UNIVERSITE DE YAOUNDÉ I

ÉCOLE NATIONALE SUPÉRIEURE POLYTECHNIQUE DE YAOUNDÉ

DÉPARTEMENT DES TELECOMMUNICATIONS

THE UNIVERSITY OF YAOUNDE I

NATIONAL ADVANCED SCHOOL OF ENGINEERING OF YAOUNDE

DEPARTMENT OF TELECOMMUNICATION

GENIE LOGICIEL

RESPONSABLE : MR. MBIETIEU

GROUPE 5

CONCEPTION ET RÉALISATION D'UN LOGICIEL DE GESTION DE TONTINE

MEMBRES

NOMS	PRENOMS	MATRICULE	Participation(%)
NLEMENYEME	ISMENE	20P177	20
DJOMGUEM DJOUYOU	CHRISTOPHE JUNIOR	22P515	20
MBANGAH	NDOH	22P560	20
BIKELE	FRITZ-GERALD	21P151	20
FOUMEGNI	LOIC	22P200	20

SOMMAIRE

I. CAHIER DES CHARGES : CRÉATION D'UN LOGICIEL DE GESTION DES TONTINES

1. **Informations Générales**

- **Date de début du projet** : 24/02/2025

-**Date de fin du projet** : 29/04/2025

- **Projet** : Développement d’un logiciel de gestion des tontines pour une association.

- **Client** : Mr AMOS

- **Développeur** : groupe 4

2. **Présentation du Projet**

Le projet vise à créer un logiciel de gestion des tontines pour une association afin d’automatiser les processus liés à l’organisation, le suivi et la gestion des activités liées aux tontines. Ce logiciel permettra de :

- Créer et gérer les tontines (définir les montants, la périodicité, les membres participants).

- Automatiser les redistributions financières selon des règles prédéfinies.

- Suivre les cotisations des membres (montants payés ou en attente).

- Gérer les paiements en retard.

- Assurer un suivi des statistiques financières grâce à un tableau de bord interactif.

Le logiciel doit être sécurisé, ergonomique, accessible en ligne via un navigateur web et compatible avec les appareils mobiles.

3. **Objectifs du Projet**

- Automatiser la gestion des tontines.

- Faciliter le suivi des cotisations et redistributions financières.

- Garantir la transparence et la sécurité des données financières.

- Permettre aux membres de consulter leurs informations personnelles (solde, historique de participation).

- Fournir des outils de reporting financiers détaillés pour l'administrateur.

4. **Périmètre Fonctionnel**

Le logiciel doit couvrir les fonctionnalités suivantes :

4.1. **Gestion des Utilisateurs**

- Rôles des utilisateurs :

  - Administrateur : Gère tous les aspects du logiciel (création de tontines, gestion des membres, suivi des finances, rapports, etc.).

  - Membre : Peut consulter son solde, participer à une tontine et effectuer des paiements.

- Comptable : Enregistre les transactions financières (cotisations, redistributions, etc.).

- Fonctionnalités :

  - Inscription et authentification des membres.

  - Gestion des rôles et permissions (Administrateur, Membre, Comptable).

  - Modification du profil utilisateur (nom, email, mot de passe).

  - Réinitialisation sécurisée du mot de passe.

4.2. **Gestion des Membres**

- Ajouter, modifier, et supprimer un membre.

- Afficher la liste des membres avec leur solde financier.

- Filtrer les membres par statut (actifs, inactifs, en retard de paiement).

- Générer un rapport des membres avec leurs contributions financières.

4.3. **Gestion des Tontines**

- Création de tontines :

  - Définir les montants des cotisations.

  - Définir la périodicité (hebdomadaire, mensuelle, etc.).

  - Inscrire les membres participants.

- Automatisation :

  - Générer automatiquement les redistributions des fonds selon des règles prédéfinies.

- Suivi :

- Afficher l'état des tontines (montants collectés, redistributions effectuées, montants restants).

- Générer des reçus pour chaque cotisation.

4.4. **Gestion des Cotisations**

- Enregistrer les paiements effectués par les membres.

- Suivre les cotisations en attente ou en retard.

- Générer une attestation de cotisation pour chaque membre.

- Offrir plusieurs modes de paiement : espèces, Mobile Money, virement bancaire.

4.5**. Gestion des Retards de Paiement**

- Identifier les membres en retard de cotisation.

- Envoyer des rappels automatiques (Email, SMS) aux membres en retard.

- Planifier des actions de recouvrement selon les décisions de l'association.

4.6. **Tableau de Bord**

- Afficher les statistiques financières en temps réel.

- Afficher des graphiques sur les flux financiers (cotisations, redistributions, paiements en retard, etc.).

- Exporter les rapports financiers au format PDF ou Excel.

5. **Contraintes et Exigences**

5.1. **Contraintes Fonctionnelles**

- Le logiciel doit fonctionner en mode web (accessible via un navigateur).

- Le système doit permettre la gestion simultanée de plusieurs tontines.

- L'interface doit être conviviale et adaptée à tous les profils d'utilisateurs.

5.2. **Contraintes Techniques**

- **Backend** : Python (framework Django) avec Django REST Framework (DRF) pour l'API.

- **Frontend** : Django Templates ou ReactJS pour une interface utilisateur interactive.

- **Base de données** : MySQL pour gérer les données des membres, tontines, transactions, etc.

- **Authentification** : Utilisation de JWT (JSON Web Tokens) ou OAuth2 pour sécuriser les accès.

- **Hébergement** : Serveur cloud (AWS, DigitalOcean, OVH) ou un serveur local selon les besoins de l'association. (Optionnel)

5.3. **Exigences de Sécurité**

- Authentification sécurisée (mot de passe haché, gestion des sessions).

- Protection contre les attaques courantes (injections SQL, XSS, CSRF).

- Chiffrement des données sensibles (exemple : informations de connexion).

- Sauvegarde régulière des données pour éviter les pertes.

5.4. **Exigences de Performance**

- Optimisation des requêtes SQL pour garantir la rapidité d'exécution.

- Support de plusieurs tontines simultanées sans ralentissement.

- L'application doit être scalable pour prendre en charge une augmentation du nombre d'utilisateurs.

5.5. **Exigences d'Accessibilité**

- Interface utilisateur responsive (compatible avec les écrans d'ordinateurs, tablettes, et smartphones).

- Compatible avec les navigateurs modernes (Chrome, Firefox, Edge).

6. **Critères de Réussite**

- Le logiciel est fonctionnel et répond aux besoins de l'association.

- Les utilisateurs (Administrateur, Membres, Comptables) peuvent facilement utiliser le système.

- Le système garantit la sécurité et la transparence des transactions financières.

Ce logiciel vise à moderniser la gestion des tontines en automatisant les tâches administratives et financières, tout en assurant un suivi efficace des cotisations et redistributions. Une fois déployé, il permettra à l'association de gagner en efficacité et en transparence.

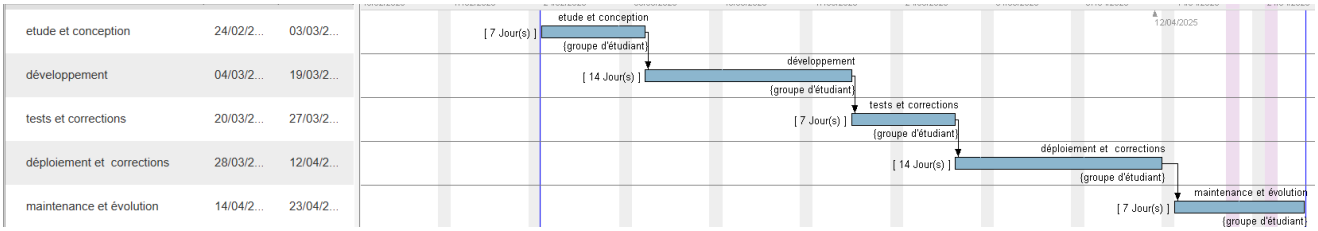
7.**plannification de GANTT**

Un diagramme de Gantt est un outil de gestion de projet qui permet de visualiser la planification et l'avancement des tâches dans le temps. Ce type de diagramme utilise des barres horizontales pour représenter la durée des différentes tâches ou activités d'un projet, le long d'une ligne de temps.

Créer un diagramme de Gantt pour le développement d'un logiciel de gestion des tontines pour une association implique plusieurs étapes clés, divisées en tâches spécifiques :

Nom :	Date de début	Date de fin :
Étude et conception	24/02/2025	03/03/2025
Développement	04/03/2025	19/03/2025
Tests et corrections	20/03/2025	27/03/2025
Déploiement et corrections	28/03/2025	12/04/2025
Maintenance et évolution	14/04/2025	23/04/2025

Figure 1.A



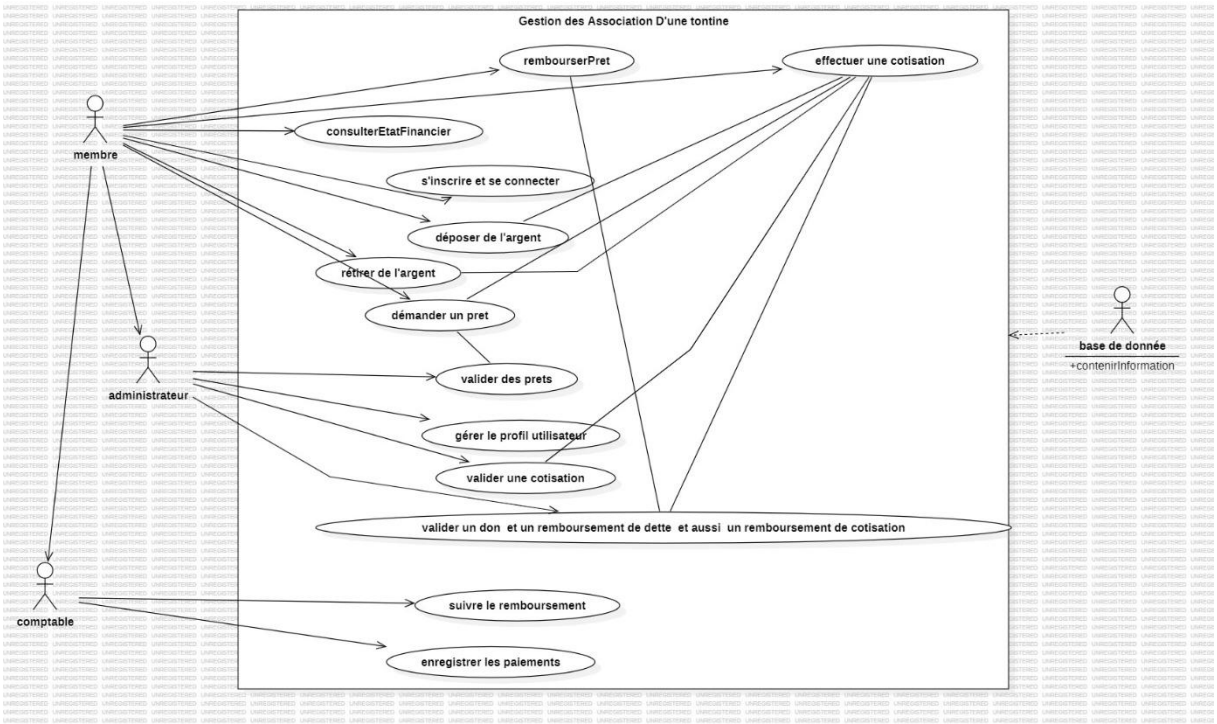
II. MODELISATION LOGICIEL

Voici une génération des diagrammes UML (cas d'utilisation, déploiement) pour le projet décrit dans le cahier des charges. Le projet consiste à développer une application de gestion d'association.

1. DIAGRAMME DE CAS D'UTILISATION

Le diagramme de cas d'utilisation illustre les interactions entre les différents acteurs (Membre, Administrateur, Comptable) et les fonctionnalités principales du système :

Figure1.B



Le diagramme de cas d'utilisation montre les différentes interactions entre les utilisateurs et les fonctionnalités d'une application de gestion des tontines. Les utilisateurs sont divisés en trois catégories : membre, administrateur et comptable. Chaque catégorie d'utilisateur a des actions spécifiques qu'elle peut effectuer.

Pour les **membres**, les actions incluent :

Rembourser un prêt

Consulter l'état financier

Déposer de l'épargne

Gérer le profil utilisateur

Retirer de l'épargne

Demander un prêt

S'inscrire et se connecter

Pour les **administrateurs**, les actions incluent :

Gérer le profil utilisateur

Valider les prêts

S'inscrire et se connecter

Pour les **comptables**, les actions incluent :

Enregistrer les paiements

Suivre les remboursements

S'inscrire et se connecter

Ce diagramme montre clairement les différentes responsabilités et actions possibles pour chaque type d'utilisateur dans le système de gestion des tontines.

2. DIAGRAMME DE CAS SEQUENCE

Un **diagramme de séquence** est un type de diagramme UML (Unified Modeling Language) qui représente les interactions dynamiques entre les objets ou les acteurs d'un système dans un ordre chronologique. Il illustre la manière dont les messages (ou appels) sont échangés entre différentes entités pour accomplir une tâche ou un cas d'utilisation spécifique.

- **Diagramme de séquence pour s'inscrire et se connecter**

Participants :

- Utilisateur

- Interface utilisateur (application mobile ou web)

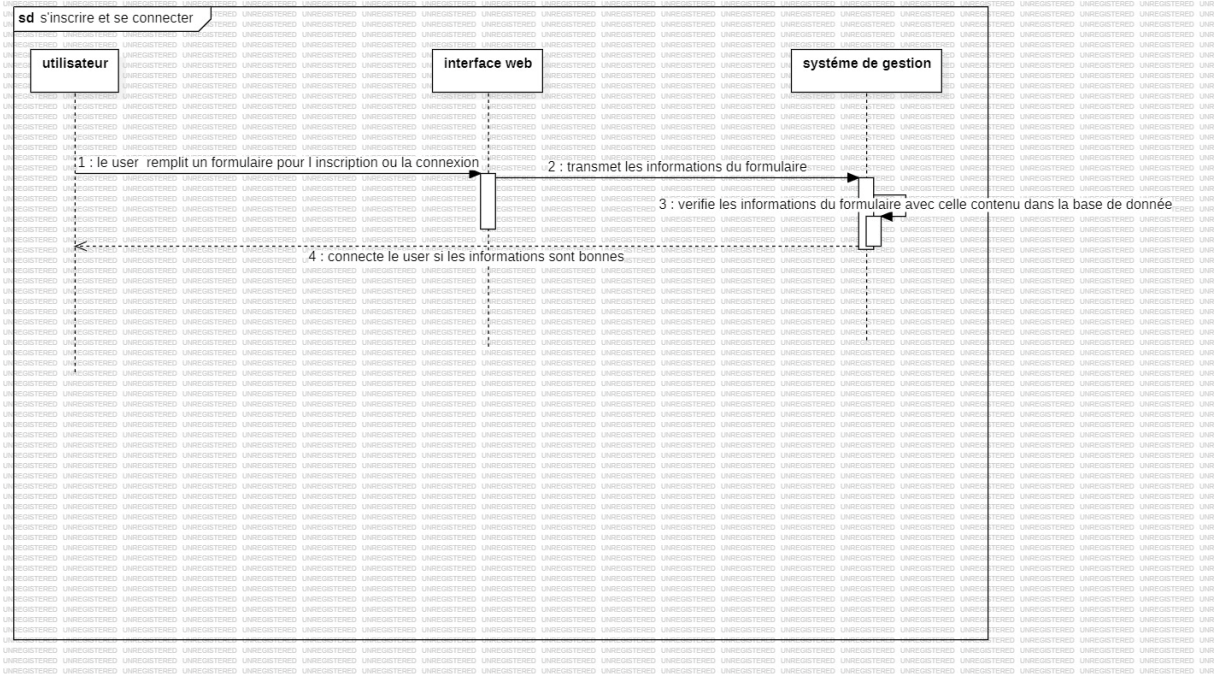
- Système de gestion (backend)

- Base de données

Étapes :

- ✓ Pour se connecter, l'utilisateur entre ses identifiants.
- ✓ L'interface utilisateur envoie les identifiants au système de gestion qui les vérifie dans la base de données.
- ✓ Si les identifiants sont corrects, l'utilisateur est connecté. Sinon, un message d'erreur est renvoyé.

Figure1.C



- **Diagramme de séquence pour demander un prêt**

Description :

Ce diagramme illustre comment un utilisateur peut demander un prêt dans l'application.

Participants :

- Utilisateur

- Interface utilisateur

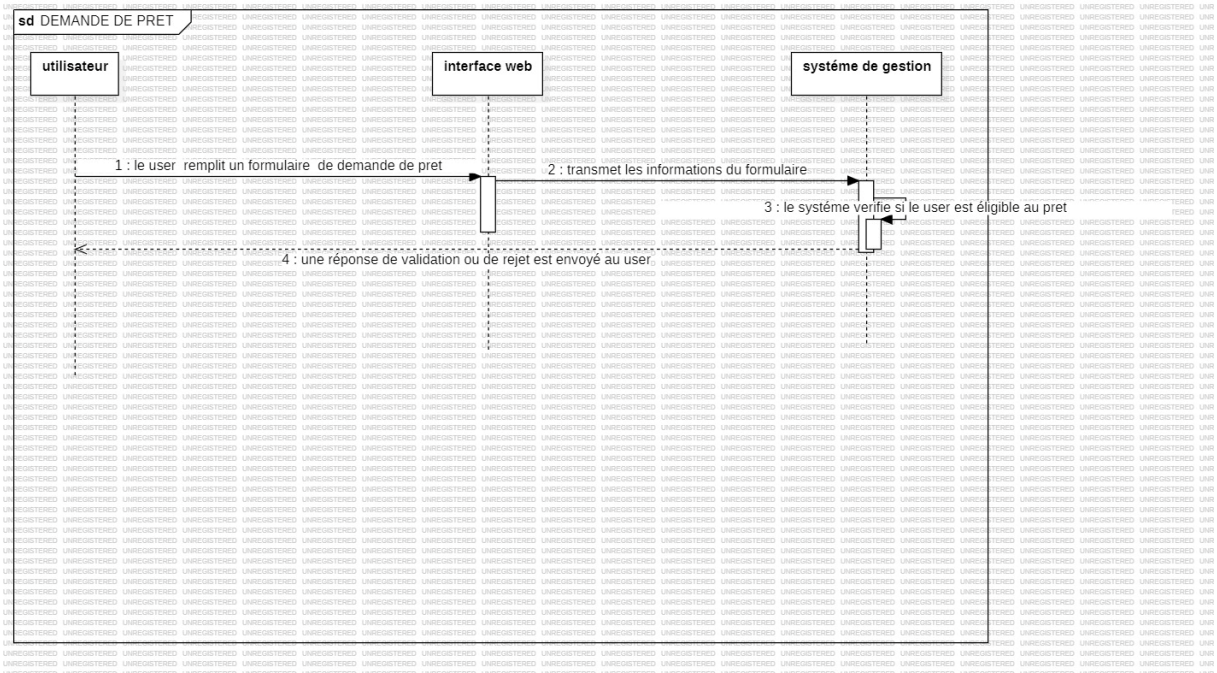
- Système de gestion (backend)

- Base de données

Étapes :

- ✓ L'utilisateur remplit un formulaire de demande de prêt (montant, durée, motif, etc.).
- ✓ L'interface utilisateur transmet ces informations au système de gestion.
- ✓ Le système vérifie si l'utilisateur est éligible pour un prêt (selon son historique ou son épargne).
- ✓ Si l'utilisateur est éligible, la demande est enregistrée dans la base de données.
- ✓ Une réponse (validation ou rejet) est envoyée à l'utilisateur

Figure2.C



- **Diagramme de séquence pour consulter l'état financier**

Description :

Ce diagramme montre comment un utilisateur consulte son état financier (solde d'épargne, prêts en cours, contributions, etc.).

Participants :

- Utilisateur

- Interface utilisateur

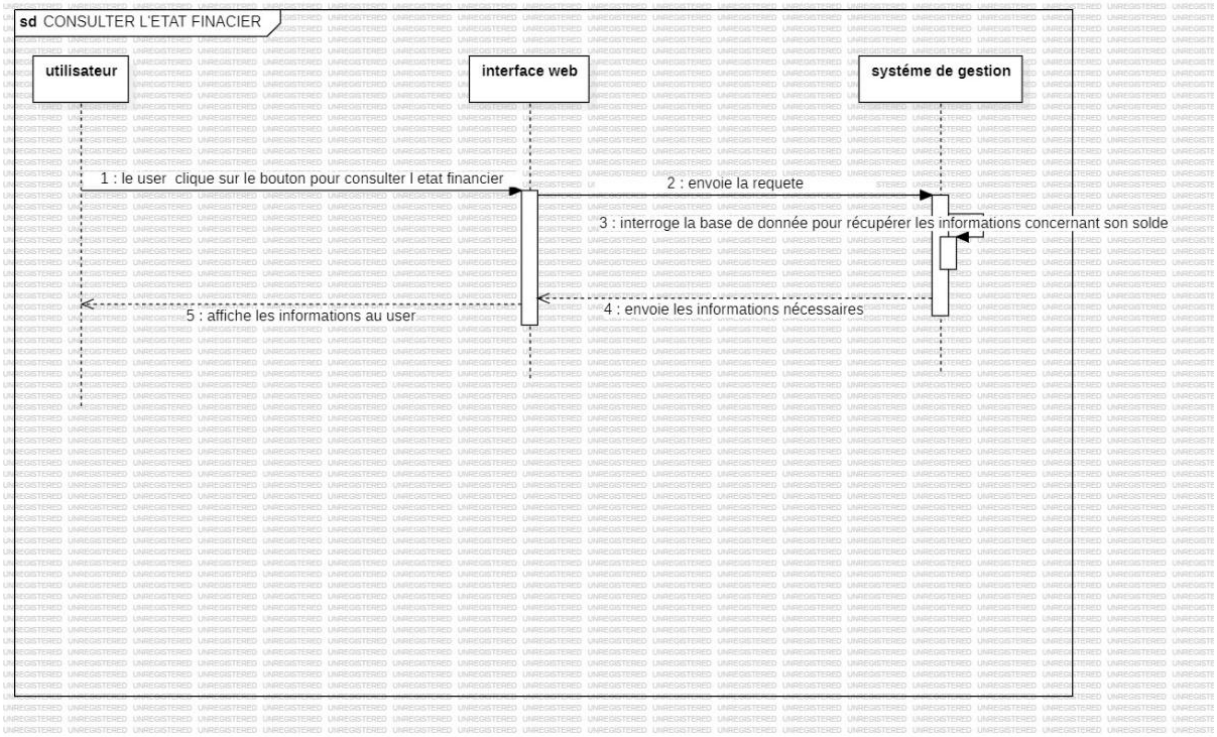
- Système de gestion (backend)

- Base de données

Étapes :

- ✓ L'utilisateur clique sur une option pour consulter ses finances.
- ✓ L'interface utilisateur envoie une requête au système de gestion.
- ✓ Le système interroge la base de données pour récupérer les informations pertinentes (solde, historique des transactions, prêts, etc.).
- ✓ Les données récupérées sont renvoyées à l'interface utilisateur.
- ✓ L'interface utilisateur affiche les informations à l'utilisateur.

Figure3.C



- **Diagramme de séquence pour déposer de l'argent**

Description :

Ce diagramme explique comment un utilisateur effectue un dépôt d’épargne dans l’application.

Participants :

- Utilisateur

- Interface utilisateur

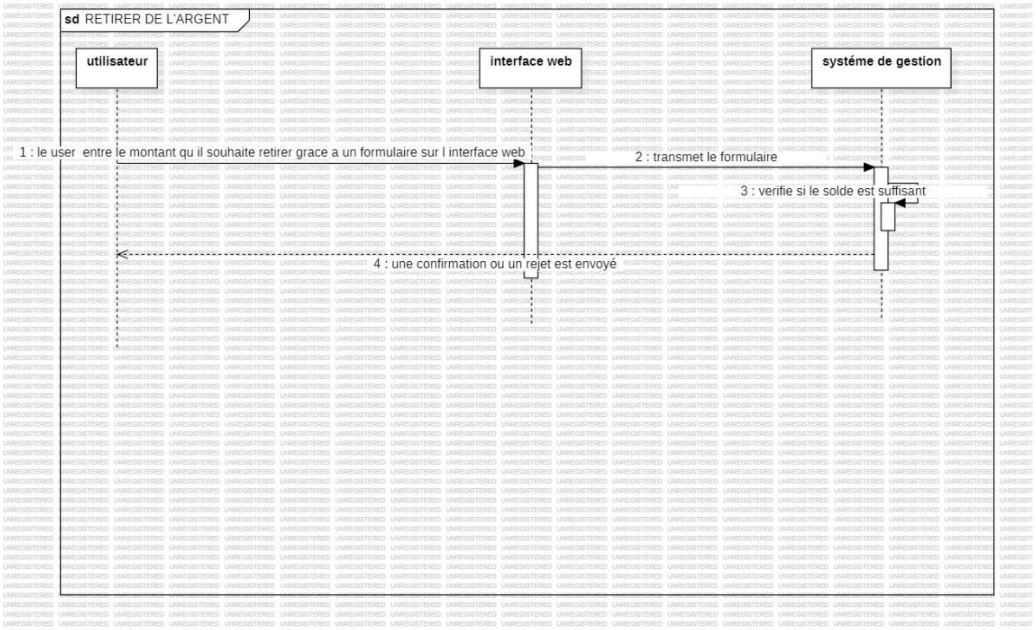
- Système de gestion (backend)

-Base de données

Étapes :

- ✓ L'utilisateur entre le montant qu'il souhaite déposer et choisit un mode de paiement (virement, carte bancaire, etc.).
- ✓ L'interface utilisateur envoie ces informations au système de gestion.
- ✓ Le système vérifie la validité de l'opération (montant, méthode de paiement, etc.).
- ✓ Si tout est valide, le dépôt est enregistré dans la base de données.
- ✓ Une confirmation de dépôt est envoyée à l'utilisateur.

Figure4.C



- **Diagramme de séquence pour retirer de l'argent**

Description :

Ce diagramme montre comment un utilisateur peut retirer de l'épargne depuis son compte.

Participants :

- Utilisateur

- Interface utilisateur

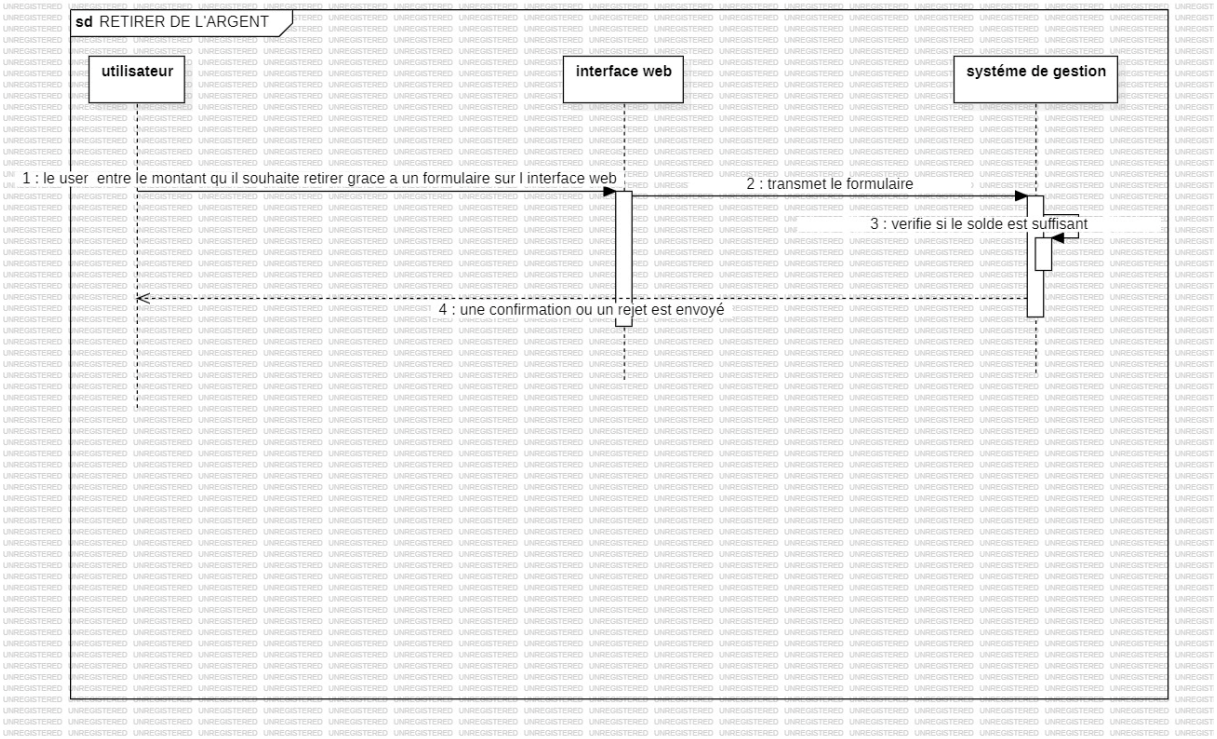
-Système de gestion (backend)

- Base de données

Étapes :

- ✓ L'utilisateur entre le montant qu'il souhaite retirer.
- ✓ L'interface utilisateur envoie la demande au système de gestion.
- ✓ Le système vérifie si l'utilisateur dispose d'un solde suffisant.
- ✓ Si le solde est suffisant, le système enregistre la transaction dans la base de données.
- ✓ Une confirmation est envoyée à l'utilisateur, et le retrait est effectué via le mode de paiement choisi (virement bancaire, etc.)

Figure5.C

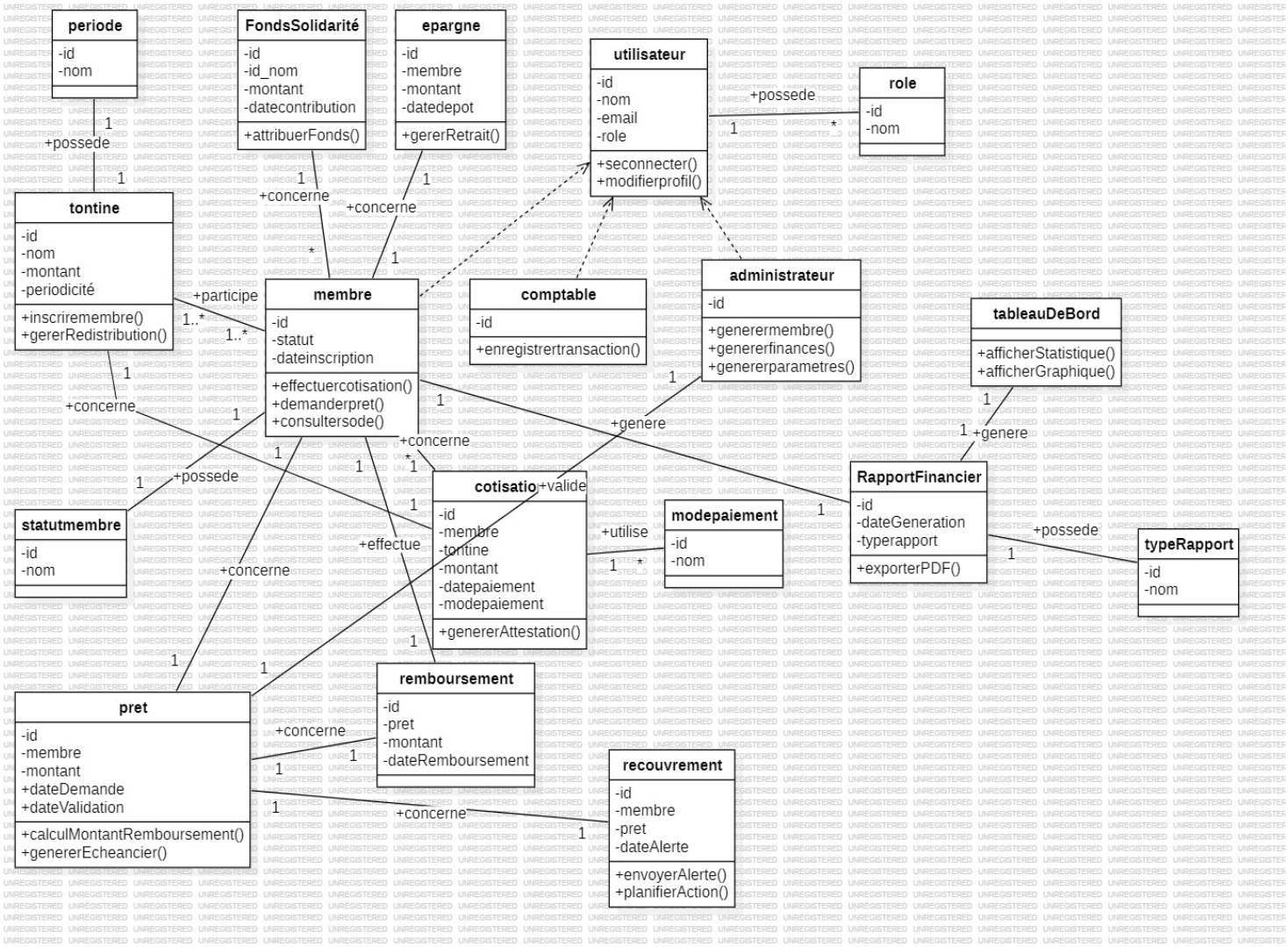


3.DIAGRAMME DE CLASSE

Le **diagramme de classe** est un type de diagramme UML (Unified Modeling Language) utilisé pour représenter la structure statique d'un système. Il décrit les classes, leurs attributs, leurs méthodes et les

relations entre elles. C'est un outil essentiel pour la modélisation d'applications logicielles, notamment dans les phases de conception.

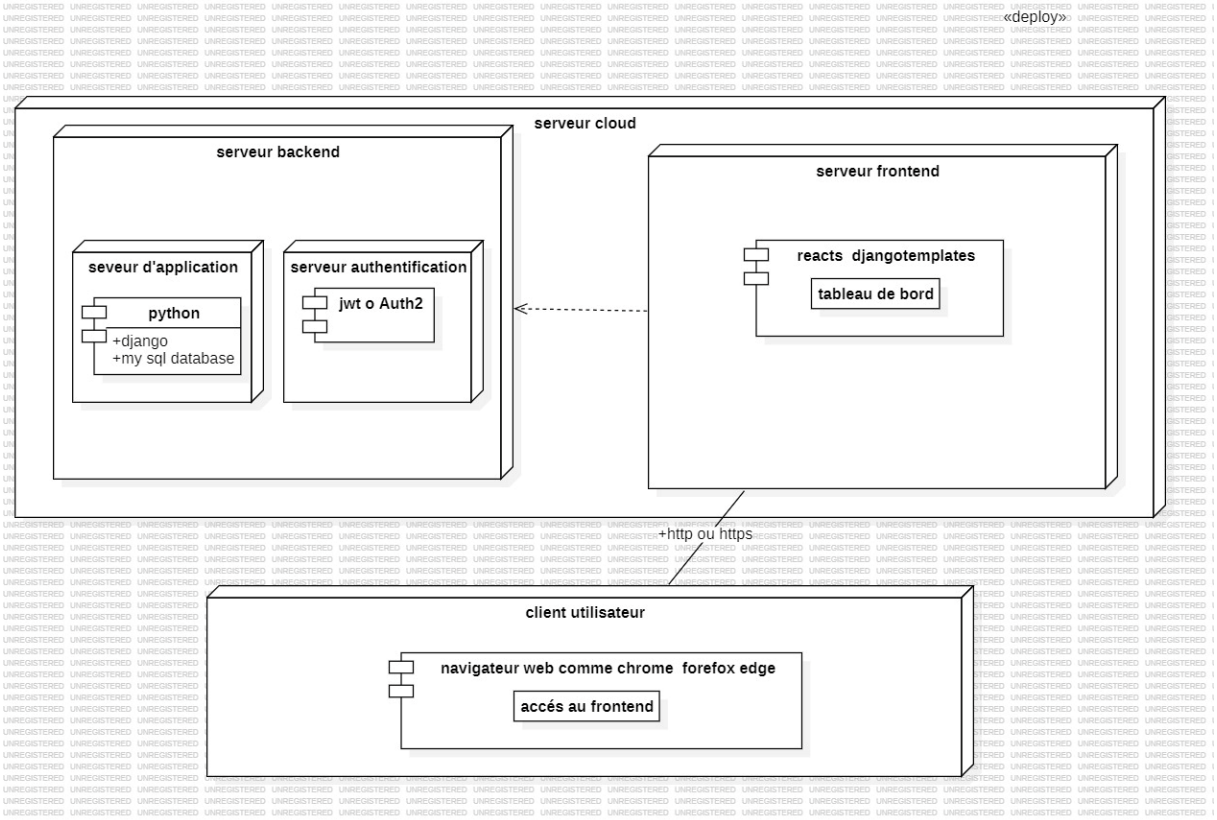
Figure 1.D



4.Diagramme de déploiement

Le diagramme de déploiement décrit les relations entre les Composants matériels et logiciels de l'application ainsi que leur Déploiement. Voici un exemple basé sur les Technologies spécifiées :

Figure1.E



• Client Utilisateur :

- Les membres, administrateurs et comptables accèdent à l'application Via un navigateur web (Google Chrome, Firefox, Edge, etc.).

- Ils interagissent avec l'interface utilisateur (Frontend) pour consulter les Informations, effectuer des actions (cotisations, prêts, etc.), ou accéder à Leurs tableaux de bord.

• Serveur Frontend :

- Il gère l'affichage et les interactions utilisateur.

- Développé avec ReactJS (ou Django Templates).

- Communique avec le backend via des appels HTTP/HTTPS à l'API REST.

• **Serveur Backend :**

- Le cœur de l'application, développé en Django avec le Django REST Framework (DRF).

- Gère la logique métier (gestion des membres, tontines, prêts, etc.).

- Stocke les données dans une base de données MySQL.

- Assure une authentification sécurisée avec JWT

• **Base de données (MySQL) :**

- Gère le stockage des données des membres, transactions, tontines, prêts, etc.

• **Cloud :**

- L'application est hébergée sur un serveur cloud (AWS, DigitalOcean, OVH, etc.).

- Permet une scalabilité et un accès en temps réel.

Résumé des relations

- Client Utilisateur : Accède au Frontend via un navigateur web.

- Frontend : Communique avec le Backend via des appels REST API (HTTP/HTTPS).

- Backend : Gère les données et la logique métier, et interagit avec la base de données pour persister les informations.

- Base de données : Stocke toutes les données de l'application

5.Choix de l'architecture logicielle (MVT)

Voici une analyse de l'architecture MVT (Modèle-Vue-Template) de notre application de tontine sur Django, basée sur les fichiers :

Composant	Correspondance dans notre projet	Rôle	Exemples concrets
M (Modèle)	Fichier **models.py**	Gère les données et la logique métier	Cotisation, Remboursement, Don, Pret, Membre
V (Vue)	Fichier **views.py**	Traite la logique et les requêtes HTTP	inscription(), creer_cotisation(), faire_don()
T (Template)	Dossier **templates/**	Affiche les données (UI)	acceuil.html, cotisations/liste.html, dons/faire.html

Bibliographie

Livres & Documents :

- ROQUES, Pascal, *UML 2 - Pratique de la modélisation (2e édition)*, Éditions Dunod, 2017.
  (Utilisé pour la génération des diagrammes UML)

Logiciels :

- GanttProject, *Logiciel de gestion de projet*, 2024. Disponible sur : https://www.ganttproject.biz
  (Utilisé pour la création du diagramme de Gantt,)