Rapport de Projet

Application Mobile CryptoNews

Application Cross-Platform pour le Suivi des Cryptomonnaies

Présenté par :
Idris Benaich
Mouad Hachoum
Amine Zmarrou

Encadré par :
Zineb Naji

Année Académique : 2024-2025

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet. Mes remerciements s’adressent particulièrement à mon encadrant pour ses conseils précieux et son expertise. Je remercie également le corps professoral pour la formation de qualité dispensée ainsi que ma famille et mes amis pour leur soutien constant.

Résumé

Ce rapport documente le développement de CryptoNews, une application mobile cross-platform développée avec React Native et Expo Router. L'application offre des fonctionnalités complètes pour le suivi des cryptomonnaies : données de marché en temps réel, agrégation de nouvelles, gestion de portefeuille et fonctionnalités communautaires. Le pro- jet intègre Firebase pour l'authentification et la persistence des données.

Mots-clés : Application Mobile, React Native, Firebase, Cryptomonnaies, Cross-Platform

Table des matières

Table des figures

- *Table des Figures**

Chapitre 1

Présentation du Cadre du Projet

1.1 Contexte Général

Le marché des cryptomonnaies connaît une croissance exponentielle, créant une forte demande pour des applications mobiles accessibles fournissant des informations en temps réel. Les utilisateurs recherchent des solutions pour suivre leurs investissements, consulter les actualités du marché et gérer leurs portefeuilles de manière simple et efficace.

1.2 Problématique

Les investisseurs en cryptomonnaies ont besoin d'une plateforme unifiée permettant de:

— Consulter les prix et tendances du marché en temps réel
— Accéder aux dernières actualités crypto
— Gérer et valoriser leur portefeuille
— Échanger avec la communauté
— Sécuriser leurs données personnelles

1.3 ObjectifsduProjet

1.3.1 ObjectifsPrincipaux

1. Développer une application mobile cross-platform (iOS, Android, Web)
2. Implémenter une authentification sécurisée avec Firebase
3. Créer un système de synchronisation en temps réel
4. Concevoir une interface utilisateur intuitive et moderne
5. Intégrer des APIs externes pour les données de marché

1.3.2 Critères de Réussite

— Compatibilité multi-plateformes
— Authentification sécurisée
— Mises à jour en temps réel
— Interface responsive et intuitive
— Architecture évolutive et maintenable

1.4 Portée du Projet

Le projet couvre la conception, le développement et l'implémentation d'une application mobile complète démontrant des pratiques modernes d'ingénierie logicielle : architecture basée sur les composants, gestion d'état, intégration d'API et persistence cloud.

Chapitre 2

Conception du Projet

2.1 ArchitectureGlobale

L'application suit une architecture en couches séparant clairement les responsabilités :
— Couche Présentation : Composants UI et écrans
— Couche Application : Logique métier et gestion d'état
— Couche Service : Communication avec APIs externes
— Couche Données : Persistence et stockage

2.1.1 Diagrammed'ArchitectureTechnique

La figure 2.1 illustre l'architecture technique complète de l'application CryptoNews, montrant les relations entre les différentes couches du système.

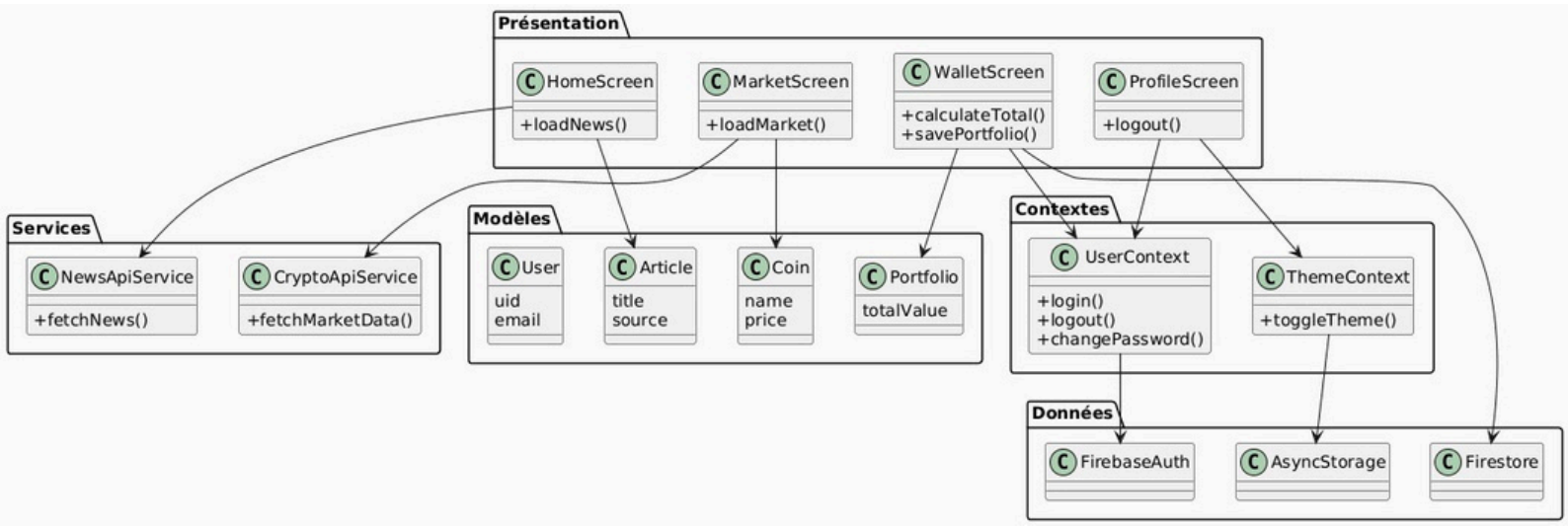

Figure 2.1 – Architecture technique de l'application CryptoNews

L'architecture se compose de quatre modules principaux :

CouchePrésentation Cettecouchecontientl'ensembledesécransdel'application:
— HomeScreen : Affiche le flux d'actualités crypto avec la méthode `loadNews()`
— MarketScreen : Présente les données de marché en temps réel via `loadMarket()`
—WalletScreen:Gèreleportefeuilleutilisateuravec `calculateTotal()` et `savePortfolio()`
— ProfileScreen : Permet la gestion du profil et la déconnexion via `logout()`

CoucheServices Lesservicesassurentlacommunicationaveclessourcesdedonnées externes :
— NewsApiService : Récupère les articles d'actualité via fetchNews()
— CryptoApiService : Obtient les données de marché des cryptomonnaies avec `fetchMarketData()`

Couche Modèles Les modèles définissent la structure des données manipulées :

— User : Stocke l'identifiant (uid) et l'email de l'utilisateur
— Article : Contient le titre et la source des articles
— Coin : Représente une cryptomonnaie avec son nom et son prix
— Portfolio : Gère la valeur totale du portefeuille

Couche Contextes et Données Cette couche assure la gestion de l'état global et la persistence :

—UserContext:Gèrel'authentificationavec `login()`,`logout()` et`changePassword()`
— ThemeContext : Contrôle le thème de l'application via `toggleTheme()`
— FirebaseAuth : Service d'authentification Firebase
— AsyncStorage : Stockage local des préférences
— Firestore : Base de données cloud pour la synchronisation

2.2 DiagrammedesCasd'Utilisation

La figure 2.2 présente les différents cas d'utilisation de l'application selon le statut d'authentification de l'utilisateur.

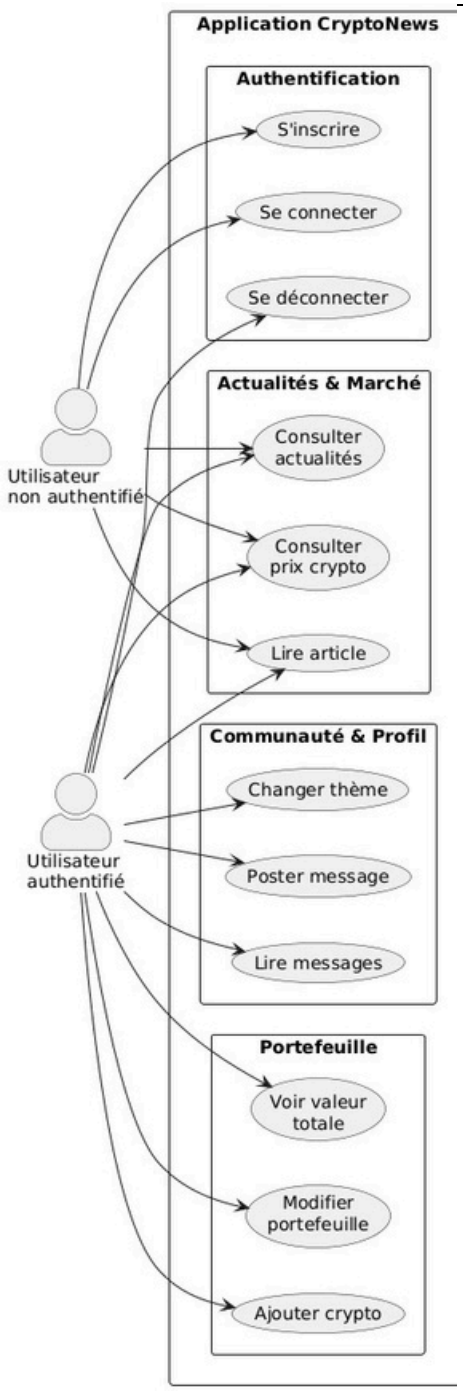

Figure 2.2 – Diagramme des cas d'utilisation de l'application

2.2.1 Scénarios pour Utilisateur Non

Un utilisateur non authentifié peut accéder à des fonctionnalités limitées :

Authentifié

Authentification

- S'inscrire :Créationd'unnouveaucompteutilisateur
- Se connecter:Connexionàuncompteexistant
- Se déconnecter :Fermeturedesession(accessibleaprèsconnexion)

Actualités & Marché

- Consulter prix crypto :Visualisationdesprixactuelsdescryptomonnaies
- Consulter actualités :Lecturedesdernièresnouvellesdusecteur
- Lire article :Accèsaucontenucompletd'unarticle

2.2.2 ScénariospourUtilisateurAuthentifié

Une fois authentifié, l'utilisateur accède à l'ensemble des fonctionnalités de l'application :

Portefeuille

— Ajouter crypto : Ajout de nouvelles cryptomonnaies au portefeuille
— Modifier portefeuille : Mise à jour des quantités détenues
— Voir valeur totale : Consultation de la valorisation globale

Communauté & Profil
— Lire messages : Consultation des messages du chat communautaire
— Poster message : Publication de messages dans le chat
— Changer thème : Basculement entre thème clair et sombre

Cette séparation des fonctionnalités encourage les utilisateurs à créer un compte tout en leur permettant de découvrir l'application avant l'inscription.

2.3 Structure de l'Application

Listing 2.1 – Organisation des dossiers

```
cryptonews-app/
        app/
            (tabs)/                 # Navigation principale
                index.tsx           # Flux de nouvelles
                market.tsx          # Donnees de marche
                wallet.tsx          # Portefeuille
                community.tsx       # Chat communautaire
                profile.tsx         # Profil utilisateur
            auth/                   # Authentification
                login.tsx
                register.tsx
                security.tsx
            news/[id].tsx           # Detail article
        src/
            config/
                firebase.js         # Configuration Firebase
            context/
                UserContext.js      # Data authentification
                ThemeContext.js     # Data theme
            services/               # Data
                newsApi.js
                cryptoApi.js
        components/             # Composants utilisables
        assets/                 # Ressources
```

2.4 ArchitecturedeNavigation

L'application utilise Expo Router avec un système de routage basé sur les fichiers :

2.4.1 NavigationparOnglets

— Home (/) - Flux de nouvelles
— Market (/market) - Données de marché
— Wallet (/wallet) - Gestion de portefeuille
— Community (/community) - Chat communautaire
— Profile (/profile) - Profil utilisateur

2.4.2 NavigationModale

— Login (/auth/login)
— Register (/auth/register)
— Security (/auth/security)

2.5 Schéma de Base de Données

2.5.1 Firebase Firestore Collections

Collection users

Stocke les données de portefeuille des utilisateurs :

Listing 2.2 – Structure users

```
{
    userId: string,
    portfolio: {
        bitcoin: string,
        ethereum: string,
        solana: string,
        binancecoin: string,
        ripple: string
    }
}
```

Collection messages

Stocke les messages du chat communautaire :

Listing 2.3 – Structure messages

```
{
    messageId: string,
    text: string,
    userId: string,
    userEmail: string,
    createdAt: Timestamp
```

2.6 Conception de l'Interface

2.6.1 Système de Thème

L'application supporte deux thèmes avec basculement dynamique :
Thème Sombre :
— Arrière-plan : #0B0E14
— Surface : #151C24
— Primaire : #3B82F6
— Texte : #F8FAFC
Thème Clair :
— Arrière-plan : #FFFFFF
— Surface : #F5F5F5
— Primaire : #007AFF
— Texte : #000000

2.6.2 Flux de Données

Le pattern de flux de données unidirectionnel :

1. Interaction Utilisateur → Événement déclenché
2. Mise à Jour d'État → Context API gère l'état global
3. Appels API → Services récupèrent les données
4. Traitement → Validation et transformation
5. Mise à Jour UI → Re-rendu des composants
6. Persistence → Sauvegarde dans Firestore

Chapitre 3

Réalisation du Système

3.1 Interface de l'Application

3.1.1 Écran d'Accueil - Flux de Nouvelles

L'écran d'accueil affiche les dernières actualités du monde des cryptomonnaies avec une interface intuitive.

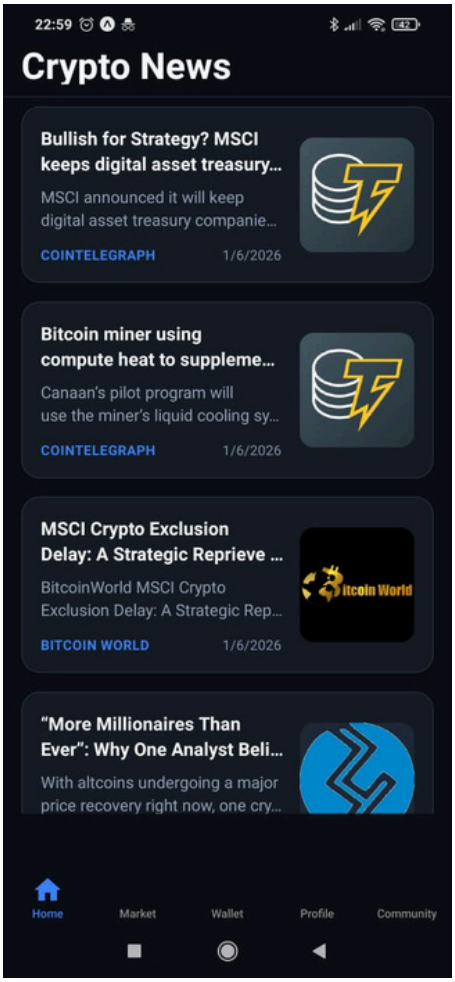

Figure 3.1 – Écran d'accueil avec flux de nouvelles crypto

Fonctionnalités:

— Affichage des articles d'actualité avec images
— Pull-to-refresh pour actualiser le contenu
— Navigation vers les détails de l'article
— Badges de source et de date de publication

3.1.2 DétailsdesArticles

Les utilisateurs peuvent accéder au contenu complet des articles avec possibilité de partage.

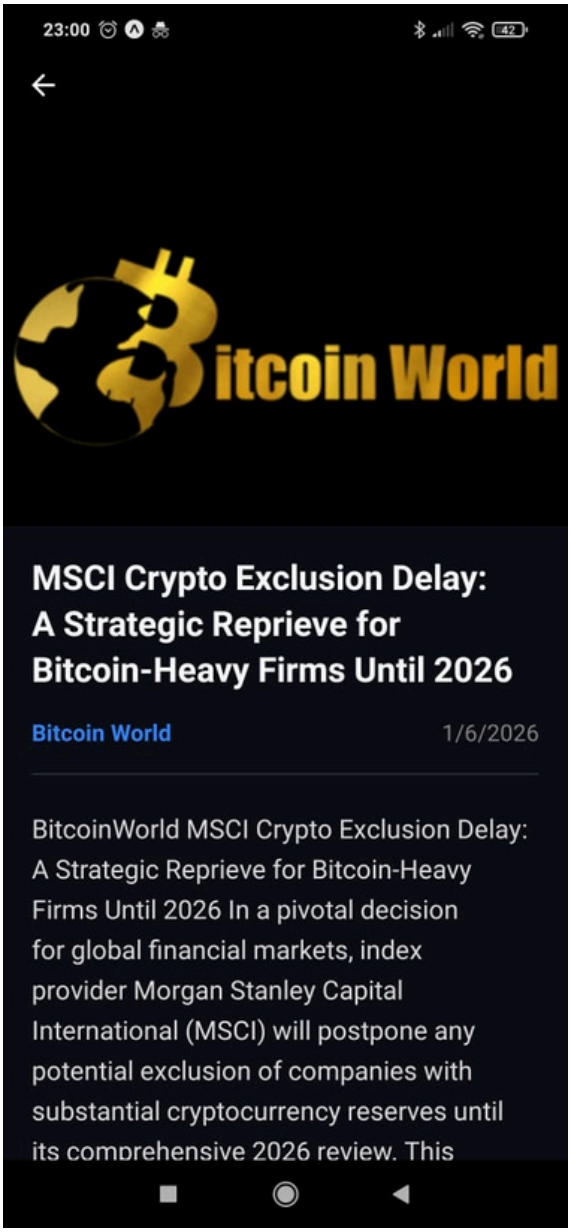

Figure 3.2 MSCI Crypto

Figure3.3–ETH, XRP,SOL,ADA

Éléments affichés :

— Image en-tête de l'article
— Titre et source
— Date de publication
— Contenu complet formaté
— Bouton de partage

3.1.3 Écran Market - Données de Marché

L'écran Market présente les données en temps réel des principales cryptomonnaies.

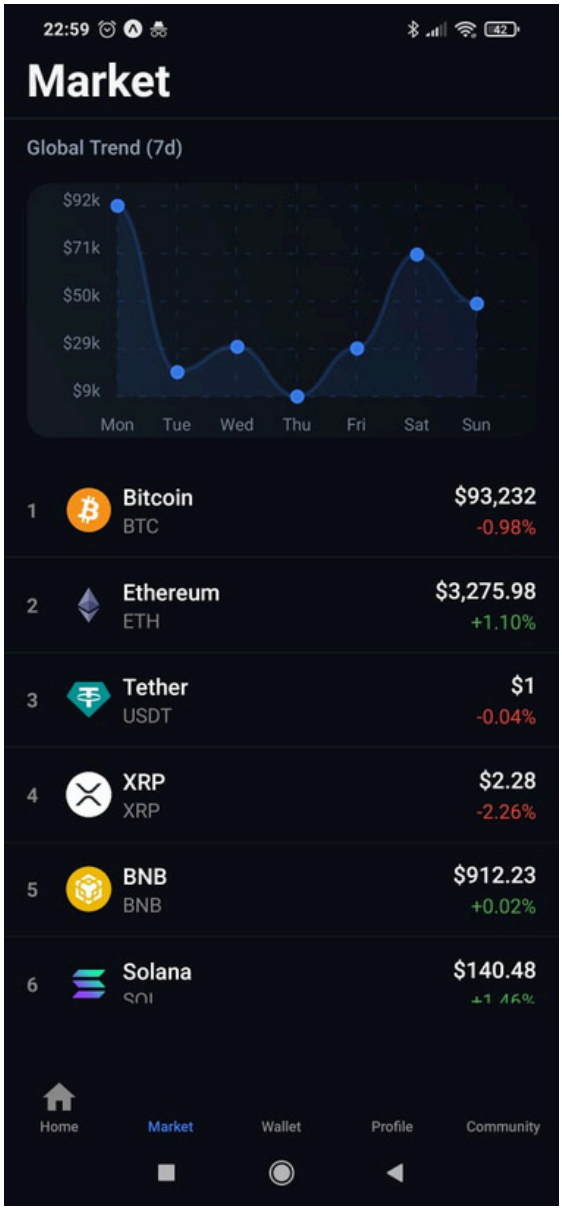

Figure 3.4 – Écran Market avec graphique de tendance

Informations affichées :

— Top 5 cryptomonnaies par capitalisation
— Prix actuel en USD
— Variation sur 24h (codée par couleur)

— Graphique de tendance sur 7 jours
— Logo et symbole de chaque crypto

3.1.4 GestiondePortefeuille

Le wallet permet aux utilisateurs de suivre leurs investissements avec valorisation en temps réel.

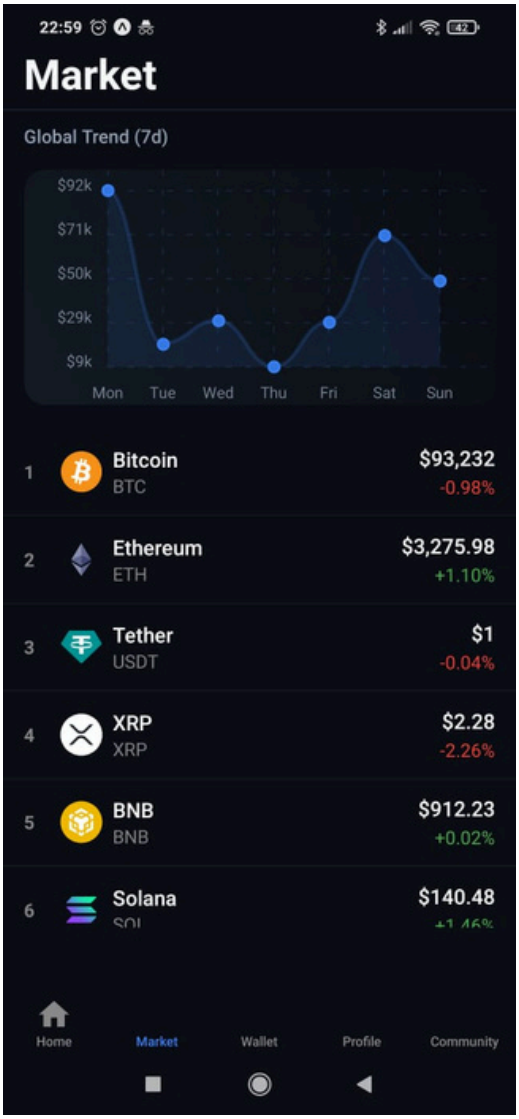

Figure 3.5 – Écran Wallet - Gestion de portefeuille

Fonctionnalités du portefeuille :

— Affichage du solde total valorisé
— Saisie de quantités pour chaque crypto
— Calcul automatique de la valeur
— Sauvegarde cloud via Firebase
— Mise à jour des prix en temps réel

Le calcul de la valeur totale suit la formule :

$$\text{Valeur}_{\text{Total}} = \sum_{i=1}^{n} (\text{quantité}_i \times \text{prix}_{\text{actuel}}) \quad (3.1)$$

3.1.5 Profil Utilisateur et Paramètres

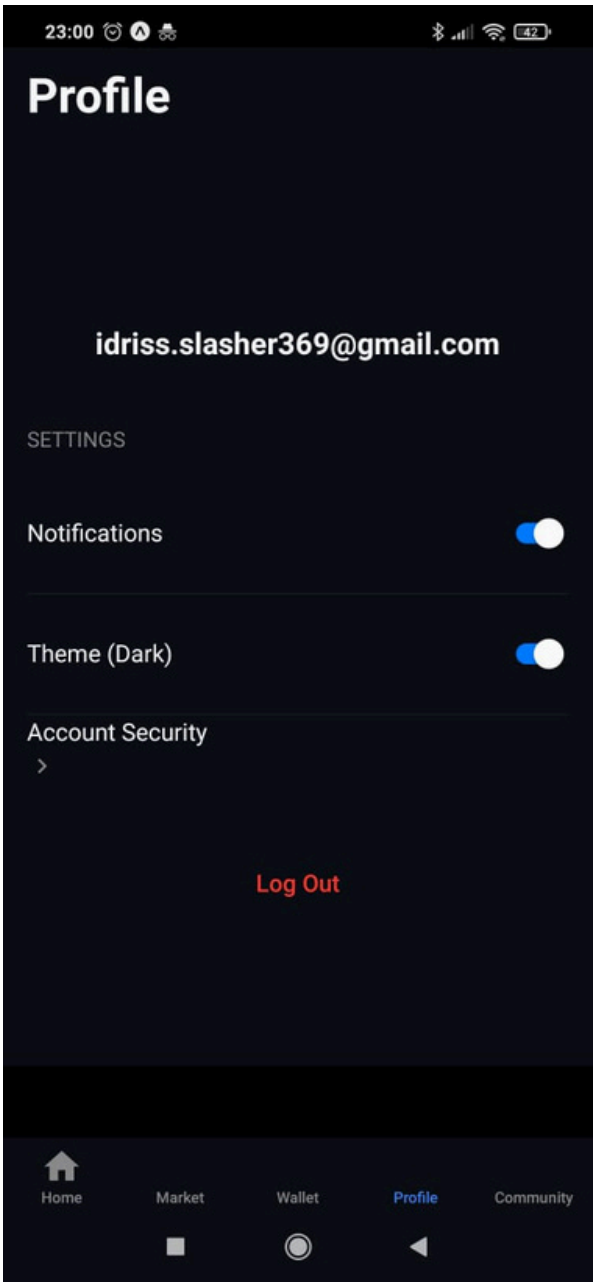

Figure 3.6 – Écran Profile avec paramètres

Options disponibles :

— Affichage de l'email utilisateur
— Toggle notifications
— Basculement thème Clair/Sombre
— Accès aux paramètres de sécurité
— Bouton de déconnexion

3.1.6 Sécurité du Compte

L'écran de sécurité permet la gestion sécurisée du mot de passe.

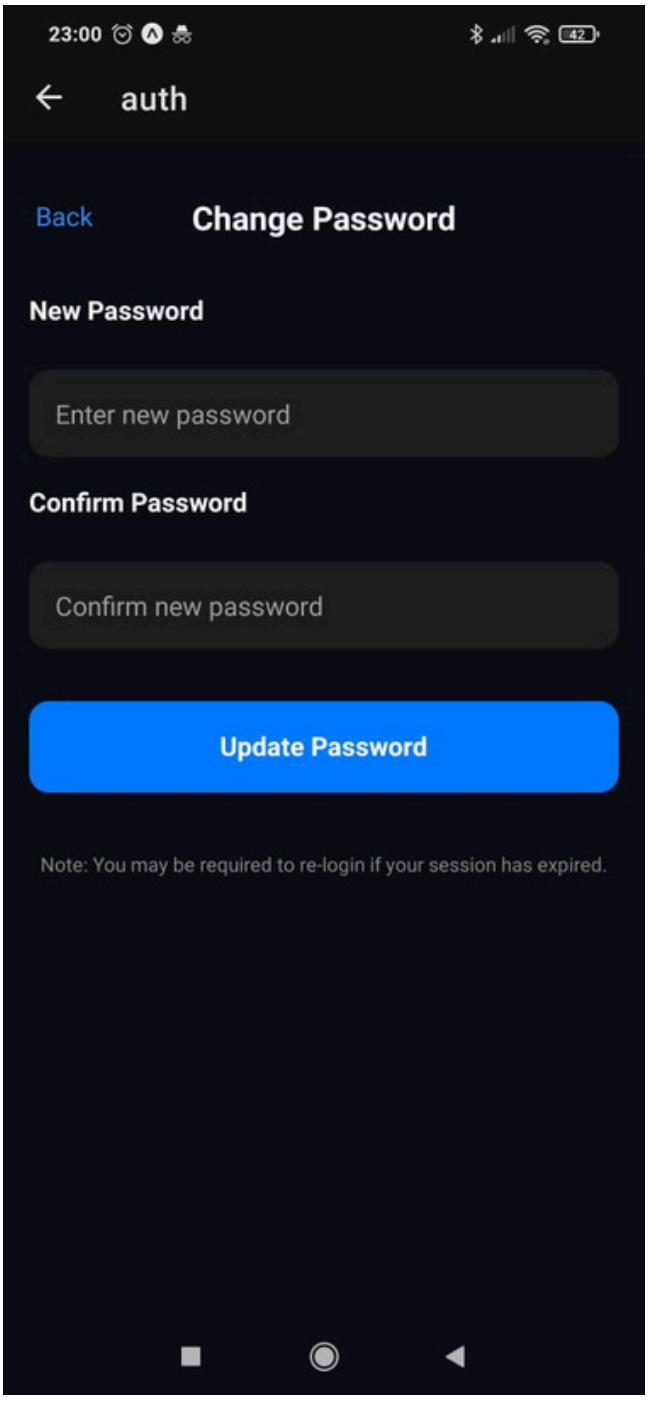

Figure 3.7 – Écran de changement de mot de passe

Mesures de sécurité :

— Validation de longueur minimale (6 caractères)
— Confirmation du nouveau mot de passe
— Vérification d'authentification récente
— Saisie sécurisée avec texte masqué

3.1.7 ChatCommunautaire

Le module Community offre un espace d'échange en temps réel entre utilisateurs.

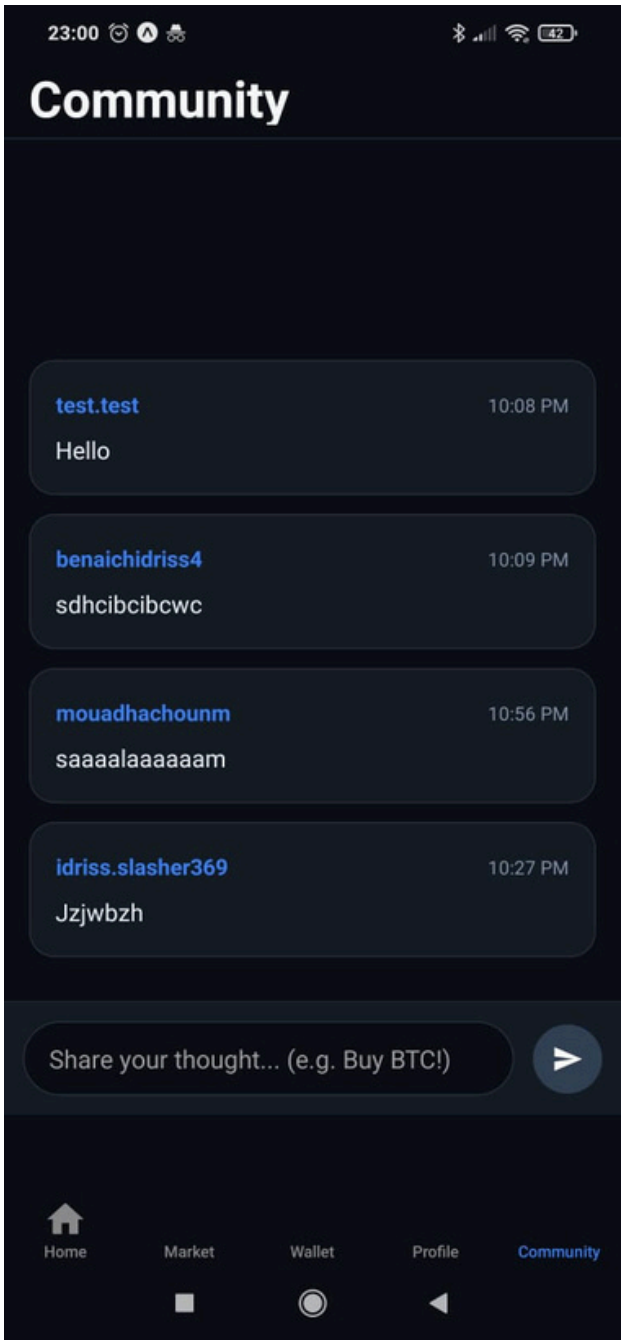

Figure 3.8 – Chat communautaire en temps réel

Caractéristiques du chat :

— Messages synchronisés en temps réel
— Affichage de l'auteur et timestamp
— Champ de saisie avec envoi rapide
— Défilement automatique vers nouveaux messages
— Authentification requise pour participer

3.2 Fonctionnalités Principales Implémentées

3.2.1 Système d'Authentification

Implémentation complète avec Firebase Authentication :

Listing 3.1 – Configuration Firebase Auth

```
import { initializeAuth, getReactNativePersistence} from
    firebase/auth';
import AsyncStoragefrom '@react-native-async-storage/async-
    storage ';

let auth;
if (Platform.OS === 'web') {
     auth = getAuth(app);
} else {
     auth = initializeAuth(app, {
         persistence: getReactNativePersistence(AsyncStorage)
     });
}
```

Fonctionnalités d'authentification :

— Inscription avec email/mot de passe

,

— Connexion sécurisée
— Persistence de session cross-platform
— Changement de mot de passe
— Déconnexion avec nettoyage de session

3.2.2 ChatenTempsRéel

Utilisation de Firestore avec listeners temps réel :

Listing 3.2 – Listener messages temps réel

```
useEffect(() => {
    const q = query(
        collection(db, 'messages'),
        orderBy('createdAt', 'desc')
    );

    const unsubscribe = onSnapshot(q, (snapshot) => {
        const msgs= snapshot.docs.map(doc=> ({
            id: doc.id,
            ...doc.data()
        }));
        setMessages(msgs);
    });

    return () => unsubscribe();
}, []);
```

3.2.3 GestiondePortefeuille

Calcul et persistence de la valorisation :

Listing 3.3 – Calcul valorisation portefeuille

```
const calculateTotal = () => {
    let total = 0;
    SUPPORTED_COINS.forEach(coin=> {
        cons qty = parseFloat(holdings[coin.id] || '0');
        t    price = prices[coin.id] || 0;
        ifons(!isNaN(qty)) {
        t   total += qty * price;
        }
    });
    return total;
};
```

3.2.4 Système de Thème

Implémentation du basculement thème avec persistence :

Listing 3.4 – ThemeContext

```
export const ThemeProvider = ({ children }) => {
    const [theme, setTheme] = useState('dark');

    const toggleTheme = async () => {
        const newTheme = theme === 'dark' ? 'light' : 'dark';
        setTheme(newTheme);
        await AsyncStorage.setItem('theme', newTheme);
    };

    return (
        <ThemeContext.Provider value={{ theme, toggleTheme }}>
            {children}
        </ThemeContext.Provider>
    );
};
```

3.3 DéfisTechniquesRencontrés

3.3.1 Persistence Cross-Platform

Problème : Différences de comportement entre web et mobile pour Firebase Auth.
Solution : Initialisation conditionnelle selon la plateforme avec AsyncStorage pour mobile et stockage navigateur pour web.

3.3.2 SynchronisationTempsRéel

Problème : Gestion des listeners Firestore et prévention des fuites mémoire.
Solution : Utilisation correcte de useEffect avec cleanup function pour désinscrire les listeners au démontage des composants.

3.3.3 PerformanceduCalculdePortefeuille

Problème : Recalculs fréquents impactant les performances.
Solution : Optimisation avec calculs mémorisés et mises à jour conditionnelles basées sur les changements réels de données.

3.3.4 GestionduClavierMobile

Problème : Le clavier masquait les champs de saisie sur mobile.
Solution : Implémentation de KeyboardAvoidingView avec comportement adapté à chaque plateforme.

Chapitre 4

Technologies Utilisées

4.1 Framework et Outils de Développement

4.1.1

React Native

Version : Latest stable

Avantages:

— Développement cross-platform (iOS, Android, Web)
— Base de code unique
— Performances natives
— Large écosystème de bibliothèques
— Hot reloading pour développement rapide

4.1.2 Expo

Version : SDK 51+

Fonctionnalités utilisées :

— Expo Router pour la navigation
— APIs natives pré-configurées
— Build et déploiement simplifiés
— Expo Image pour optimisation d'images
— Expo Constants pour configuration

4.1.3 Expo Router

Caractéristiques :

— Routage basé sur les fichiers
— Navigation par onglets
— Stack navigation
— Modales
— Deep linking automatique

4.2 Services Backend

4.2.1 Firebase

Services utilisés :

1. Firebase Authentication
   — Méthode : Email/Password
   — Gestion de sessions
   — Persistence sécurisée
   — Ré-authentification pour opérations sensibles
2. Firebase Firestore
   — Base de données NoSQL cloud
   — Synchronisation temps réel
   — Collections : users, messages
   — Requêtes avec orderBy et where
   — Listeners en temps réel

4.2.2 Configuration Firebase

Listing 4.1 – firebase.js

```
impor { initializeApp } from 'firebase/app';
t { getFirestore } from 'firebase/firestore';
impor { initializeAuth } from 'firebase/auth';
t
const firebaseConfig = {
t apiKey: "...",
    authDomain:"...",
    projectId: "...",
    storageBucket: "...",
    messagingSenderId:"...",
    appId: "..."
};

const app = initializeApp(firebaseConfig);
expor cons db = getFirestore(app);
t t auth = initializeAuth(app, { /* ... */ });
expor cons
t t
```

4.3 Gestiond'État

4.3.1 React Context API

Contexts implémentés :

1. UserContext
   — État d'authentification global
   — Fonctions : login, register, logout, changePassword
   — Surveillance de l'état utilisateur

2. ThemeContext
   — Gestion du thème clair/sombre
   — Persistence des préférences
   — Application globale du thème

4.3.2 React Hooks

Hooks utilisés :
— useState - État local des composants
— useEffect - Effets de bord et cleanup
— useContext - Accès aux contexts globaux
— useCallback - Mémorisation de fonctions

4.4 Bibliothèques et Composants UI

4.4.1 Visualisation de Données

react-native-chart-kit
— Graphiques linéaires pour tendances
— Configuration personnalisable
— Gradients et couleurs
— Support des données temporelles

4.4.2 Système d'Icônes

Lucide React Native
— Bibliothèque d'icônes moderne
— Personnalisation taille et couleur
— Icônes : Home, TrendingUp, Wallet, MessageSquare, User, etc.

4.4.3 Gestiond'Images

Expo Image
— Chargement optimisé
— Cache automatique
— Placeholders
— Support formats multiples

4.5 Stockage de Données

4.5.1 AsyncStorage

Utilisation :
— Persistence du thème utilisateur
— Cache de données temporaires
— Stockage de préférences

Listing 4.2 – Exemple AsyncStorage

```
// Sauvegarde
await AsyncStorage.setItem('theme', 'dark');

// Récupération
const theme = await AsyncStorage.getItem('theme');
```

4.5.2 Firebase Firestore

Opérations CRUD :
— Create : addDoc, setDoc
— Read : getDoc, getDocs, onSnapshot
— Update : updateDoc
— Delete : deleteDoc

4.6 Outils de Développement

4.6.1 Environnement de Développement

— Node.js : v18+
— npm/yarn : Gestionnaire de paquets
— Expo CLI : Outils de développement
— VS Code : Éditeur recommandé

4.6.2 Extensions VS Code Utiles

— ES7+ React/Redux/React-Native snippets
— Prettier - Code formatter
— ESLint
— React Native Tools

4.6.3 Debugging

— React Native Debugger
— Chrome DevTools (pour web)
— Expo DevTools
— Console.log pour traces

4.7 APIs Externes

4.7.1 API Crypto (Mock)

Services simulés :
— Prix des cryptomonnaies
— Variations 24h
— Données de graphiques
— Capitalisation boursière

Note : Pour une version production, intégrer CoinGecko API ou similaire.

4.7.2 APINews(Mock)

Données simulées :
— Articles d'actualité crypto
— Images, titres, sources
— Dates de publication
— Contenu des articles
Note : Pour production, utiliser CryptoPanic API ou NewsAPI.

4.8 ArchitectureTechniqueComplète

Couche	Technologies
Frontend	ReactNative,Expo
Navigation	ExpoRouter
ÉtatGlobal	ContextAPI,Hooks
Backend	Firebase(Auth+Firestore)
StockageLocal	AsyncStorage
UIComponents	Custom+LucideIcons
Charts	react-native-chart-kit

Figure 4.1 – Stack technologique complète

Conclusion

Synthèse

Le projet CryptoNews a permis de développer une application mobile cross-platform complète et fonctionnelle. L'application répond aux objectifs fixés en offrant :

— Une interface utilisateur moderne et intuitive
— Des fonctionnalités de suivi des cryptomonnaies en temps réel
— Une gestion de portefeuille avec valorisation automatique
— Un système de chat communautaire synchronisé
— Une authentification sécurisée avec Firebase

Compétences Acquises

Ce projet a permis de maîtriser :

1. Le développement React Native et Expo
2. L'intégration de services Firebase
3. La gestion d'état avec Context API
4. L'architecture d'applications mobiles
5. Les patterns de développement modernes

Perspectives d'Évolution

Améliorations possibles :

— Intégration d'APIs réelles (CoinGecko, CryptoPanic)
— Système d'alertes de prix
— Graphiques avancés avec indicateurs techniques
— Fonctionnalités sociales étendues
— Mode hors ligne avec cache
— Tests automatisés (Jest, Detox)

Conclusion Finale

CryptoNews démontre l'application pratique de technologies modernes pour créer une solution mobile complète. Le projet constitue une base solide pour comprendre le dévelop- pement d'applications cross-platform et l'intégration de services cloud. Les compétences acquises sont directement applicables en environnement professionnel.

Bibliographie

[1] Meta Platforms, Inc. (2024).React Native Documentation. https://reactnative.dev/docs/getting- started

[2] Expo. (2024). 1 :41textitExpo Documentation. https://docs.expo.dev/

[3] Google LLC. (2024). Firebase Documentation. https://firebase.google.com/docs

[4]Expo.(2024). Expo Router Documentation . https://expo.github.io/router/docs/

Annexe A

Installation et Configuration

A.1 Prérequis

— Node.js v18+
— npm ou yarn
— Compte Firebase
— Expo CLI
— Émulateur Android ou Simulateur iOS (optionnel)

A.2 Installation

Listing A.1 – Commandes d'installation

```
Installer Expo CLI
npm install -g expo-cli
Cloner le projet
git clone [url-du-depot]
cd cryptonews-app
Installer les d pendances
npm install
D marrer le serveur de d veloppement
npx expo start
```

A.3 Configuration Firebase

1. Créer un projet Firebase
2. Activer Authentication (Email/Password)
3. Créer une base Firestore
4. Copier la configuration dans `src/config/firebase.js`