



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de
HONORIS UNITED UNIVERSITIES

PROJET DE dev mobile

Conception et Développement

Riminsta : clone d instagram

Réalisée par : JABAL Rim / MOUJTAHIDE Hamza/ HAJI Achraf

2025/2026

Table des matières

Introduction générale	4
Chapitre 1 : Présentation du cadre du projet	5
1.1 Introduction	5
1.2 Présentation du contexte du projet	5
1.3 Périmètre Fonctionnel	5 1.4
Contraintes Techniques	6 1.5
Architecture Technique	7 1.6
Environnement Technique	7 1.8
Planning prévisionnel	8 1.9
Choix du Modèle de Développement	8
Chapitre 2 : Spécification des besoins	12
2.1 Description des cas d'utilisation	12
Chapitre 3 : Conception du système	13
3.1 Modélisation dynamique	13
• 3.1.1 Diagrammes de séquences.....	13
• 3.1.2 Diagrammes d'états.....	17
• 3.1.3 Diagrammes d'activité	19
3.2 Modélisation statique	24
• 3.2.1 Diagramme de classes	25
• 3.2.2 Diagramme de cas d'utilisation global	26
Chapitre 3 : Réalisation du système	28 4.1
Environnement de développement	28 4.2
Technologies utilisées	28 4.3
Principales interfaces graphiques (Console)	28
Conclusion générale	31
Bibliographie et Nétographie	32

Introduction générale

Dans le cadre de notre formation en développement d'applications mobiles, nous avons réalisé un projet de réseau social nommé **Riminsta**.

Ce projet a pour objectif de mettre en pratique l'ensemble des concepts étudiés durant le cours, notamment :

- Le développement mobile cross-platform avec **React Native et Expo**
- La **Programmation Orientée Composants** avec les composants fonctionnels React
- L'utilisation de **TypeScript** pour le typage statique et la robustesse du code
- La **gestion d'état** avec Context API, useReducer et useState hooks
- L'intégration de **Firebase** (Authentication, Firestore Database, Cloud Storage)
- La **navigation** avec React Navigation (Stack et Tab Navigators)
- La **communication temps réel** avec les listeners Firestore (onSnapshot)
- La **validation de formulaires** avec Yup schemas
- L'**authentification OAuth** avec Google Sign-In
- La **gestion de version** avec Git et GitHub

L'application permet de gérer :

- **L'authentification des utilisateurs** avec email/mot de passe et Google OAuth
- **La publication de contenus** (posts avec images, stories éphémères, reels)
- **Les interactions sociales** (likes, commentaires, partages, sauvegarde)
- **Les profils utilisateurs** avec statistiques et personnalisation
- **Le système de follow/followers** pour créer des communautés
- **La messagerie instantanée** en temps réel entre utilisateurs
- **Les notifications automatiques** pour toutes les interactions
- **La recherche d'utilisateurs** et découverte de contenus

Ce rapport présente l'architecture du projet, les différents concepts implémentés, ainsi que les fonctionnalités développées.

Chapitre 1

Présentation du cadre de projet

Élément	Description
Nom du projet	Riminsta
Type	Application mobile cross-platform (iOS/Android)
Objectif	Réseau social de partage de photos et vidéos avec interactions temps réel

2. Contexte

Dans un monde où les réseaux sociaux sont devenus essentiels pour la communication et le partage d'expériences, nous avons souhaité développer une application mobile moderne permettant aux utilisateurs de créer, partager et interagir avec du contenu multimédia. L'application doit offrir une expérience utilisateur fluide, intuitive et performante, comparable aux standards actuels des applications de réseaux sociaux.

3. Périmètre Fonctionnel

3.1 Gestion de l'Authentification

- Inscription utilisateur (username, email, mot de passe)
- Connexion avec email/mot de passe
- Connexion avec Google OAuth
- Persistance de session avec AsyncStorage
- Déconnexion sécurisée

3.2 Gestion des Publications

- Créer un post avec image et caption
- Upload d'image vers Firebase Storage
- Afficher le fil d'actualité des posts
- Liker/Unliker un post (avec animation)
- Commenter un post
- Sauvegarder un post pour consultation ultérieure
- Supprimer ses propres posts

3.3 Gestion des Profils

- Afficher son profil avec statistiques (posts/followers/following)
- Modifier son profil (nom, username, bio, photo)
- Consulter le profil d'autres utilisateurs
- Grille de posts par utilisateur
- Onglet des posts sauvagardés

3.4 Interactions Sociales

- Suivre/Ne plus suivre un utilisateur
- Consulter la liste des followers
- Consulter la liste des following
- Système de notifications automatiques (likes, comments, follows)
- Badge de notifications non lues

3.5 Messagerie

- Liste des conversations avec aperçu du dernier message
- Chat en temps réel avec un utilisateur
- Envoi/réception de messages instantanés
- Indicateur de messages non lus
- Démarrer une nouvelle conversation

3.6 Stories et Contenus Éphémères

- Créer une story avec image
- Stories disparaissant automatiquement après 24h
- Visualiser les stories des utilisateurs suivis
- Header avec stories dans le fil d'actualité

3.7 Recherche et Découverte

- Rechercher des utilisateurs en temps réel
- Grille de découverte de posts populaires
- Filtrage par username

3.8 Reels (Fonctionnalité Bonus)

- Lecture de vidéos courtes verticales
- Navigation par swipe vertical
- Interactions (likes, commentaires)

4. Contraintes Techniques

Concept	Exigence
Framework Mobile	React Native avec Expo SDK 54
Langage	TypeScript pour typage statique et robustesse
Architecture Composants	Composants fonctionnels React avec Hooks
Navigation	React Navigation → Stack Navigator + Tab Navigator
Backend	Firebase (Authentication, Firestore Database, Cloud Storage)
Gestion d'État	Context API (useReducer) pour auth globale + useState pour états locaux
Base de Données	Firestore NoSQL avec 8 collections (users, posts, comments, messages, etc.)
Temps Réel	Listeners Firestore (onSnapshot) pour chat et notifications
Validation	Yup schemas pour validation de formulaires
OAuth	expo-auth-session pour Google Sign-In
Upload Fichiers	Firebase Storage pour images (posts, profils, stories)
Performance	FlatList optimisé, Image caching, Memoization
Responsive Design	SafeAreaView, Dimensions API, design adaptatif
Gestion Version	Git/GitHub avec commits descriptifs et collaboration

Collection	Contenu
users	Profils utilisateurs (uid, username, email, photoURL, bio, followers[], following[])
posts	Publications (userId, imageUrl, caption, likes[], comments count, timestamp)
comments	Commentaires (postId, userId, text, timestamp)
conversations	Conversations (participants[], lastMessage, lastMessageTime)
messages	Messsages individuels (conversationId, senderId, text, read, timestamp)
notifications	Notifications (userId, type, fromUserId, postId, read, timestamp)
stories	Stories éphémères (userId, imageUrl, createdAt, expiresAt)
savedPosts	Posts sauvegardés (userId, postId, savedAt)

5. Architecture Technique

L'application Riminsta suit une architecture en couches typique des applications React Native modernes, avec une séparation claire des responsabilités:

5.1 Couche Présentation (UI)

- **Composants React Native** pour l'interface utilisateur
- **React Navigation** pour la gestion des écrans et transitions
- **Styled Components** et **StyleSheet** pour le styling
- **Composants réutilisables** (Button, Loader, Screen, etc.)

5.2 Couche Logique Métier

- **Hooks personnalisés** (useAuth) pour la logique réutilisable
- **Context API** pour l'état global (authentification)
- **useState/useReducer** pour l'état local des composants
- **Validation Yup** pour les règles métier des formulaires

5.3 Couche Données

- **Firebase Services** comme Backend-as-a-Service
 - Authentication pour la gestion des utilisateurs
 - Firestore pour le stockage NoSQL temps réel
 - Storage pour les fichiers multimédias
- **AsyncStorage** pour la persistance locale

5.5 Structure du Projet

```

app/
  └── components/          # Composants réutilisables
    ├── Auth/               # Composants d'authentification
    ├── Home/               # Composants du feed
    ├── Shared/              # Composants partagés
    └── ...
  └── screens/             # Écrans de l'application
    ├── AuthScreens/        # Écrans d'authentification
    └── MainScreens/        # Écrans principaux
  └── hooks/                # Hooks personnalisés
    └── useAuth.tsx          # Hook authentication
  └── lib/                  # Services externes
    └── firebase.ts          # Configuration Firebase
  └── constants/            # Constantes globales
    ├── theme.ts             # Thème et couleurs
    └── typography.ts        # Typographie
  └── schema/               # Schémas de validation
    ├── signIn.ts            # Fonction pour se connecter
    └── signUp.ts            # Fonction pour s'inscrire
  └── types/                # Types TypeScript
    └── index.d.ts           # Typescript declarations
  └── utils/                # Fonctions utilitaires

```

6. Environnement Technique

Composant	Technologie	Version
Framework Mobile	React Native	0.81.5
Plateforme de développement	Expo	SDK 54.0.0
Langage	TypeScript	5.7.2
Backend (BaaS)	Firebase	9.x
Base de données	Cloud Firestore	NoSQL
Stockage fichiers	Firebase Storage	-
Authentification	Firebase Authentication	-
Navigation	React Navigation	6.x
Gestion d'état	Context API + Hooks	React 18

Validation	Yup	1.x
OAuth	expo-auth-session	Latest
Persistence locale	AsyncStorage	Latest
Sélecteur d'images	expo-image-picker	Latest
Lecteur vidéo	expo-av	Latest
IDE	Visual Studio Code	Latest
Gestion de version	Git/GitHub	-
Gestion de projet	Taiga	Scrum
Package Manager	npm	10.x

7. Planning

Phase	Description	Durée	Livrables
Sprint 1	Configuration & Infrastructure	2 semaines	Setup Expo, Firebase, Navigation, Theme
Sprint 2	Authentification	2 semaines	Welcome, Login, Register, OAuth Google, useAuth hook
Sprint 3	Posts & Feed	2 semaines	Home feed, Post composant, Création post, Likes, Commentaires, Save
Sprint 4	Profils & Social	2 semaines	Profile screen, Edit profile, UserProfile, Follow system, FollowersList
Sprint 5	Messagerie & Notifications	2 semaines	Messages inbox, Chat temps réel, Notifications, Badge count
Sprint 6	Stories, Search & Polish	2 semaines	Stories, CreateStory, Search, Reels, Bug fixes
Phase 7	Documentation	1 semaine	README, Rapport technique, Screenshots
Phase 8	Livraison finale	3 jours	Push GitHub, Setup Taiga, Préparation soutenance

↳ Choix du Modèle de Développement

1. Modèle Retenu: Méthodologie Agile - Framework Scrum

Pour le développement de l'application **Riminsta**, nous avons opté pour la **méthodologie Agile** en utilisant le framework **Scrum**. Ce choix s'est imposé comme le plus adapté aux caractéristiques de notre projet et aux contraintes de notre équipe.

2. Justification du Choix

2.1 Adéquation avec la Nature du Projet

Projet à exigences évolutives:

- Les fonctionnalités d'un réseau social peuvent nécessiter des ajustements fréquents
- Les retours utilisateurs peuvent influencer les priorités de développement
- Les technologies mobiles évoluent rapidement, nécessitant de l'adaptabilité

Projet complexe et innovant:

- Intégration de multiples services (Firebase, OAuth, temps réel)
- Nombreuses dépendances technologiques à gérer
- Nécessité de tests réguliers sur différents environnements (iOS/Android)

2.2 Avantages de l'Approche Agile pour Notre Équipe

Flexibilité et Adaptation:

- Possibilité de réorienter les priorités entre les sprints
- Ajustement des fonctionnalités selon les difficultés rencontrées
- Réponse rapide aux problèmes techniques (bugs, incompatibilités)

Livrailles Incrémentales:

- Chaque sprint produit une version fonctionnelle de l'application
- Possibilité de tester et valider progressivement les fonctionnalités

- Réduction des risques par détection précoce des problèmes

Collaboration Renforcée:

- Réunions quotidiennes (Daily Standup) pour synchronisation
- Revues de sprint pour démonstration et feedback
- Rétrospectives pour amélioration continue

Visibilité du Progrès:

- Backlog transparent avec user stories priorisées
- Burndown charts pour suivre l'avancement
- Démonstrations régulières au professeur/encadrant

2.3 Comparaison avec d'Autres Modèles

Critère	Modèle en Cascade	Agile/Scrum	Notre Choix
Flexibilité	✗ Faible (changements coûteux)	✓ Élevée (changements intégrés)	Scrum
Retour utilisateur	✗ Tardif (fin de projet)	✓ Continu (chaque sprint)	Scrum
Gestion des risques	✗ Risques découverts tardivement	✓ Détection précoce	Scrum
Documentation	✓ Complète dès le début	⚠ Évolutive	Scrum + Doc finale
Complexité projet	✓ Adapté si exigences stables	✓ Adapté si exigences évolutives	Scrum
Taille équipe	⚠ Équipes moyennes/grandes	✓ Petites équipes (3-9)	Scrum
Durée projet	⚠ Projets longs (>6 mois)	✓ Projets courts/moyens	Scrum

Chapitre 2

Conception du système

3.1 Modélisation dynamique

3.2.1 Diagrammes de séquences

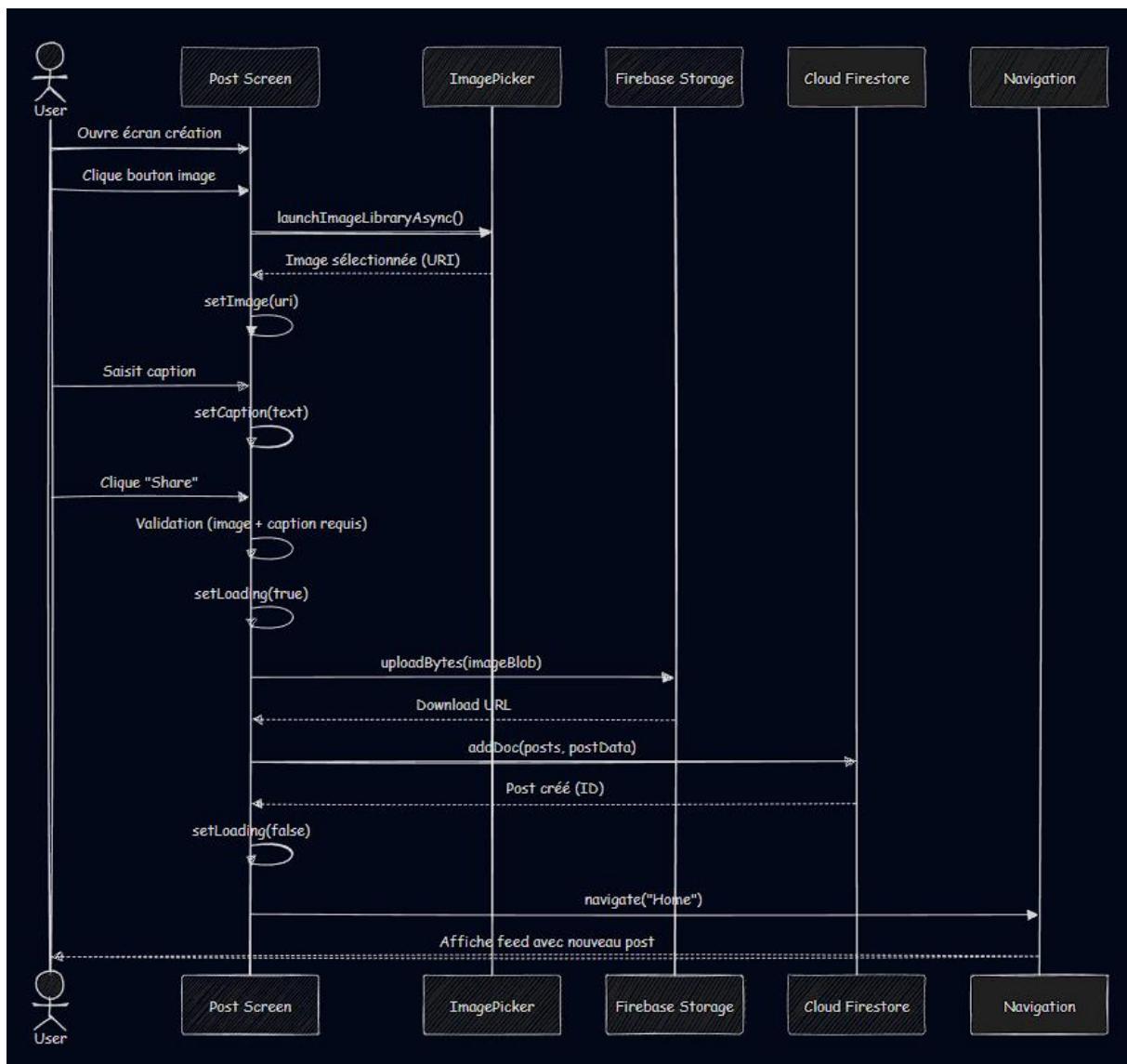


Figure 3.1 – Diagramme de séquence - ajouter **Création de Post**

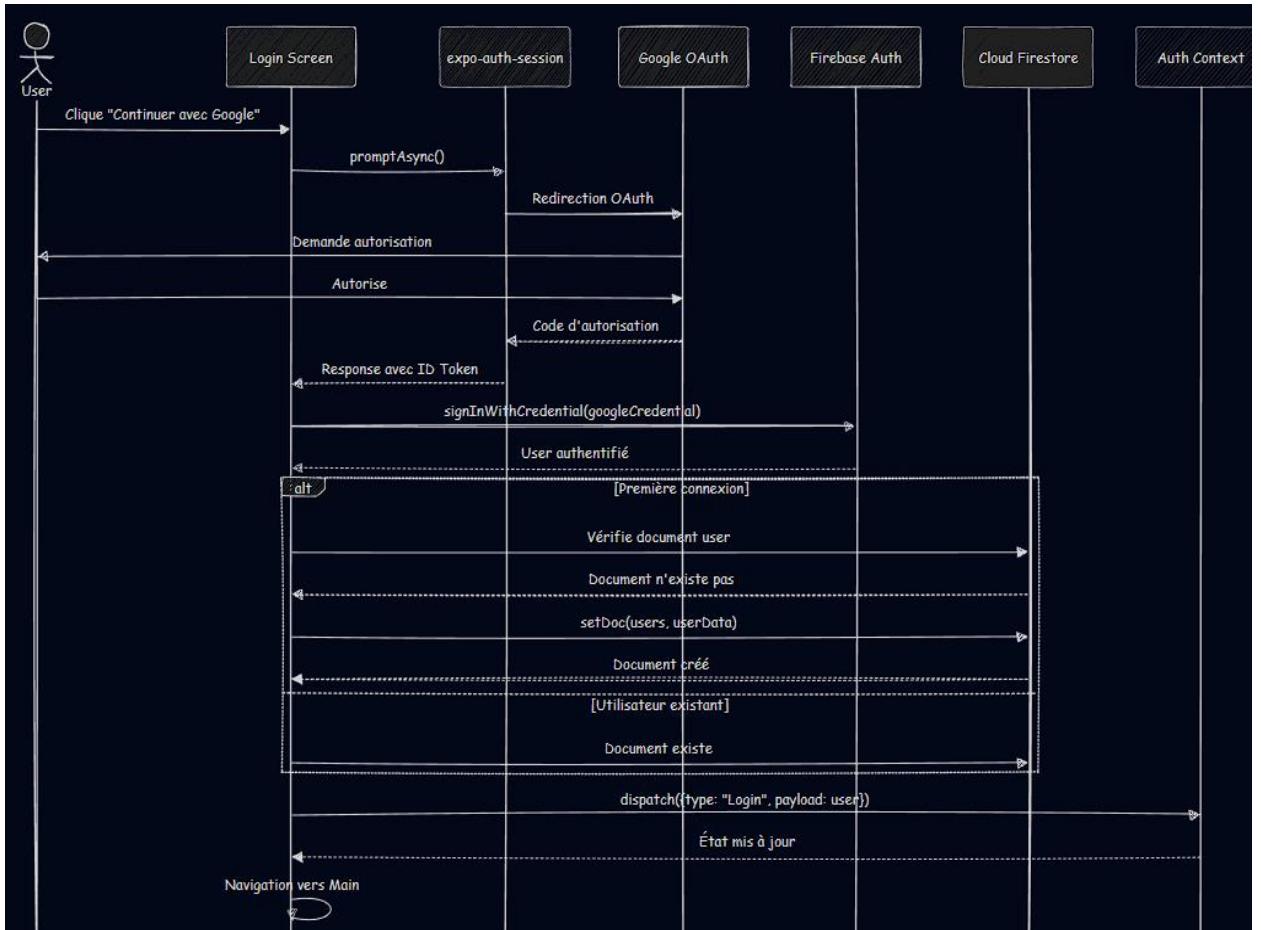


Figure 3.2 – Diagramme de séquence - **Authentification Google OAuth**

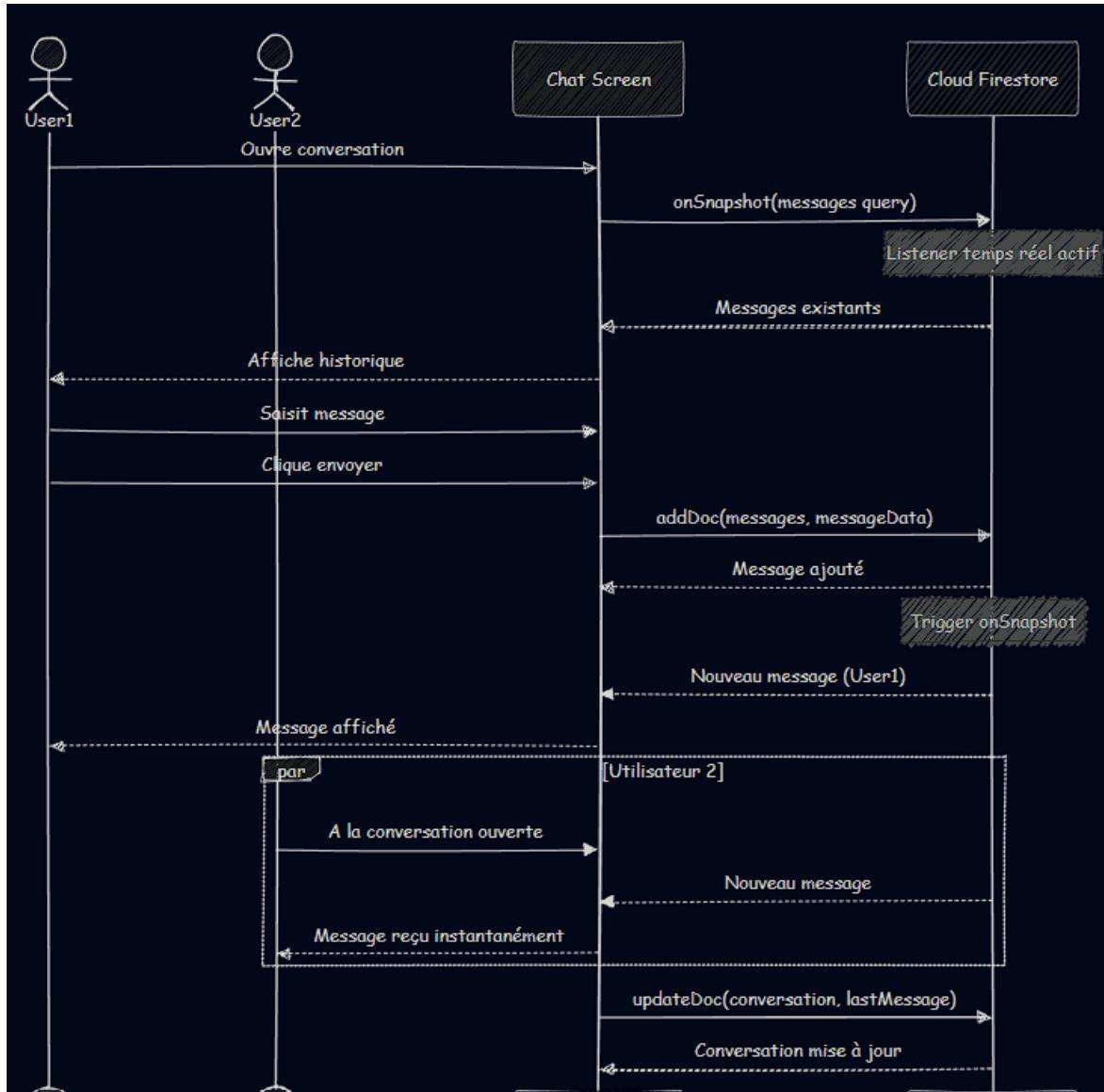


Figure 3.3 – Diagramme de séquence - **Chat Temps Réel**

3.2.3 Diagrammes d'états

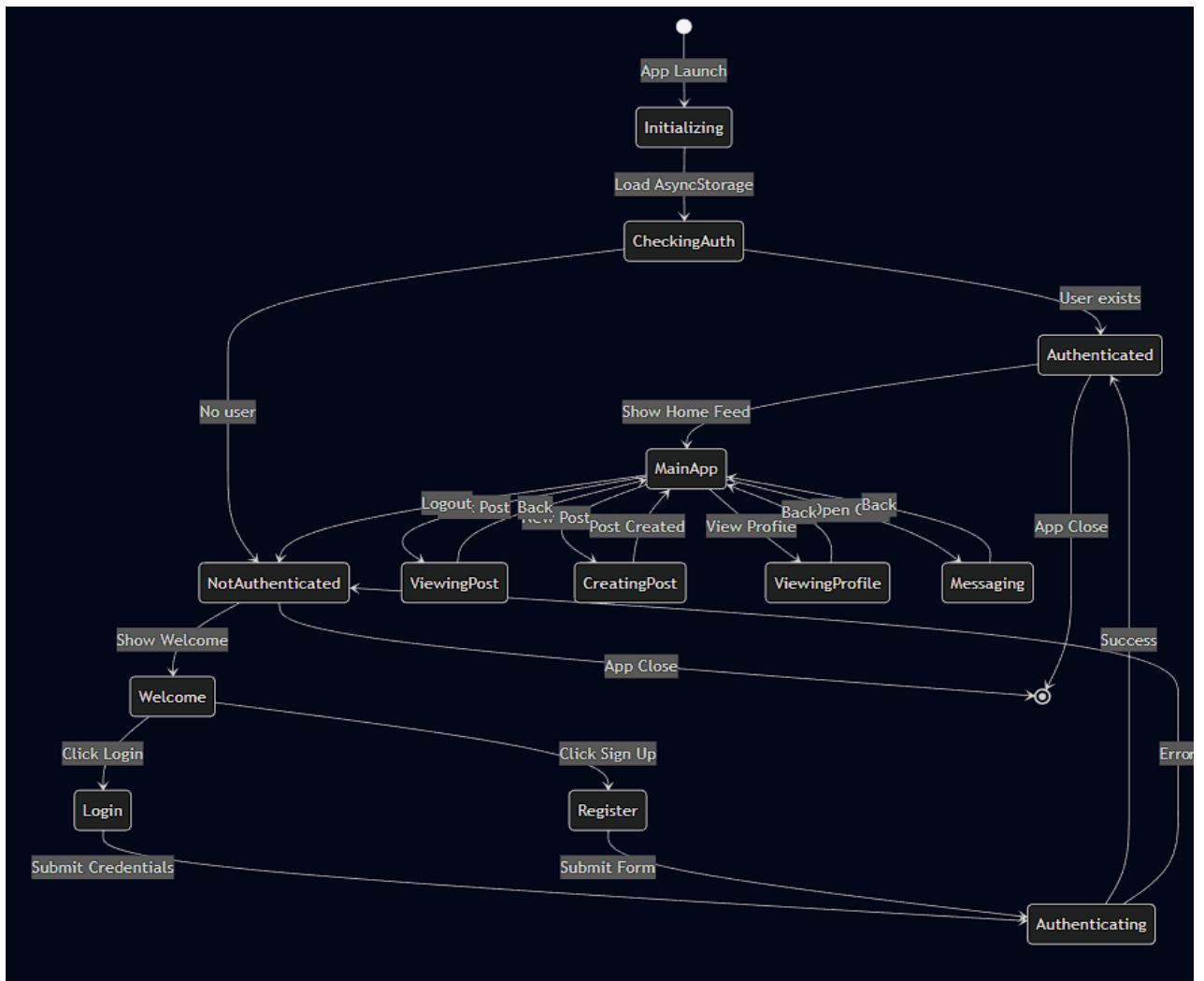


Figure 3.6 – Diagramme d'états – **Gestion de Session**

3.2.5 Diagrammes d'activité

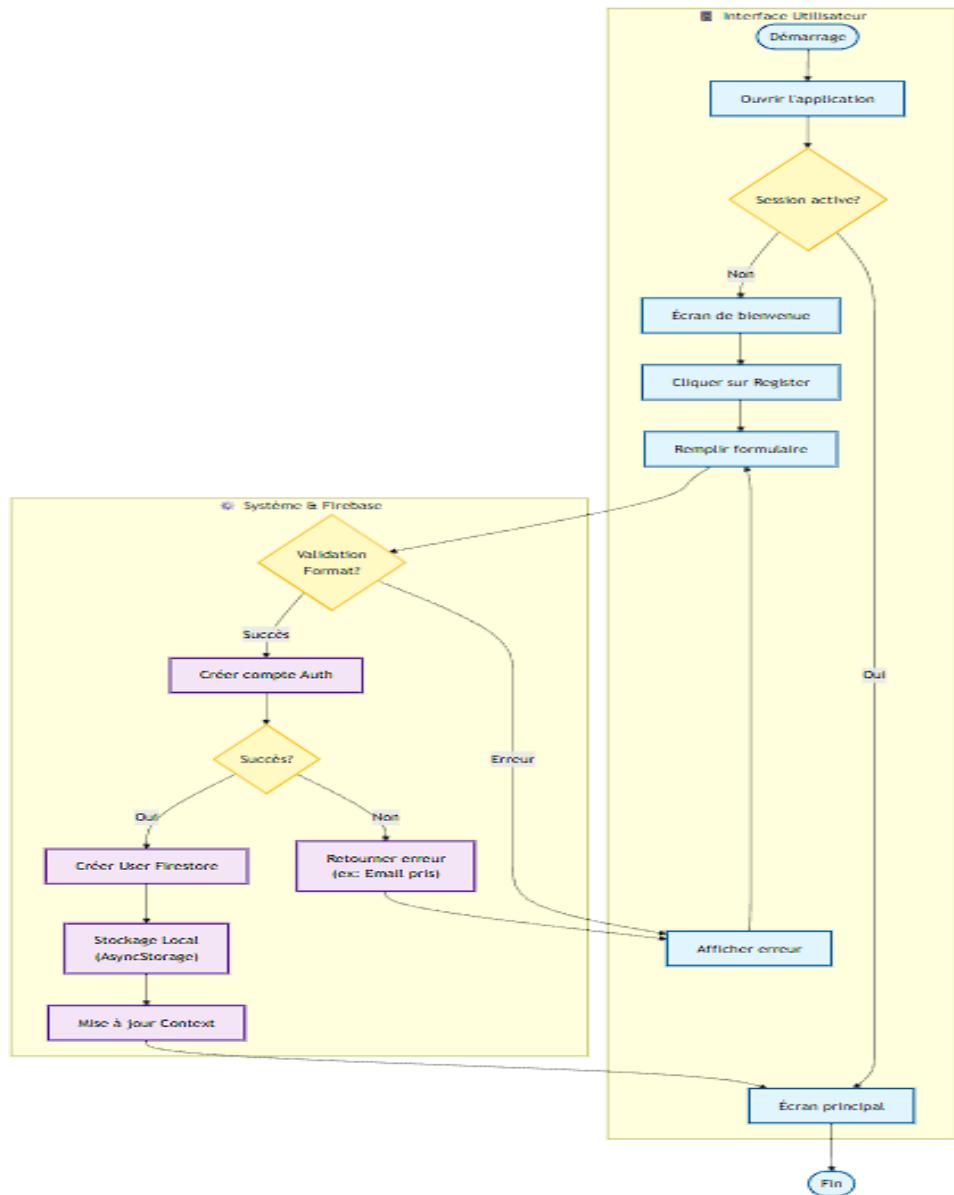


Figure 3.9 – Diagramme d'activité - inscription d'un utilisateur

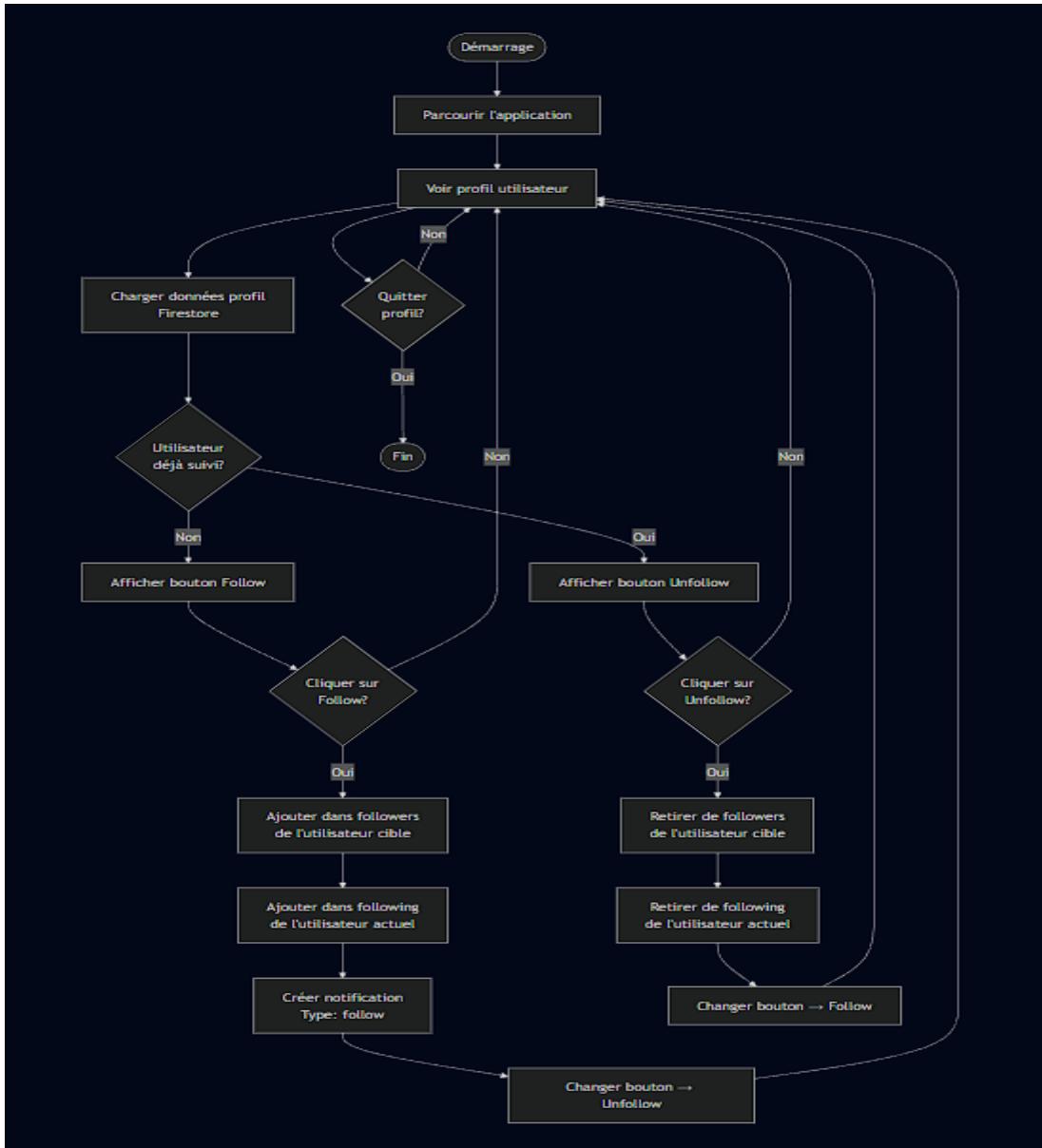


Figure 3.10 – Diagramme d'activité - **Follow/Unfollow un Utilisateur**

3.3 Modélisation statique

3.3.1 Diagramme de classes

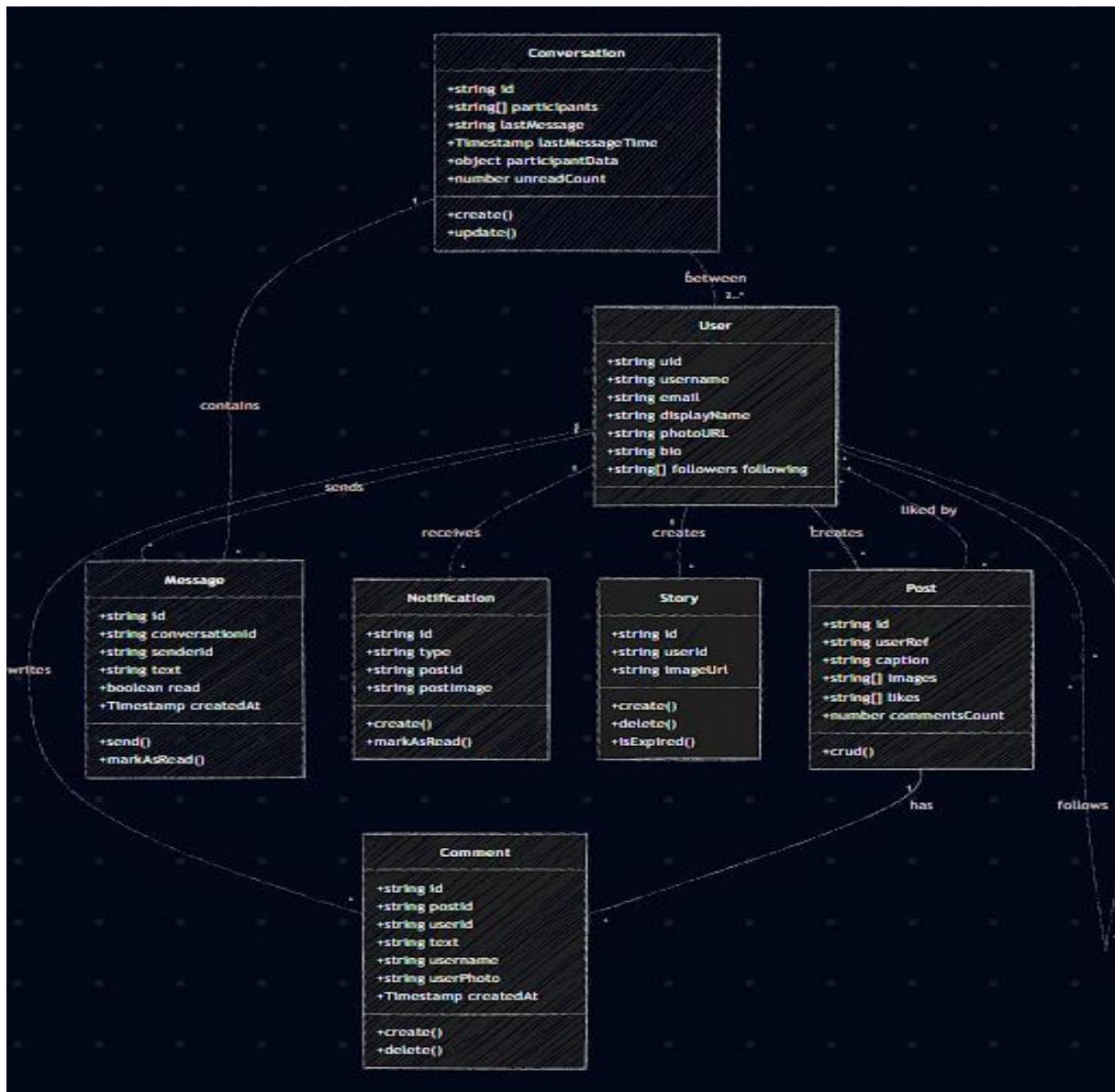


Figure 3.13 – Diagramme de classes

3.3.2 use case

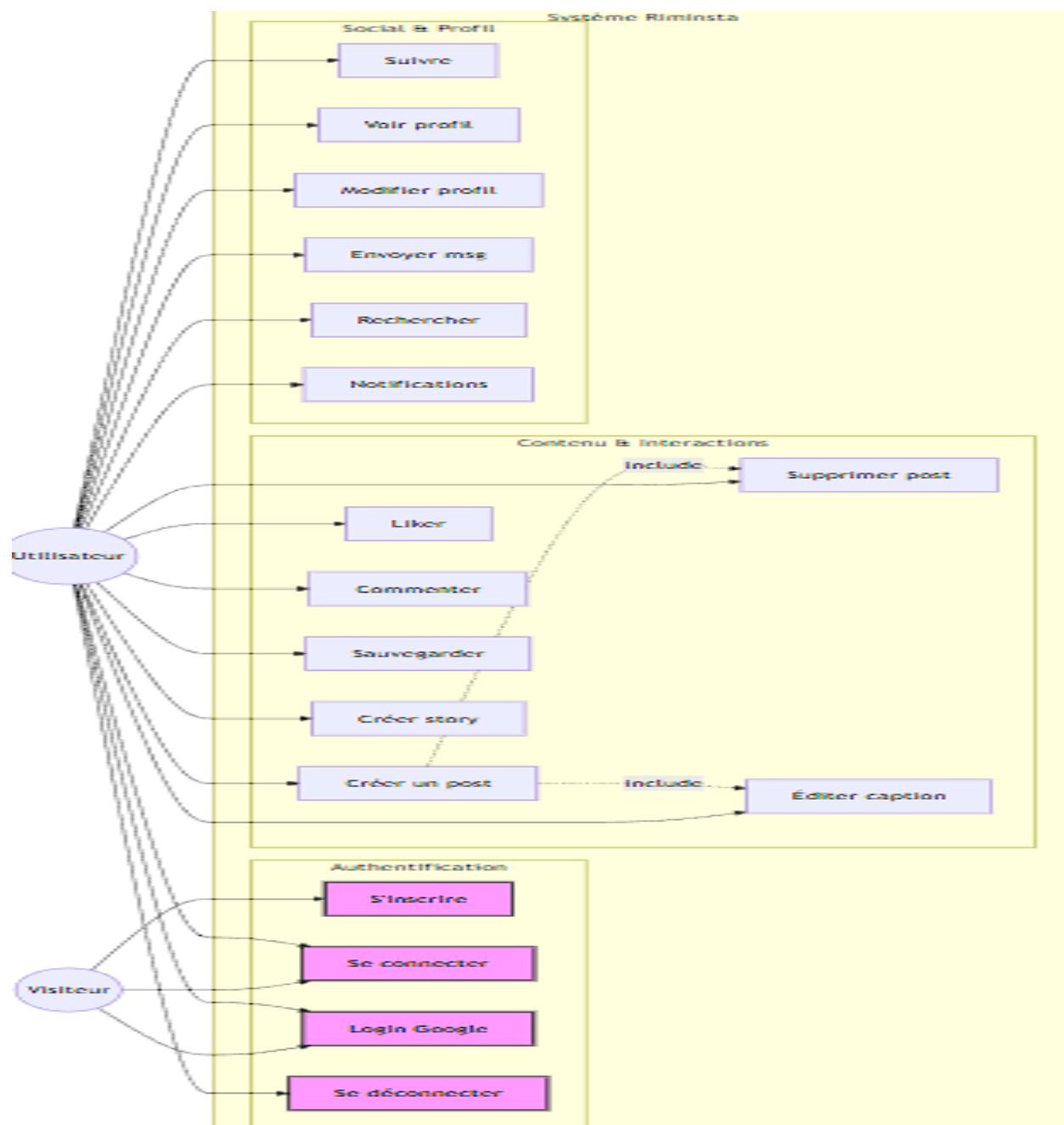


Figure 3.13 – Diagramme de cas d'utilisation

3.3.3 Architecture de l'application

```

app/
  └── components/          # Composants réutilisables
      ├── Auth/             # Boutons, messages d'erreur
      ├── Home/             # Header, Posts, Stories
      ├── Reels/             # Composants reels
      ├── Search/            # Grille de photos
      └── Shared/            # Loader, Screen, Modals
  └── screens/              # Écrans de l'application
      └── AuthScreens/       # Welcome, Login, Register

```

```
|   └── MainScreens/      # Home, Profile, Messages, etc.  
|   ├── constants/        # Thème, typographie  
|   ├── hooks/            # useAuth  
|   ├── lib/               # Configuration Firebase  
|   ├── schema/           # Validation Yup  
|   ├── types/             # Types TypeScript  
|   └── utils/            # Fonctions utilitaires
```

Chapitre 3

Réalisation du système

4.1 Environnement de développement

TECHNOLOGIES UTILISEES

Voici le tableau adapté pour votre projet Riminsta:

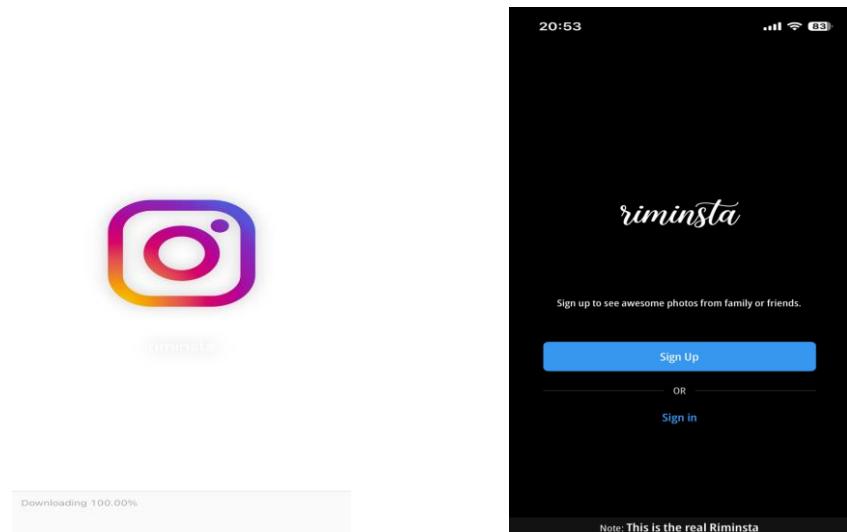
TECHNOLOGIES UTILISÉES

Catégorie	Technologie	Version	Utilisation
Framework Mobile	React Native	0.81.5	Framework principal pour développement cross-platform
Plateforme	Expo	SDK 54.0.0	Environnement de développement et build mobile
Langage	TypeScript	5.7.2	Langage principal avec typage statique
Backend (BaaS)	Firebase	9.x	Backend-as-a-Service complet
Authentication	Firebase Authentication	9.x	Gestion des utilisateurs et sessions
Base de données	Cloud Firestore	NoSQL	Stockage temps réel des données
Stockage fichiers	Firebase Storage	-	Hébergement images et médias
Navigation	React Navigation	6.x	Gestion Stack et Tab navigation
State Management	Context API + Hooks	React 18	Gestion d'état global et local
Validation	Yup	1.x	Validation des formulaires
OAuth Provider	expo-auth-session	Latest	Authentification Google
Persistence locale	AsyncStorage	Latest	Stockage local sécurisé
Sélecteur images	expo-image-picker	Latest	Accès caméra et galerie
Lecteur vidéo	expo-av	Latest	Lecture vidéos (Reels)
Package Manager	npm	10.x	Gestion des dépendances

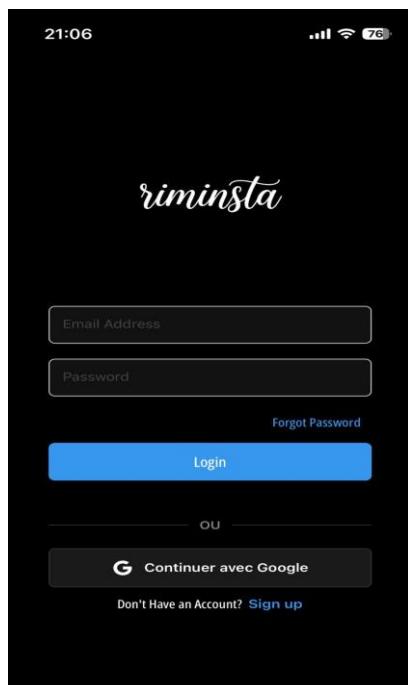
IDE	Visual Studio Code	Latest	Environnement de développement
Version Control	Git/GitHub	-	Gestion de versions et collaboration
Gestion projet	Taiga	-	Méthodologie Scrum et suivi

4.3- CONSOLE

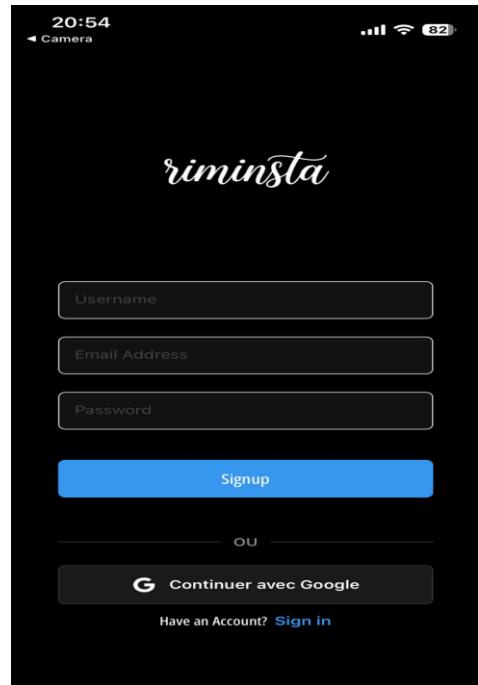
1. **Welcome Screen**



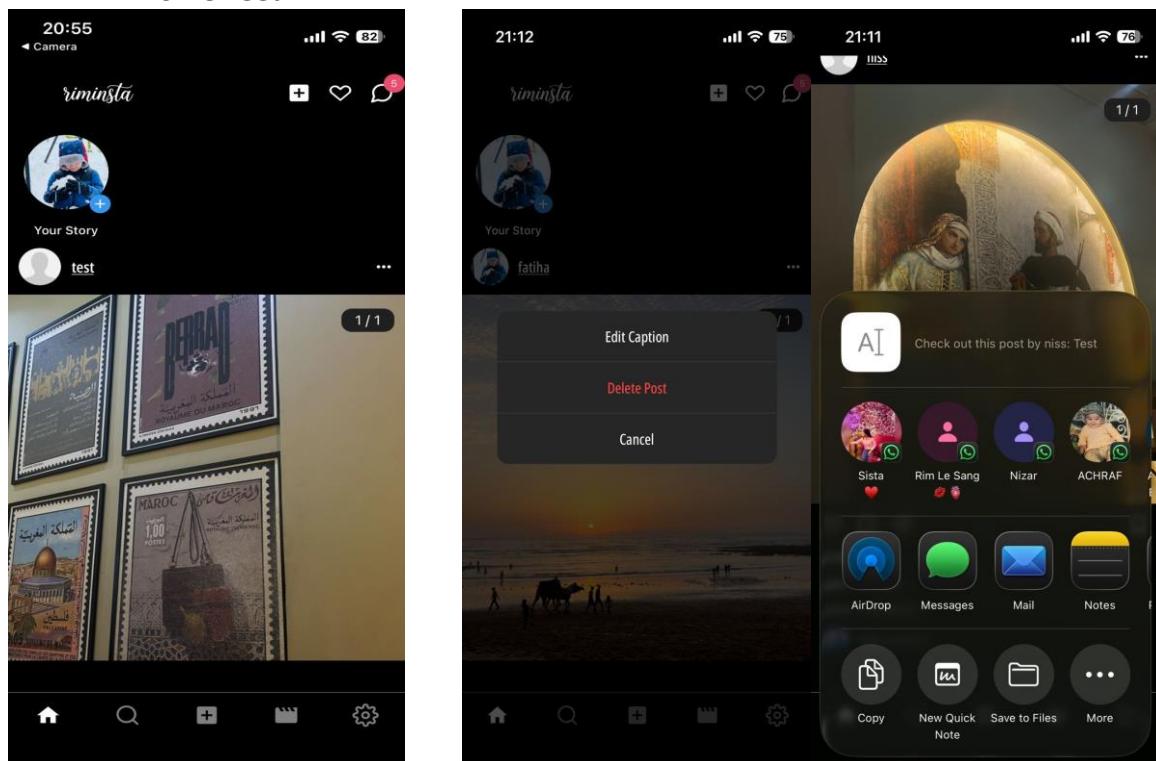
2. **Login Screen**



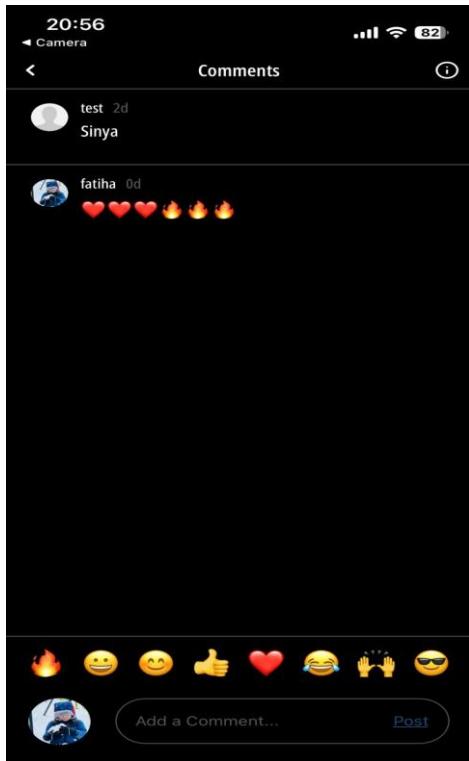
3. **Register Screen**



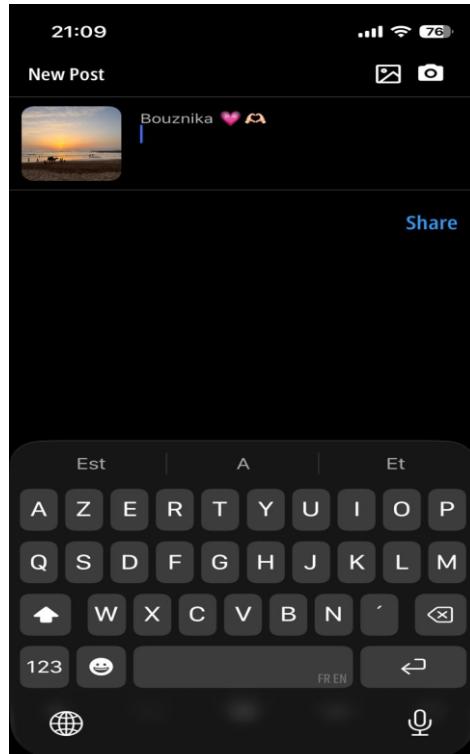
4. **Home Feed**



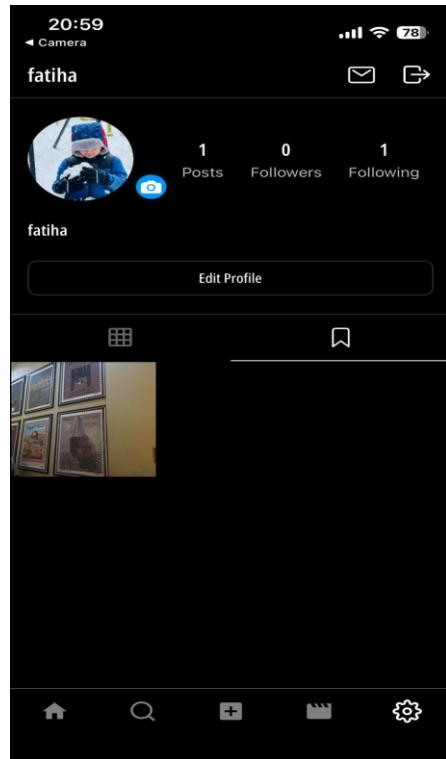
5. **Post avec Commentaires**



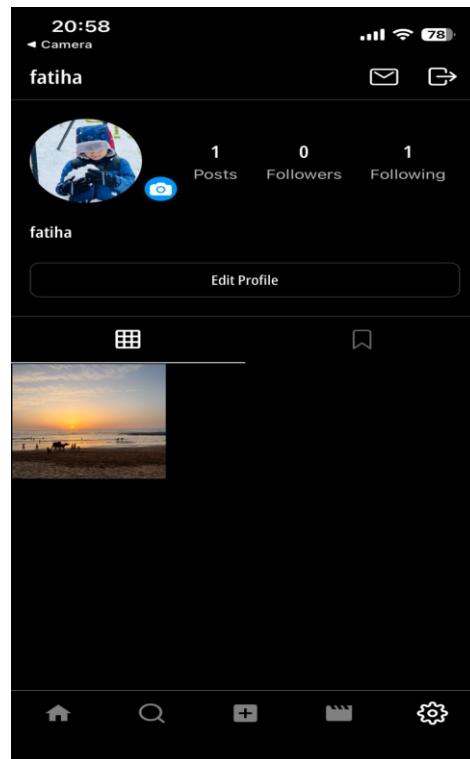
6. **Création de Post**



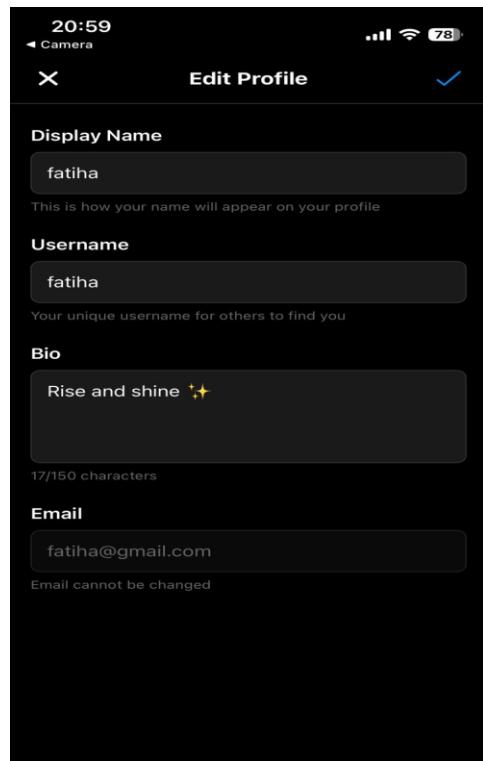
7. **Post Sauvegardé**



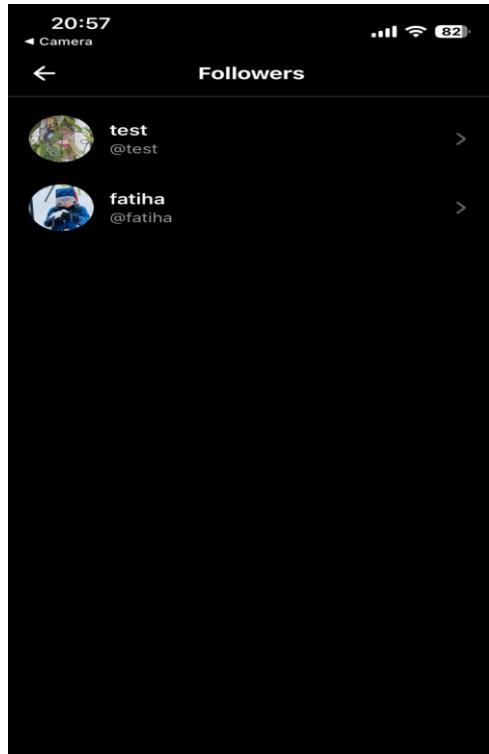
8. **Profil Personnel**



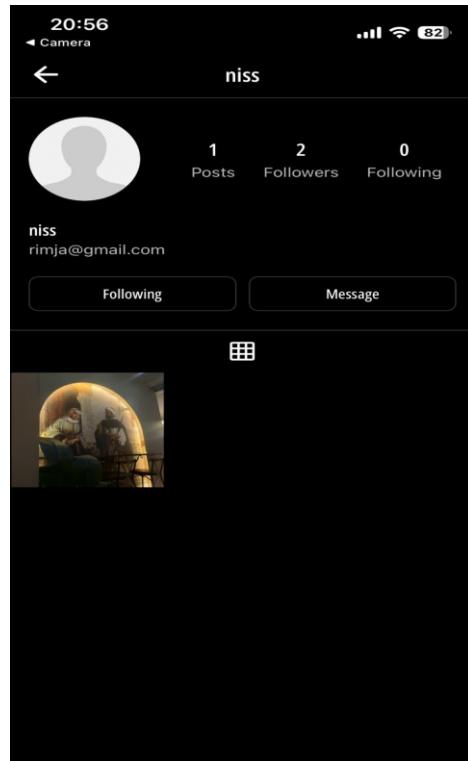
9. **Edit Profile**



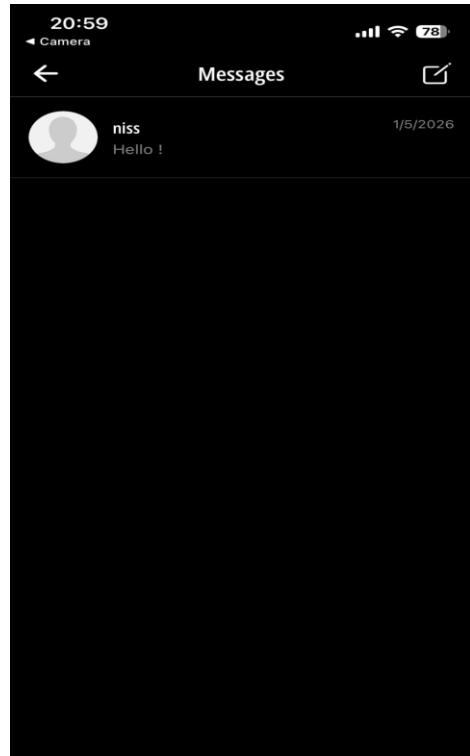
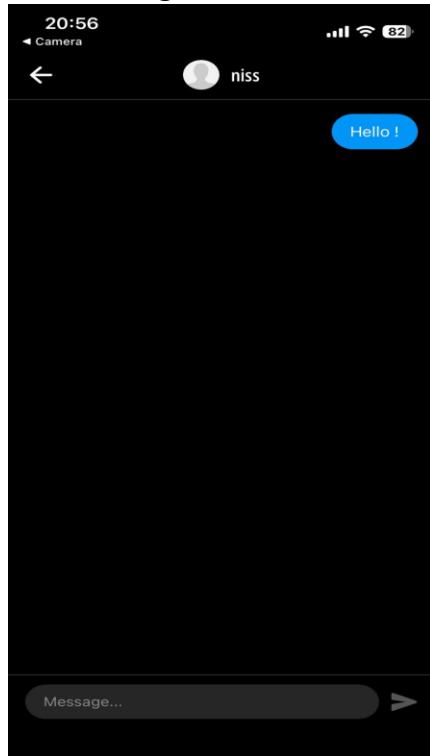
10. **Liste des Followers**



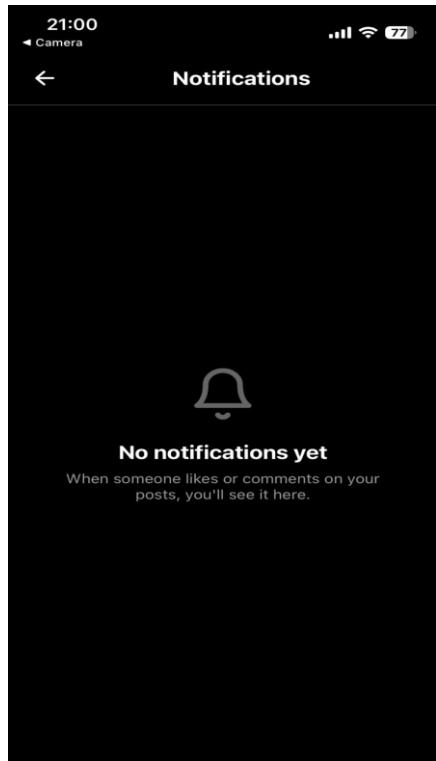
11. **Profil d'un Autre Utilisateur**



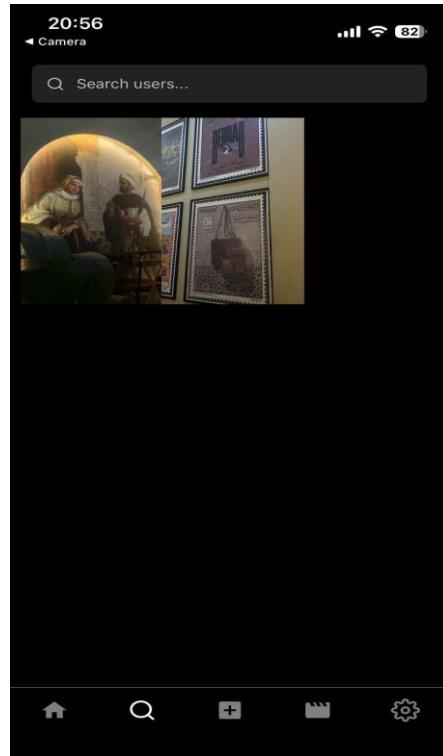
12. **Messages Inbox**



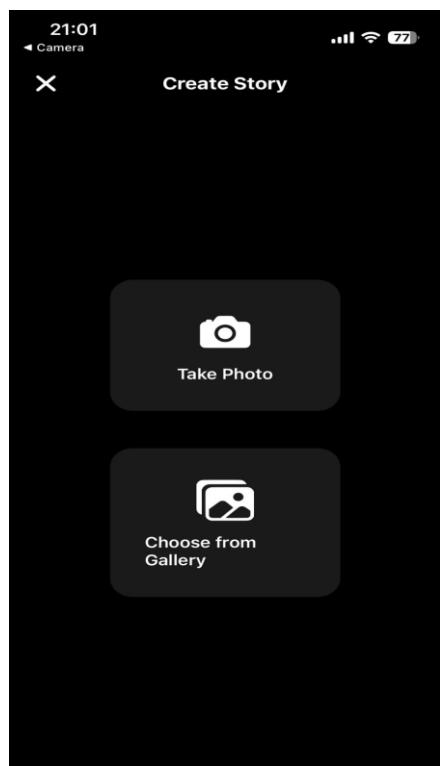
13. **Notifications**



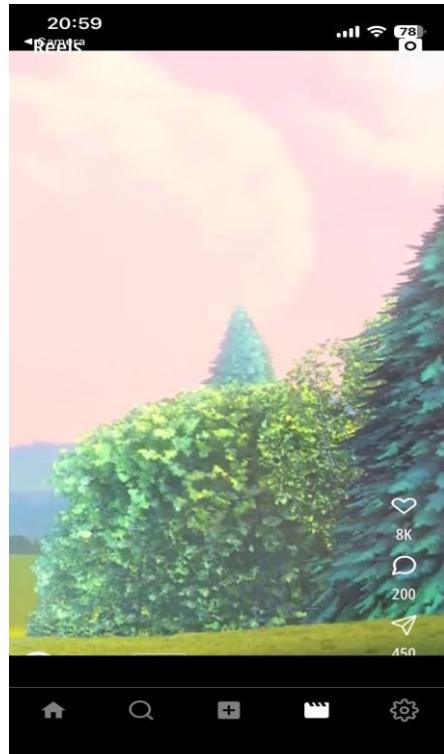
14. **Search**



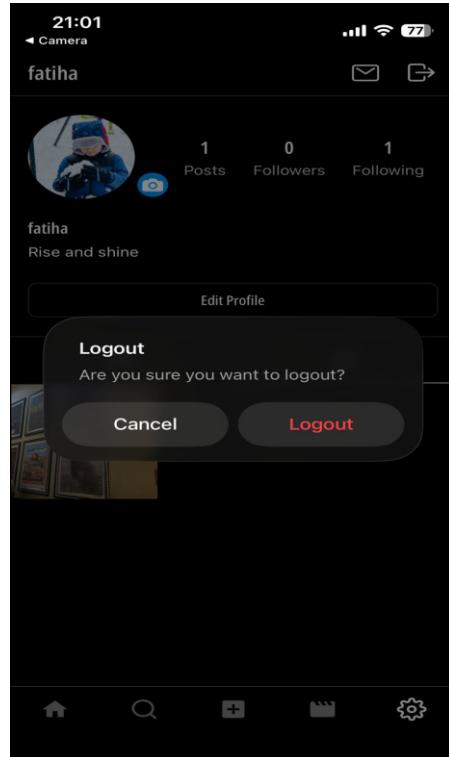
15. **Stories Creation**



16. **Reels**



17. **logout**



Conclusion générale

Le projet **Riminsta** a permis de développer une application mobile complète de réseau social en utilisant React Native et l'écosystème Firebase. Ce projet a été l'occasion de mettre en pratique les concepts fondamentaux du développement mobile moderne ainsi que les technologies actuelles de l'industrie du développement logiciel.

À travers ce travail, nous avons implémenté les principes de la **programmation orientée composants** avec React, illustrés par la création de composants réutilisables (PostHeader, PostFooter, Screen, Button, etc.) et l'utilisation des hooks personnalisés (useAuth). L'utilisation de **TypeScript** nous a permis de garantir la robustesse du code grâce au typage statique, réduisant considérablement les erreurs potentielles lors du développement.

La **gestion d'état** avec Context API et les hooks React (useState, useReducer, useEffect) a assuré une coordination efficace entre les différents composants de l'application. L'intégration de **Firebase** comme Backend-as-a-Service a permis d'implémenter rapidement l'authentification multi-providers, la base de données temps réel avec Firestore, et le stockage de fichiers multimédias avec Cloud Storage.

Les **listeners temps réel** avec onSnapshot() ont été exploités pour la messagerie instantanée et les notifications, démontrant la maîtrise des concepts de

programmation asynchrone et événementielle. La **navigation** avec React Navigation a permis de créer une expérience utilisateur fluide avec Stack et Tab Navigators, tandis que la validation avec Yup a renforcé la sécurité des formulaires d'inscription et de connexion.

L'**architecture en couches** adoptée (Présentation/UI, Logique Métier, Données) facilite la maintenance et l'évolution future de l'application. La séparation claire entre les composants, les screens, les hooks et les services Firebase constitue une base solide pour le développement d'applications mobiles professionnelles. L'utilisation de la **méthodologie Agile Scrum** avec Taiga a permis une gestion de projet efficace et une collaboration structurée entre les membres de l'équipe.

Ce projet nous a également permis de maîtriser des concepts avancés tels que:

- L'**authentification OAuth** avec Google Sign-In via expo-auth-session
- La **persistence locale** avec AsyncStorage pour maintenir les sessions utilisateurs
- L'**optimisation des performances** avec FlatList, memoization et image caching
- La **gestion des médias** avec expo-image-picker pour l'accès à la caméra et la galerie
- Les **notifications push** et le système de badge pour alerter les utilisateurs des interactions

Les défis techniques rencontrés, notamment la configuration du Google OAuth, la gestion des index composites Firestore, et l'implémentation du chat temps réel, ont renforcé notre capacité à résoudre des problèmes complexes et à rechercher des solutions dans la documentation officielle et les ressources communautaires.

En perspective, l'application pourrait être enrichie par plusieurs améliorations:

Court terme:

- Ajout de stories vidéo (actuellement images uniquement)
- Implémentation du système de mentions (@username) dans les commentaires
- Ajout de hashtags pour faciliter la découverte de contenus
- Système de filtres et effets pour les photos

Moyen terme:

- Notifications push natives avec Firebase Cloud Messaging
- Partage de posts vers d'autres réseaux sociaux
- Mode sombre/clair personnalisable
- Support multi-langues (i18n)
- Statistiques détaillées pour les profils (analytics)

Long terme:

- Système de vérification des comptes (badge bleu)
- Publicités sponsorisées pour monétisation
- Live streaming vidéo en temps réel
- Intelligence artificielle pour recommandations personnalisées
- Migration vers une architecture microservices pour scalabilité accrue
- Application web avec React.js partageant le même backend Firebase

Ce projet constitue une démonstration concrète de notre capacité à concevoir, développer et livrer une application mobile complexe en utilisant les technologies et méthodologies actuelles de l'industrie. Les compétences acquises (développement mobile cross-platform, gestion de backend cloud, architecture moderne, travail en équipe agile) sont directement transférables au monde professionnel et constituent une base solide pour notre future carrière d'ingénieurs en développement logiciel.

Annexes

A. Glossaire

- **Firebase:** Plateforme backend-as-a-service de Google
- **Firestore:** Base de données NoSQL en temps réel
- **OAuth:** Protocole d'authentification standard
- **React Native:** Framework mobile JavaScript
- **Expo:** Plateforme de développement React Native
- **TypeScript:** Superset typé de JavaScript
- **SDK:** Software Development Kit

B. Références

- Documentation React Native: <https://reactnative.dev/>
- Documentation Expo: <https://docs.expo.dev/>
- Documentation Firebase: <https://firebase.google.com/docs>
- React Navigation: <https://reactnavigation.org/>

