pour récupérer une nouvelle version du JAR.
  - Système de thèmes : JavaFX gère différents fichiers CSS pour changer l'apparence.
  - Mode Online/Offline : en cas de coupure internet, stocker les données localement et synchroniser plus tard.
  - Système de plugins : chargement dynamique de JAR pour ajouter de nouvelles fonctions (export PDF, etc.).

2.4 Langage d'interrogation (lex & yacc)

- **Objectif** : Concevoir un mini-langage de type SQL en Python pour interroger les documents (articles du journal, historique des interactions dans MongoDB, etc.).
- **Exemple de syntaxe** :

```
sql
CopierModifier
SELECT titre, auteur FROM Articles WHERE categorie = 'Evenement';
```

- **Intégration** :
  - Soit dans l'API NodeJS (via un module natif ou un wrapper).
  - Soit dans l'application Java.
- **Utilisation** :
  - L'utilisateur (ou l'admin) peut taper la requête pour récupérer des résultats.

---

3. Moteur de suggestion (interactions entre voisins)

- À chaque fois qu'un voisin participe à une activité avec un autre ou rend un service, on enregistre l'information dans la collection **Interactions**.
- **Idées d'algorithmes** :
  - **Filtrage collaboratif** : recommander à A de se rapprocher de B si B a des centres d'intérêts similaires, ou a déjà interagi avec A via un tiers.

    - **Approche basée sur des règles** : si un utilisateur a rendu service 3 fois à un voisin, on suggère de participer ensemble à un événement.
    - **Scoring** : calculer un score d'affinité entre voisins selon le nombre et le type d'interactions.
- **Exemple d'utilisation** :
    - Lorsqu'un utilisateur se connecte, le moteur calcule une liste de suggestions (amis potentiels, événements susceptibles de l'intéresser, objets en troc correspondant à ses préférences, etc.).

---

4. Conteneurisation et déploiement

- Création des **Dockerfile** et **docker-compose.yml** pour le projet.
- Permet de déployer l'ensemble de l'application (back-end, front-end, base de données, etc.) de façon cohérente.

GIT-HUB LINK : https://github.com/WalidBlhr/AnnualProject