Équipe EOS Marketplace

EOS Marketplace V1
Document d'architecture logicielle

Version 1.0

Historique des révisions

Date	Version	Description	Auteur
aaaa-mm-jj	x.x	<Détails précis du travail effectué>	<Nom>
2021-09-27	0.1	Ajout introduction et cas d'utilisation	Guilhem Dubois
2021-10-06	0.2	Ajout des diagramme de paquetage	Arthur Garnier
2021-10-08	1.0	Complétion et finalisation du document d'architecture logicielle	Équipe

Table des matières

Document d'architecture logicielle

1. Introduction

Ce document sert à décrire l'architecture de l'application EOS Marketplace selon différents angles. Pour commencer, les contraintes pouvant avoir une influence sur notre choix d'architecture seront présentées, tel que la sécurité, le budget, etc. Ensuite, les principales fonctionnalités seront représentées par des diagrammes de cas d'utilisation. La troisième partie porte sur la vue logique, comportant différents diagrammes de paquetage pour démontrer la structure logicielle de l'application cliente et du serveur. La quatrième partie porte sur la vue des processus. Elle contient différents diagrammes de séquence afin de mieux comprendre les interactions entre les différentes parties du projet dans certains scénarios. Ensuite, la vue de déploiement représente les composants physiques du système et leurs connexions. Finalement, la dernière section indique les critères de taille et de performance du système.

2. Objectifs et contraintes architecturaux

Sécurité: La sécurité est un objectif primordial pour notre application. Il faut s'assurer de protéger l'intégrité des données personnelles des utilisateurs. Il sera nécessaire d'encrypter les données tel que les mots de passe et identifiants des Anchor Wallet dans la base de données.

Réutilisation: La réutilisation est un aspect important à prendre en compte lors du développement du projet puisqu'il sera repris par une autre équipe. Il faudra donc s'assurer que la structure du code soit claire et ajouter des commentaires pour faciliter la transition entre les équipes qui travaillent sur l'application.

Modularité: L'architecture du projet doit être modulaire afin de permettre la parallélisation du développement. Ainsi, tous les membres de l'équipe pourront développer tout en étant indépendants les uns des autres.

Portabilité: Il est nécessaire que l'application soit portable afin de permettre son utilisation sur IOS et Android. Il faudra donc prendre en compte les deux plateformes dans la structure et le développement de l'application.

Échéancier: L'objectif est d'avoir compléter un prototype pour le 8 octobre ainsi qu'une version Alpha pour le 7 décembre.

Outil de développement: Nous allons utiliser Visual Code pour développer le client ainsi que le serveur ainsi que node pour exécuter des scripts.

Langage de développement: Les langages utilisés sont C++ pour la génération de contrat et typescript pour le client léger et le serveur.

3. Vue des cas d’utilisation

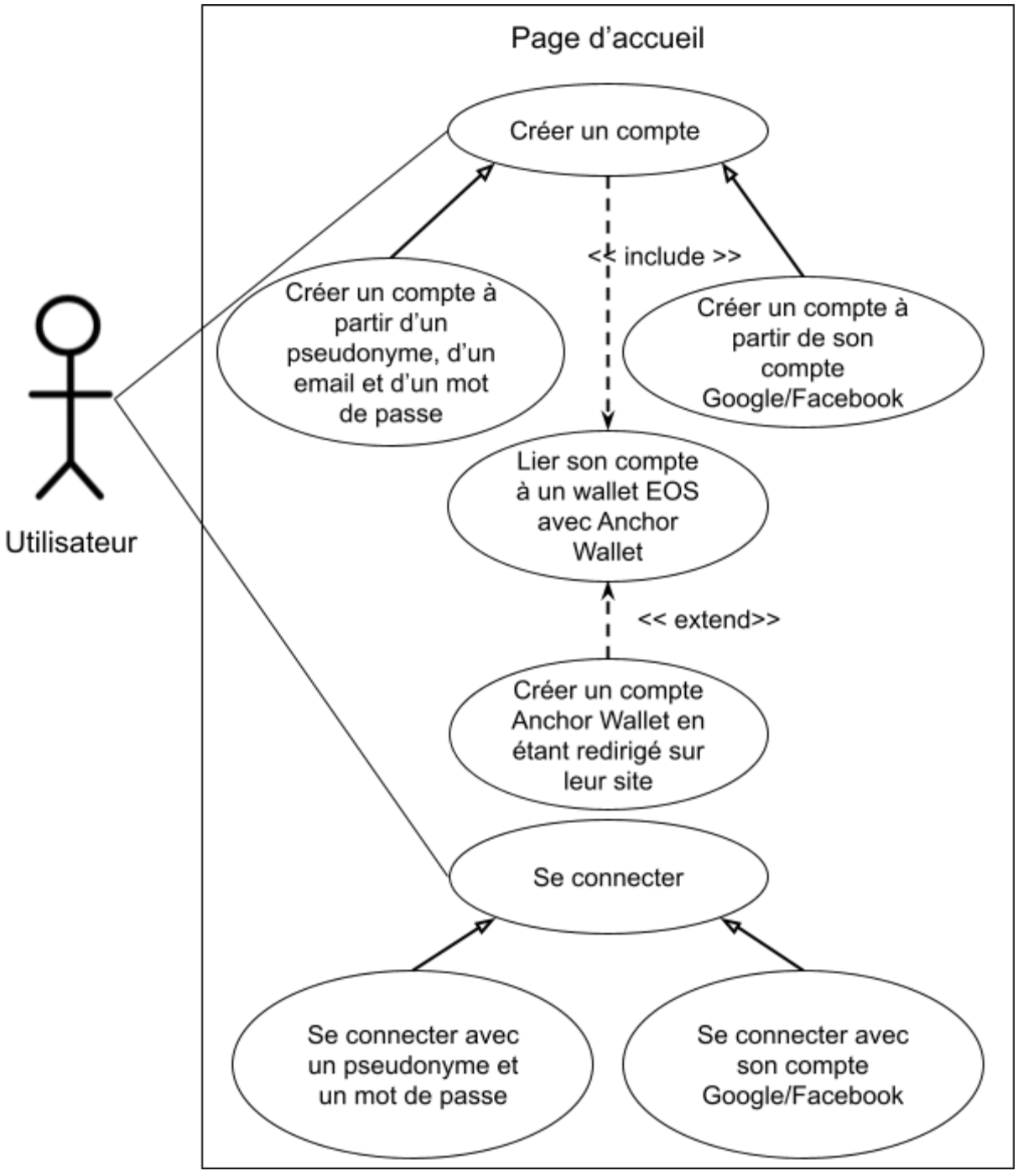


Figure 1: Diagramme des cas d’utilisation de la page d'accueil

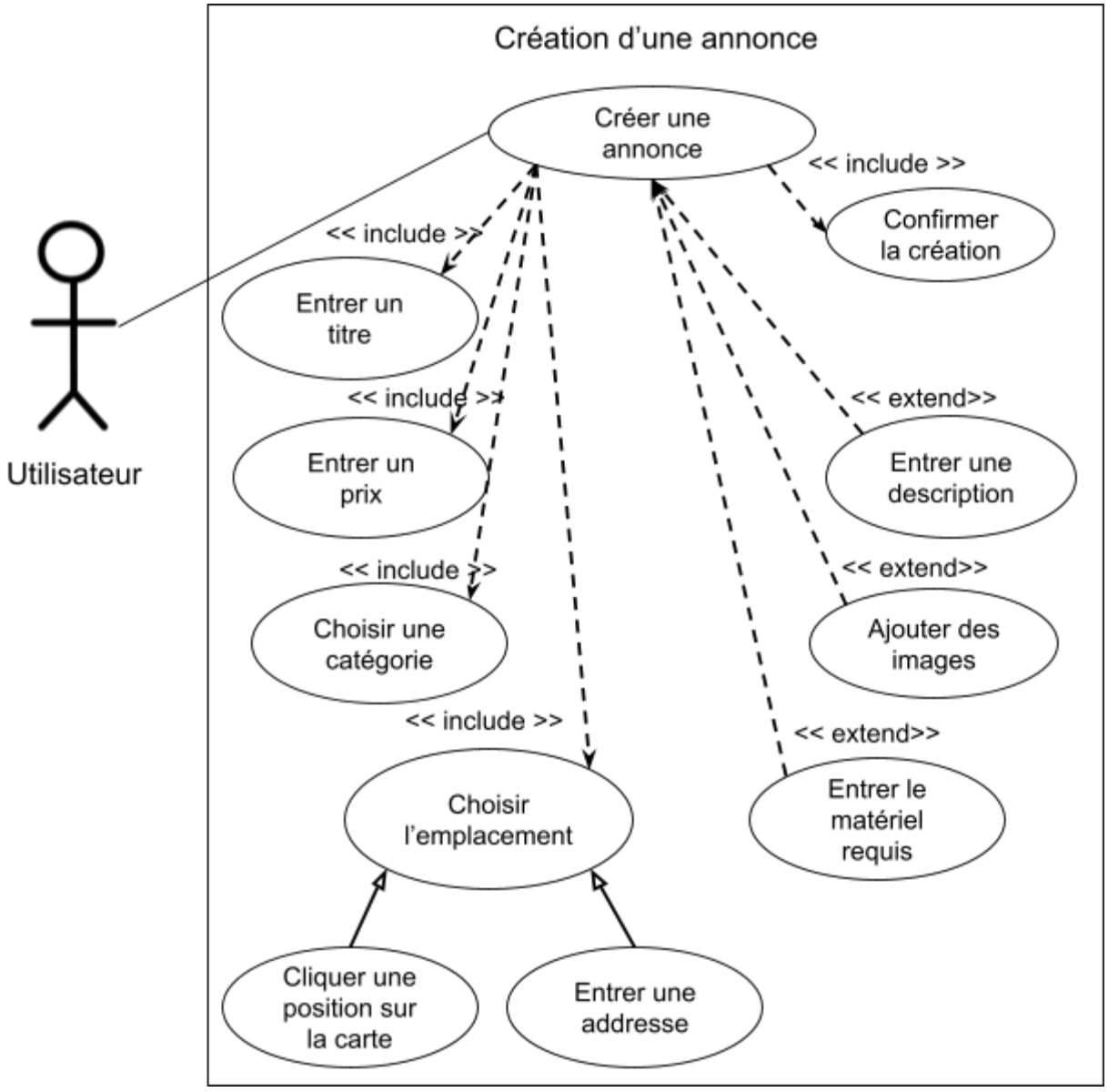


Figure 2: Diagramme des cas d'utilisation de la création d'une annonce

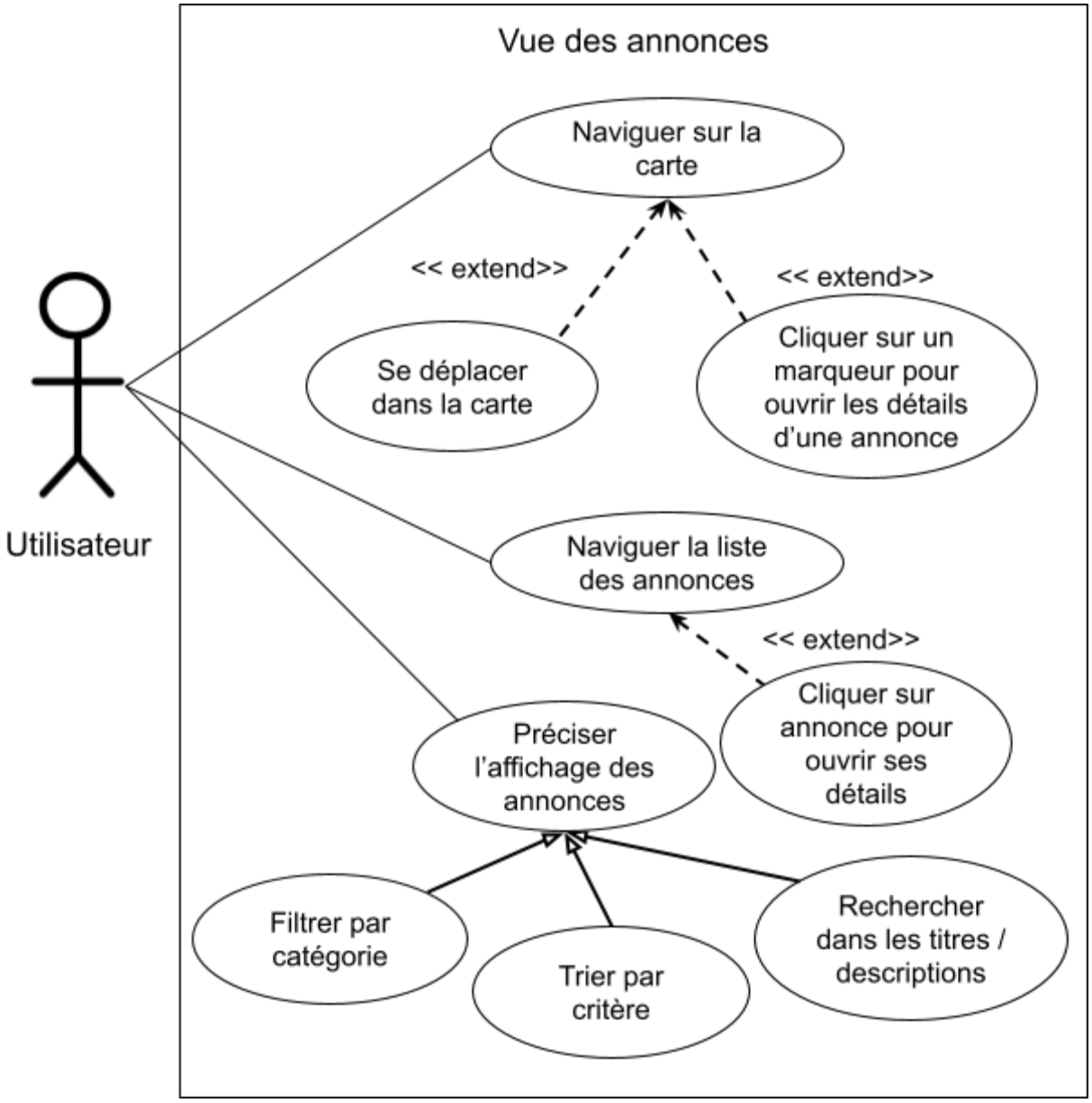


Figure 3: Diagramme des cas d'utilisation de la vue des annonces

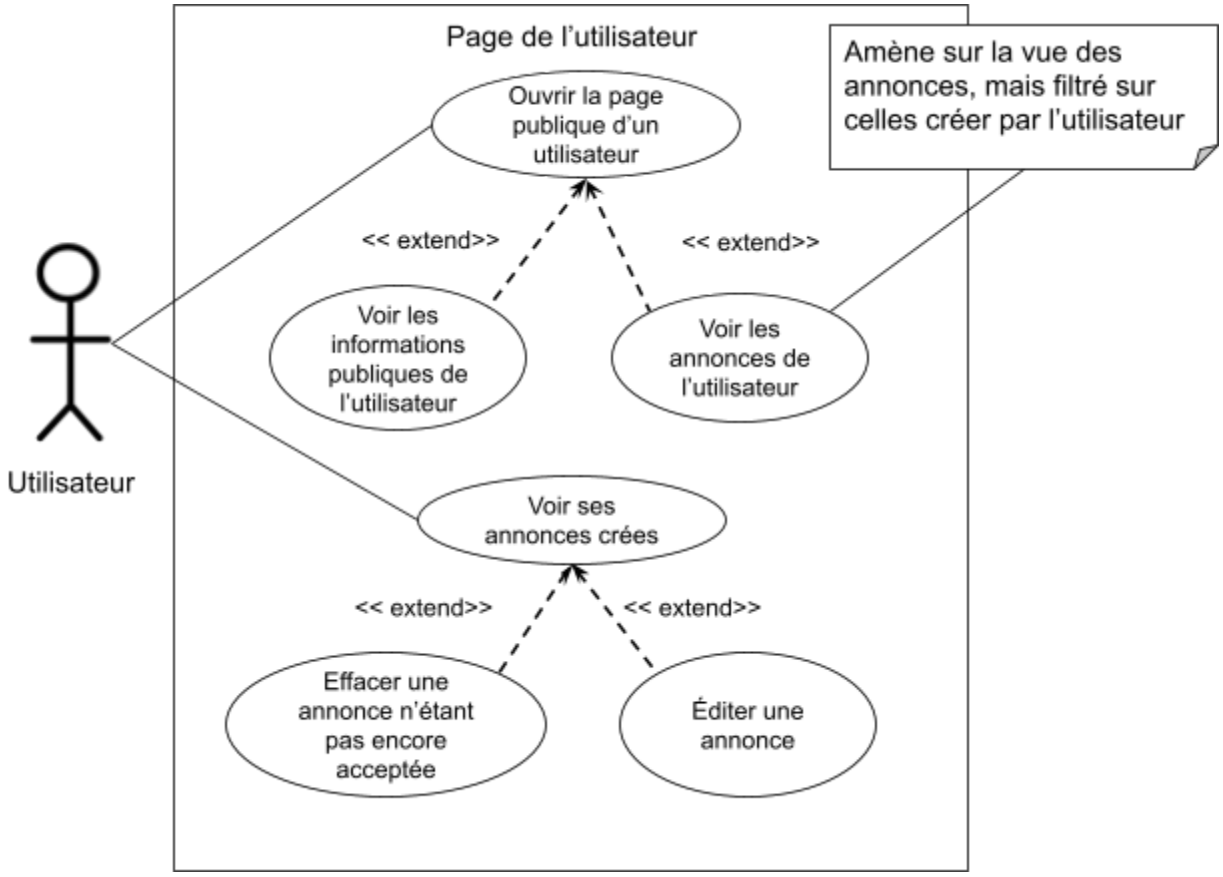


Figure 4: Diagramme des cas d'utilisation de la page d'utilisateur

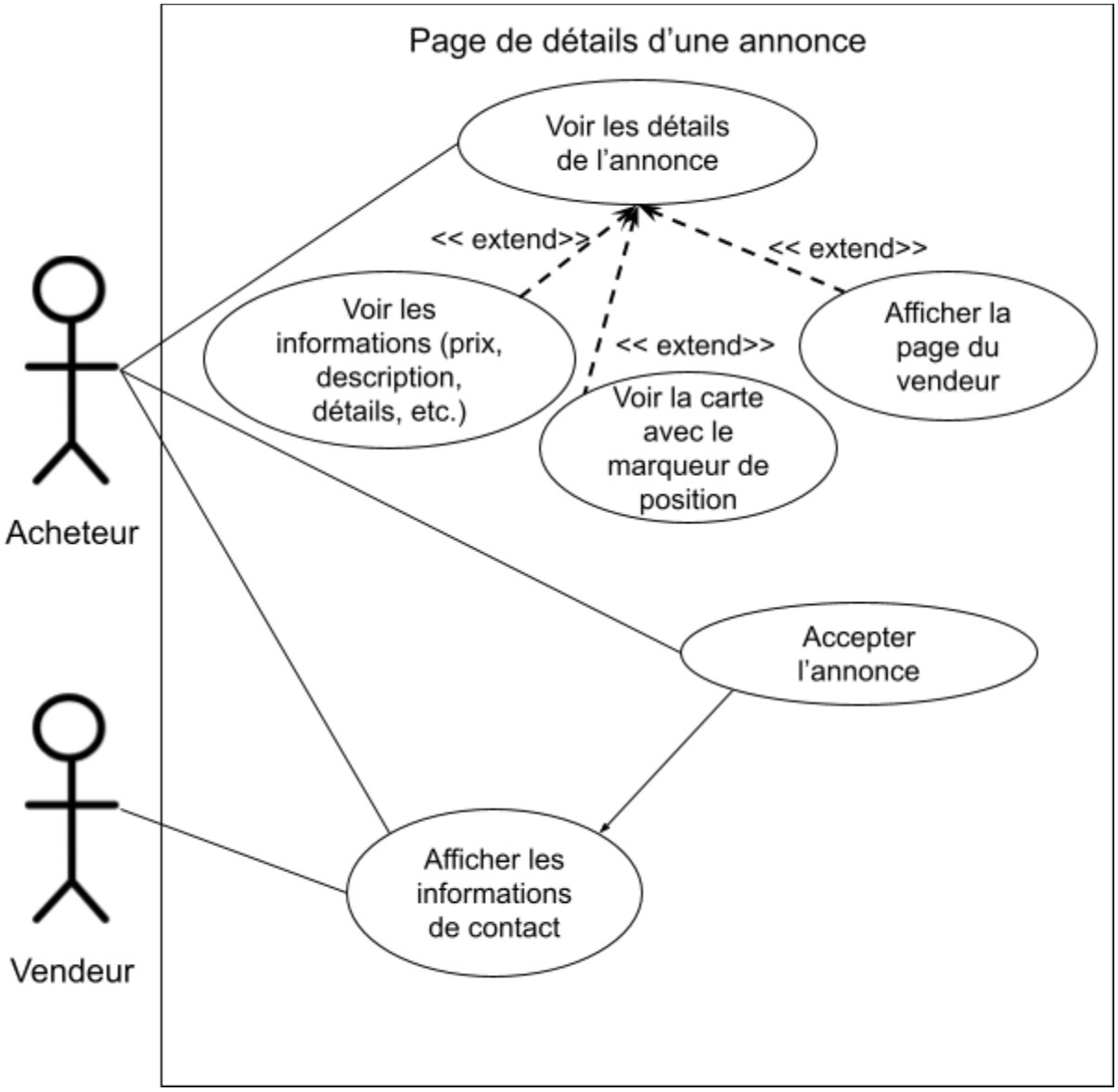


Figure 5: Diagramme des cas d'utilisation de la page de détails d'une annonce

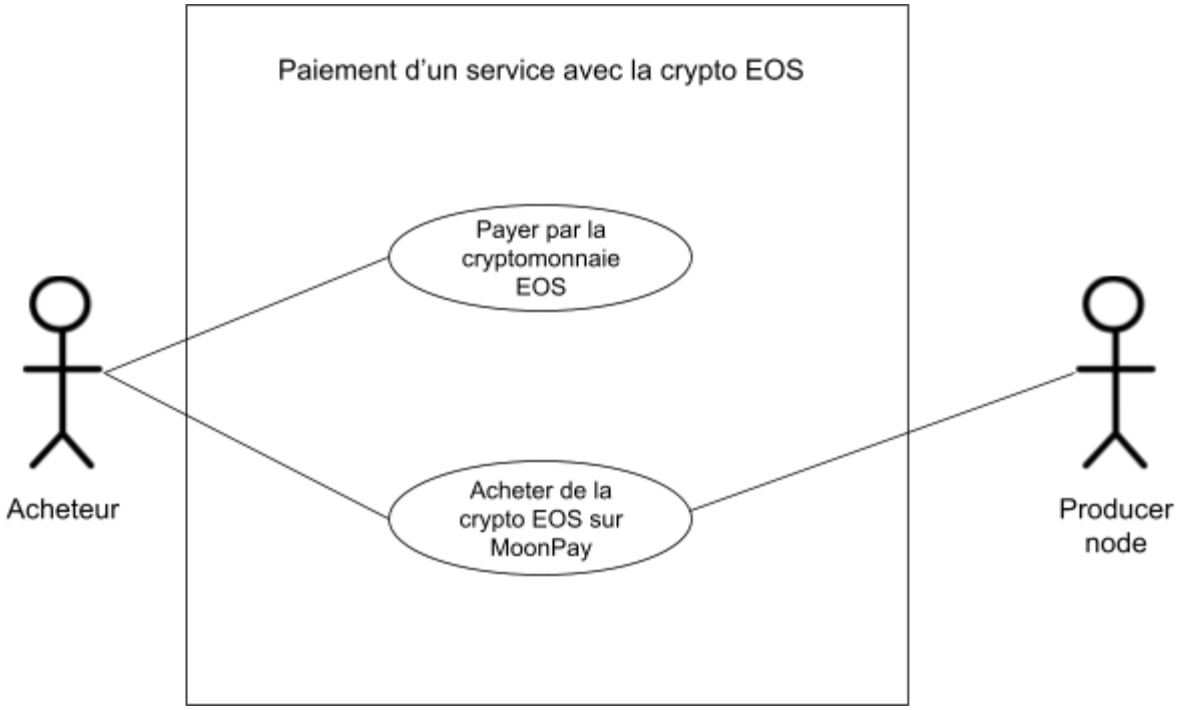
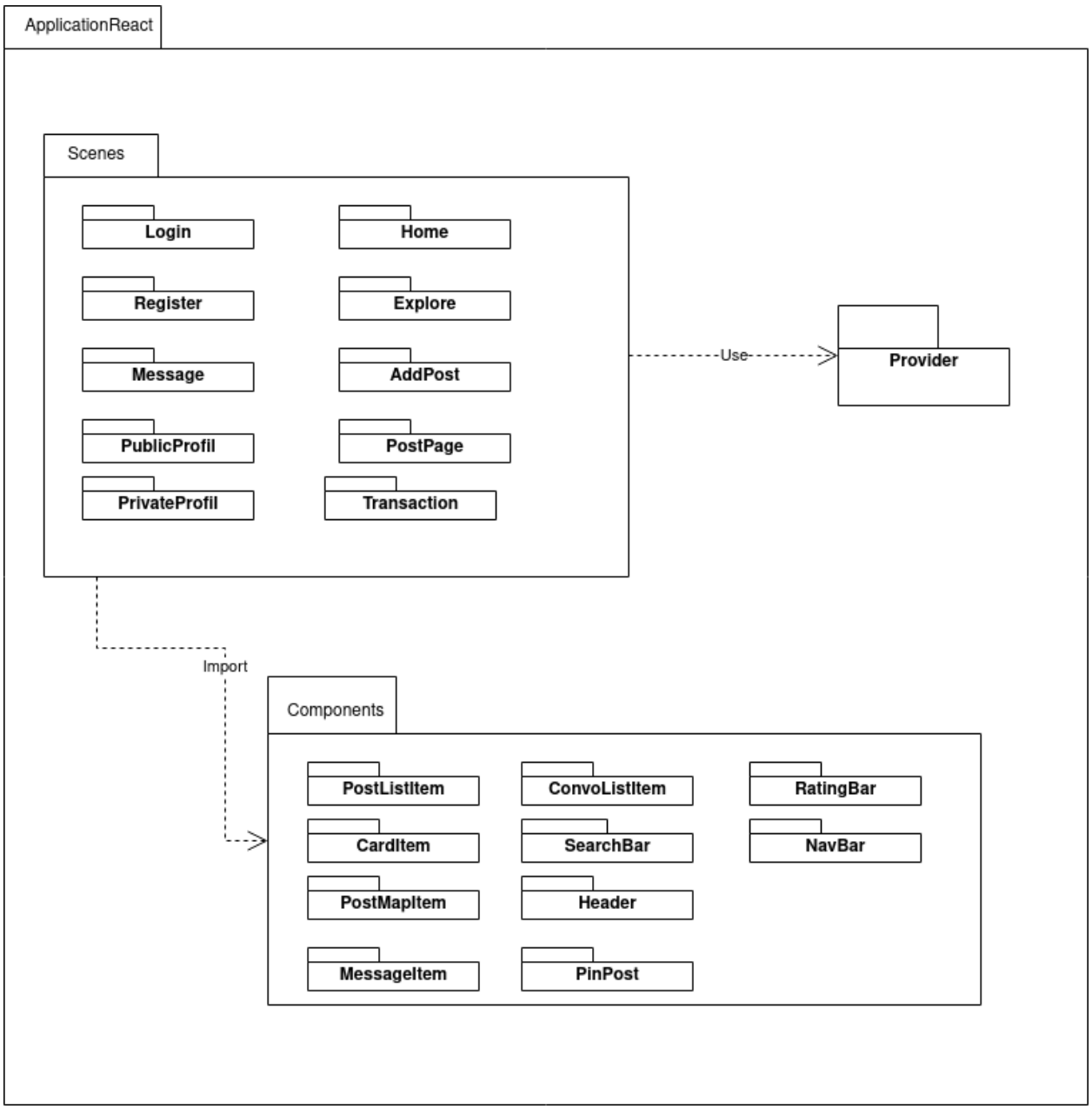


Figure 6: Diagramme des cas d'utilisation de la page de paiement

4. Vue logique

4.1 Vue

ApplicationReact
Le diagramme contient tous les modules essentielles pour l'implémentation de l'application web



Login
Le paquetage Login représente la scène qui permet l'authentification

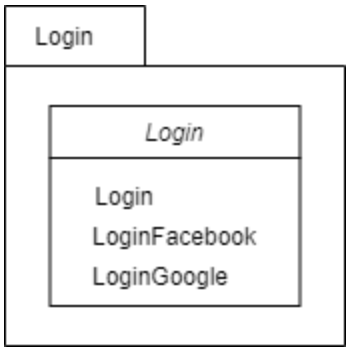


Figure 7. Diagramme de paquetage de la scène d'authentification

Register
Le paquetage Register représente la scène qui permet la création de compte et la liaison à un wallet.

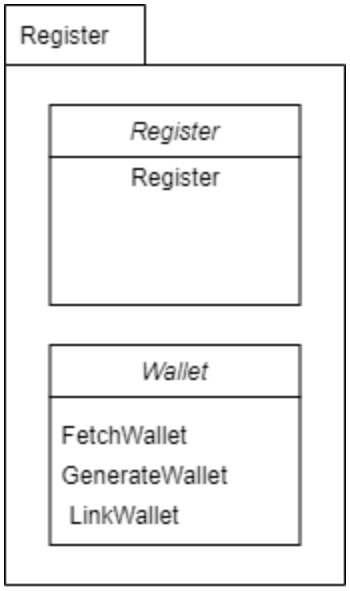


Figure 8. Diagramme de paquetage de la scène de création de compte

Home
Le paquetage Home représente la scène qui est accessible après l'authentification, présentant une carte et les différents biens/services disponibles.

Figure 9. Diagramme de paquetage de la scène Home

Explorer
Le paquetage Explorer présente la carte au complet en grand, contenant des pins indiquant des services ou des biens, ainsi que la localisation actuelle

Figure 10. Diagramme de paquetage de la scène d'exploration

AddPost
Le paquetage AddPost. définit les étapes de création d'une annonce en permettant de la localiser grâce à la carte.

Figure 11. Diagramme de paquetage de la scène pour la publication d'une annonce

Message
Le paquetage Message présente la scène permettant d'accéder à sa liste de message

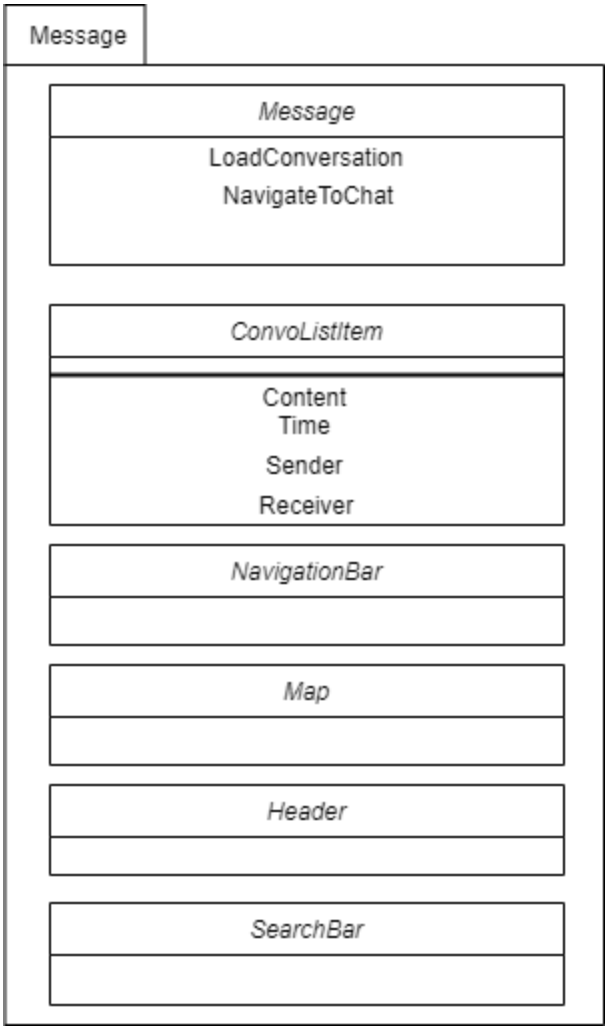


Figure 12. Diagramme du paquetage de la scène Message

Chat
Le paquetage du Chat désigne la scène permettant d'envoyer des messages et un contrat à un autre utilisateur.

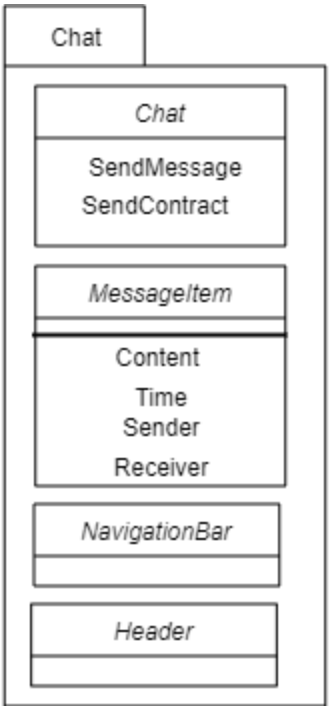


Figure 13. Diagramme de paquetage de la scène de conversation avec un utilisateur

Public Profile
Le paquetage de profile publique désigne la scène présentant le profil d'un autre utilisateur, visible par tous les utilisateurs

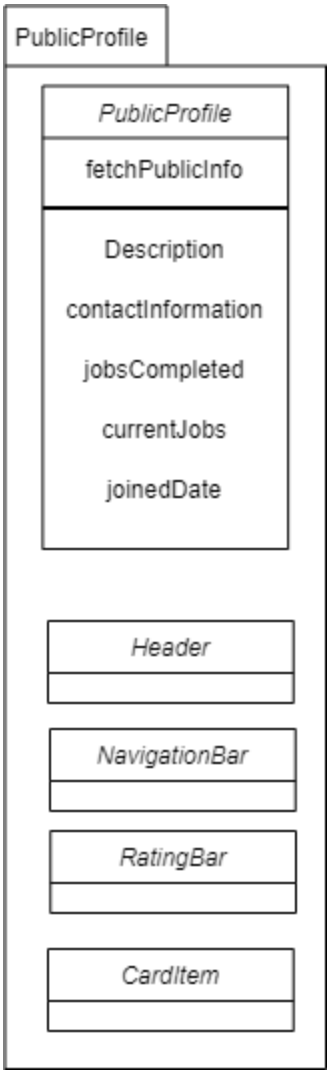


Figure 14. Diagramme de la scène du profile publique d'un utilisateur

Private Profile
Le paquetage de profile privé désigne la scène présentant le profil d'un utilisateur, uniquement visitable et éditable par lui-même

Figure 15. Diagramme de paquetage d'un profil d'utilisateur privé

PostPage
Le paquetage de la scène d'une annonce présente les différentes information contenues sur une annonce

Figure 16. Diagramme de la page d'une annonce

Transaction
Le paquetage de Transaction est issu d'un contrat et permet d'accepter mutuellement un contrat entre l'annonceur et l'intéressé.

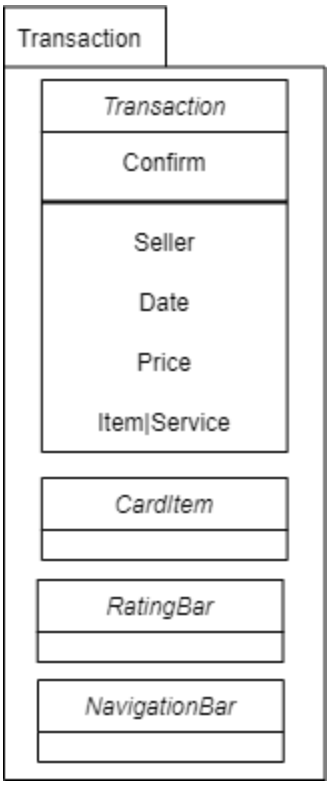


Figure 17. Diagramme de paquetage de la scène de transaction

PostListItem
Ce paquetage représente le composant d'annonce contenue sur la scène Home

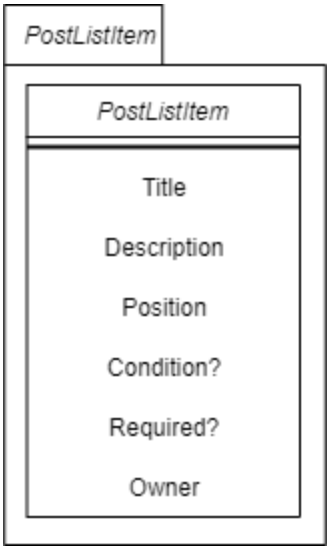


Figure 18. Diagramme de paquetage du composant d'annonce item lister sur la page d'accueil

PostMapItem
Ce paquetage représente le composant d'annonce contenue sur la scène Map

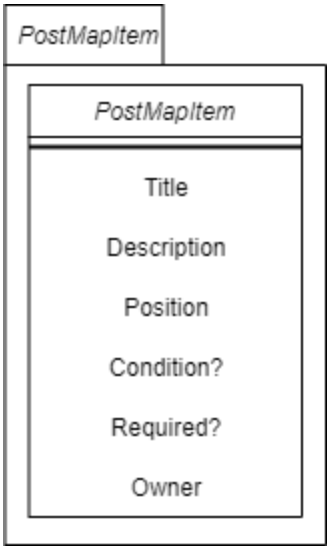


Figure 19. Diagramme de paquetage du composant d'annonce item lister sur la page de la carte

RatingBar
Ce paquetage définit le composant de notation à l'aide d'étoile utilisé dans les profils et les annonces terminées.

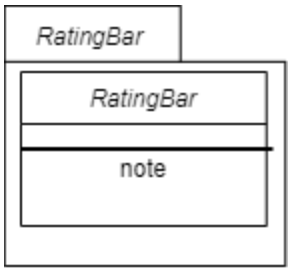


Figure 20. Diagramme de paquetage du composant de barre de notation

SearchBar
Ce paquetage représente le composant de recherche utilisé dans les vues d'accueil et d'exploration

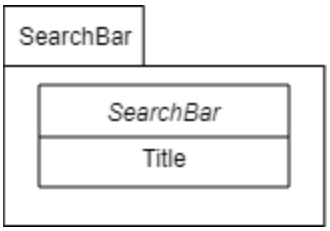


Figure 21. Diagramme de paquetage du composant de recherche

CardItem
Ce paquetage représente le composant d'annonce contenue sur la scène Map

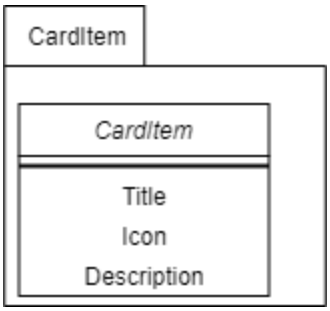


Figure 22.

Header
Ce paquetage représente le composant Header présent dans l'en-tête de l'application

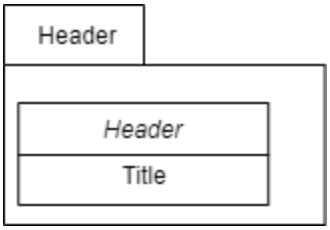


Figure 23. Diagramme de paquetage du composant header

NavigationBar
Ce paquetage représente le composant de navigation présent au pied de page de l'application

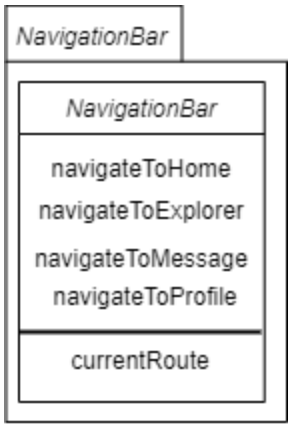


Figure 24. Diagramme de paquetage de la barre de navigation

PinPost
Ce paquetage représente le composant présent sur la carte indiquant un bien ou service.

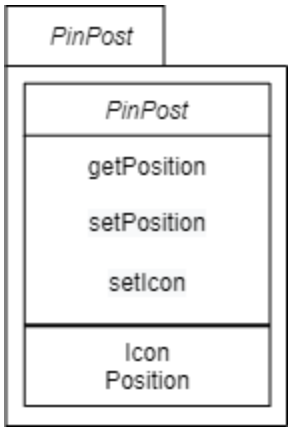
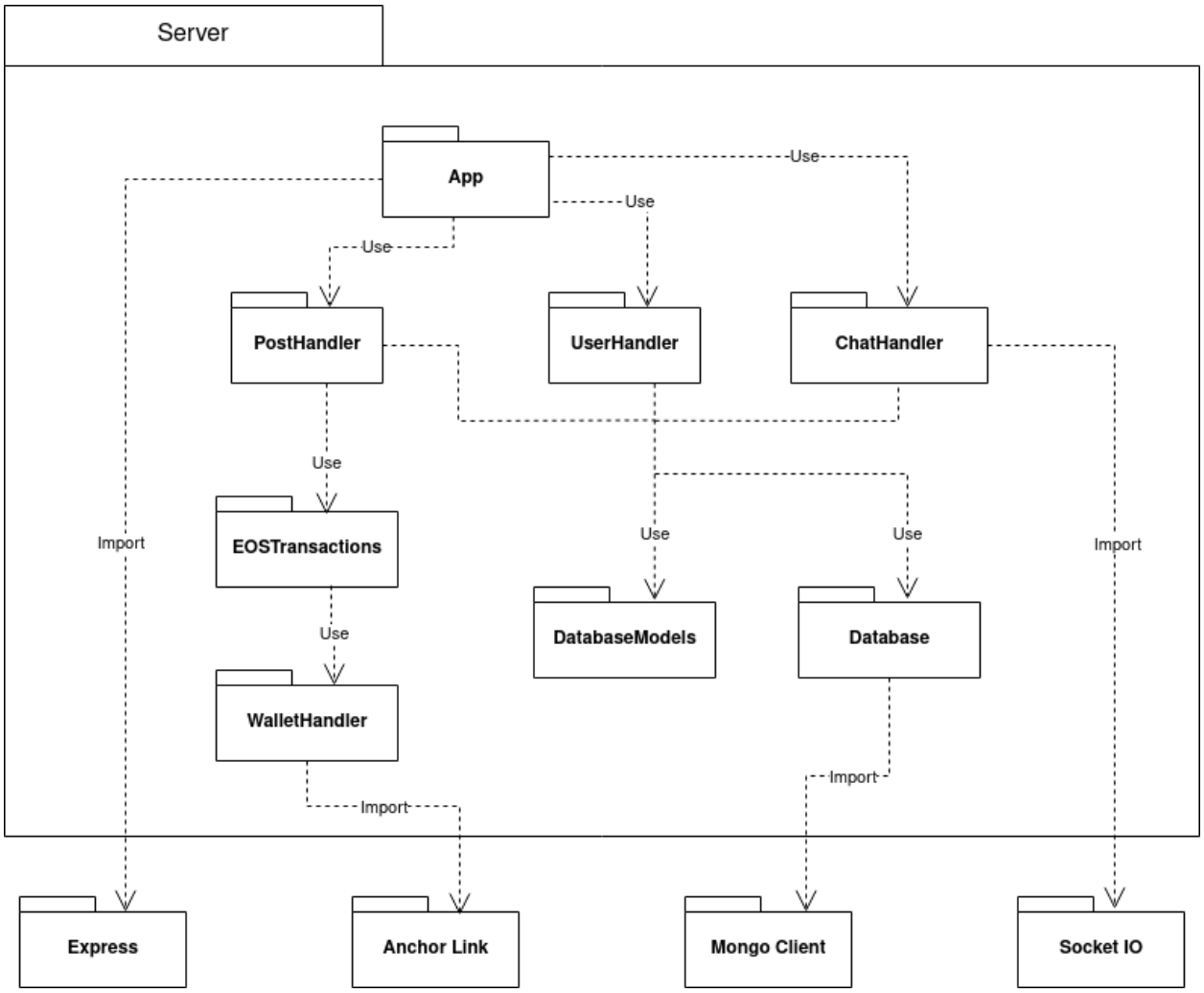


Figure 25. Diagramme de paquetage du pin de localisation d'une annonce sur la carte

Server
Le diagramme contient tous les modules essentielles pour l'implémentation du server



App
Ce paquetage contient le module de l'application de base du microservice qui roule le serveur qui accepte les requêtes des utilisateurs.

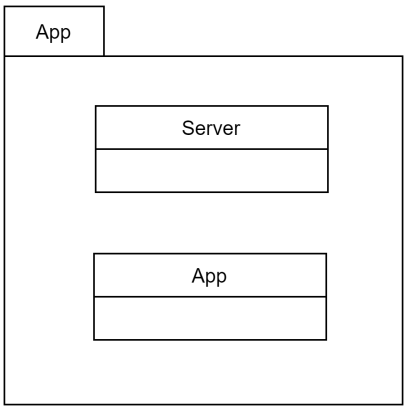


Figure 26. Diagramme de classes du paquetage App

Post Handler
Ce paquetage contient les modules nécessaires à la gestion des annonces ainsi que leurs interfaces.

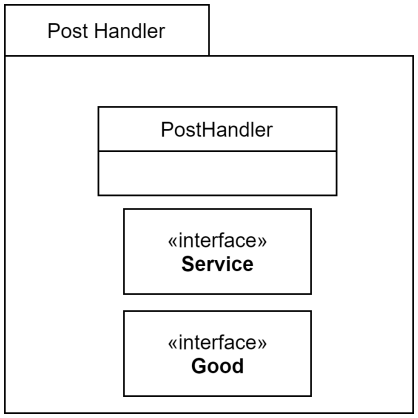


Figure 27. Diagramme de classes du paquetage Post Handler

User Handler
Ce paquetage contient les modules nécessaires à la gestion des utilisateurs ainsi que leurs interfaces.

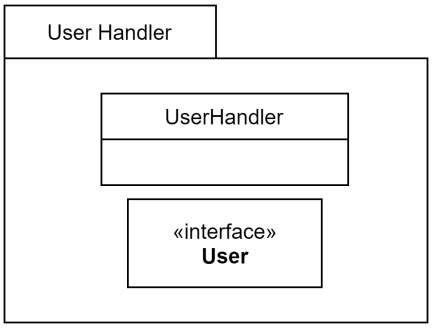


Figure 28. Diagramme de classes du paquetage User Handler

Chat Handler
Ce paquetage contient les modules qui permettent de connecter les sockets pour le système de messagerie.

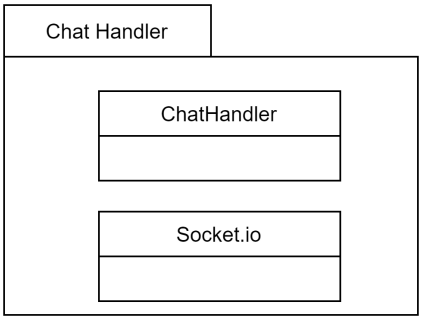


Figure 29. Diagramme de classes du paquetage Chat Handler

DatabaseModels
Ce paquetage contient les interfaces qui permettent de schématiser les données à manipuler en base de données.

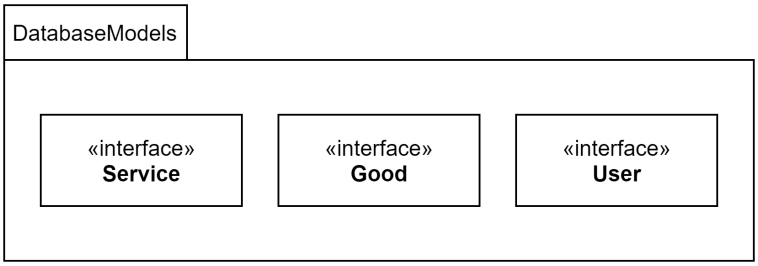


Figure 30. Diagramme de classes du paquetage DatabaseModels

Database
Ce paquetage contient tous les modules qui nous permettent de configurer nos connexions aux bases de données utilisées.

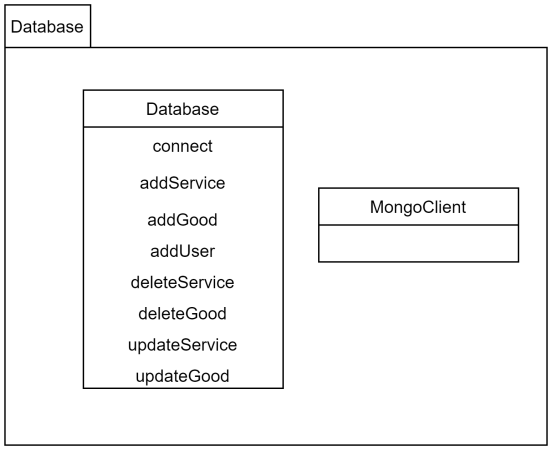


Figure 31. Diagramme de classes du paquetage Database

Wallet Handler
Ce paquetage contient les modules qui permettent de gérer les portefeuilles des utilisateurs.

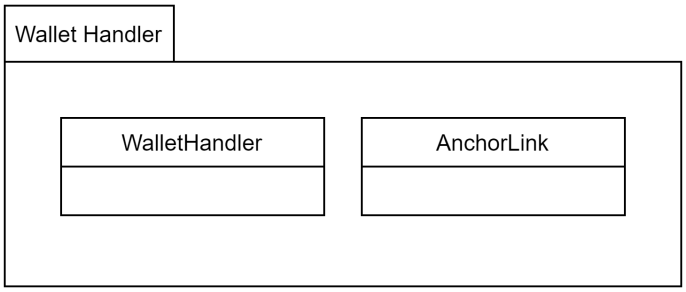


Figure 32. Diagramme de classes du paquetage Wallet Handler

EOSTransactions
Ce paquetage contient les modules qui permettent la gestion des transactions entre les parties prenantes d'un contrat.

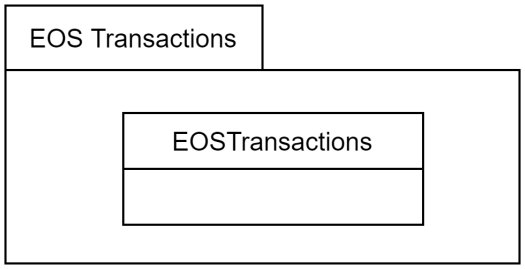


Figure 33. Diagramme de classes du paquetage EOSTransactions.

Express
Ce paquetage représente le module Express qui nous permet de définir le endpoint REST, qui sert d'interface entre le frontal et le dorsal.

Anchor Link
Ce paquetage représente le module Anchor Link qui nous permet de fournir des signatures de manière persistante, rapide et sécurisée pour les chaînes EOSIO construites au-dessus des demandes de signature EOSIO (EEP-7)

MongoClient
Ce paquetage représente le module MongoClient ,une bibliothèque Nodejs, qui gère la connexion et l'interaction avec une base de données MongoDB.

Socket.io
Ce paquetage représente le module Socket.IO qui permet une communication bidirectionnelle entre le client et le serveur.

5. Vue des processus

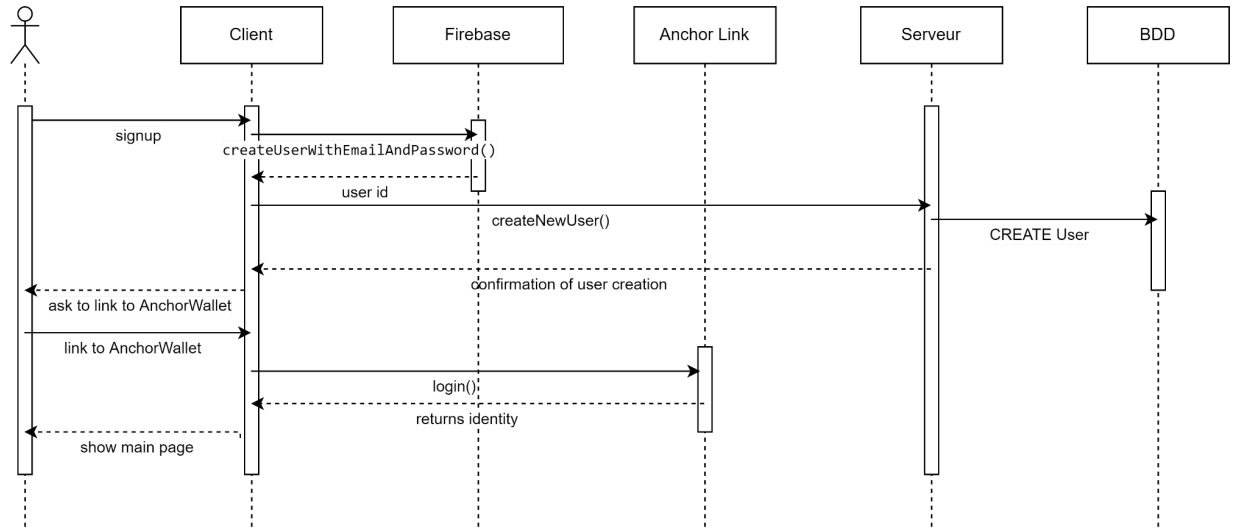


Figure - Diagramme de séquence de l'inscription

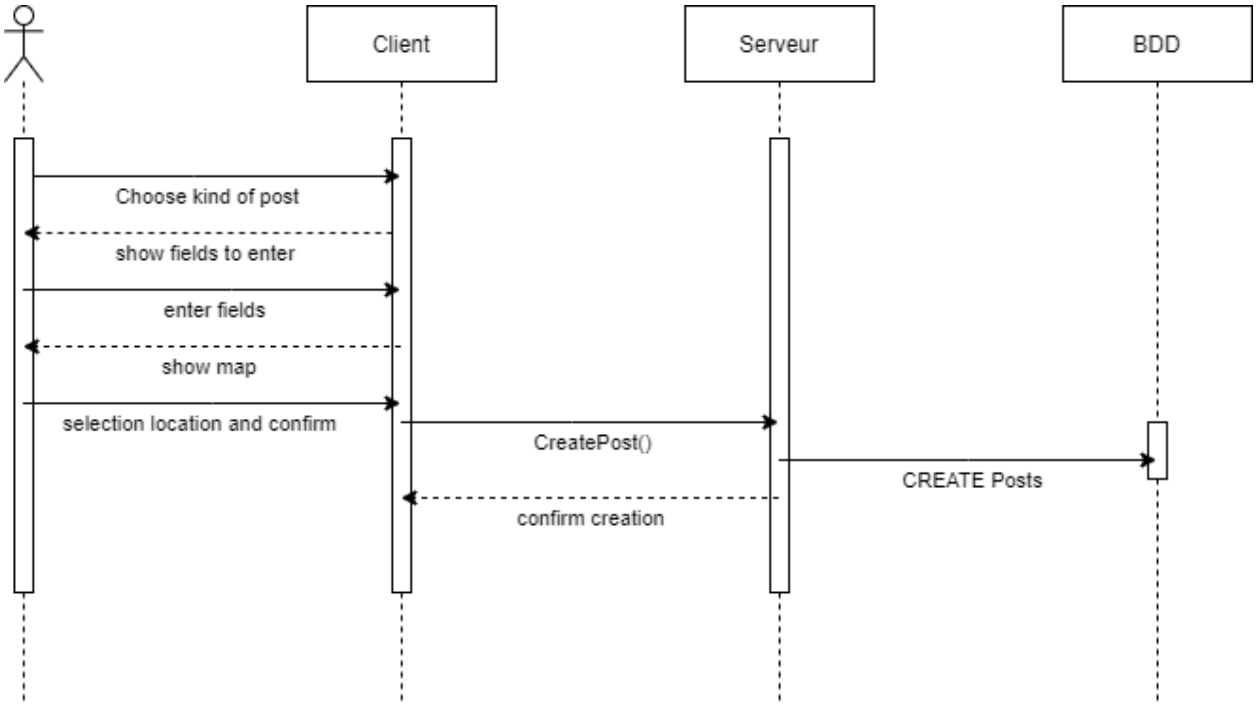


Figure - Diagramme de séquence de la création d'une annonce

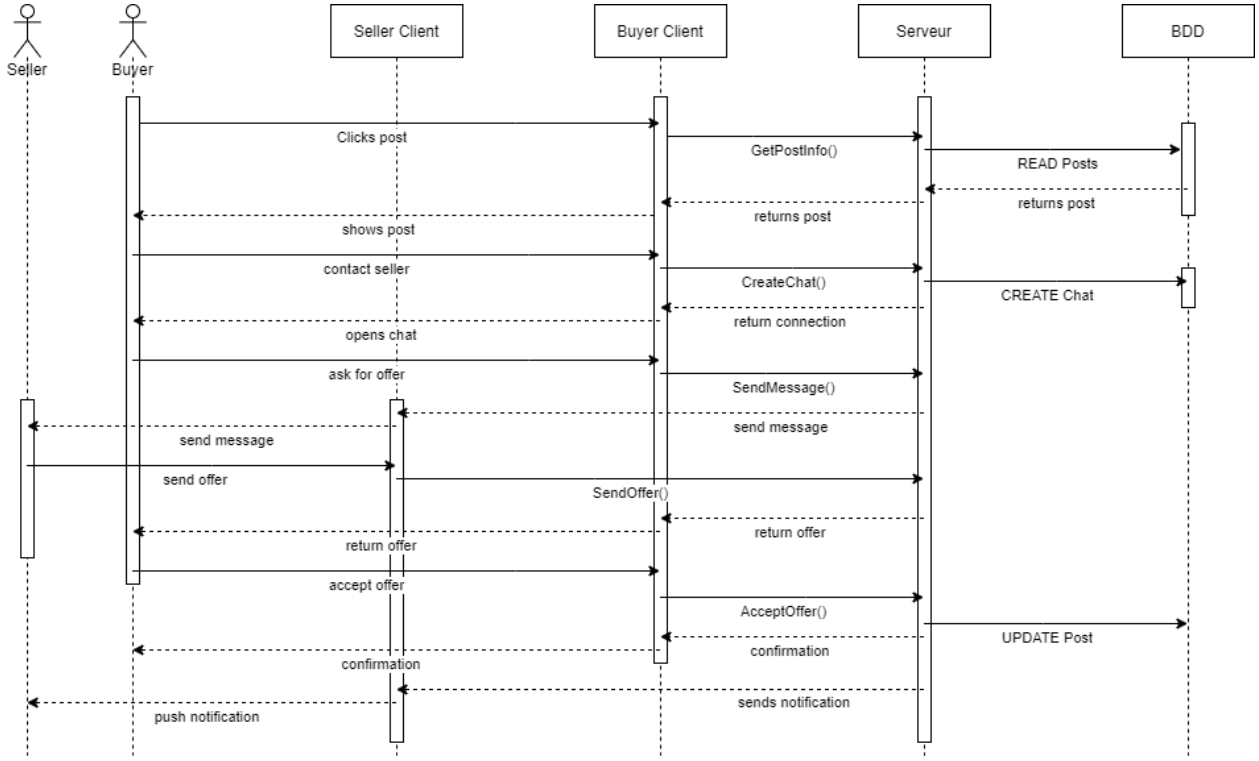


Figure - Diagramme de séquence d'acceptation d'une annonce

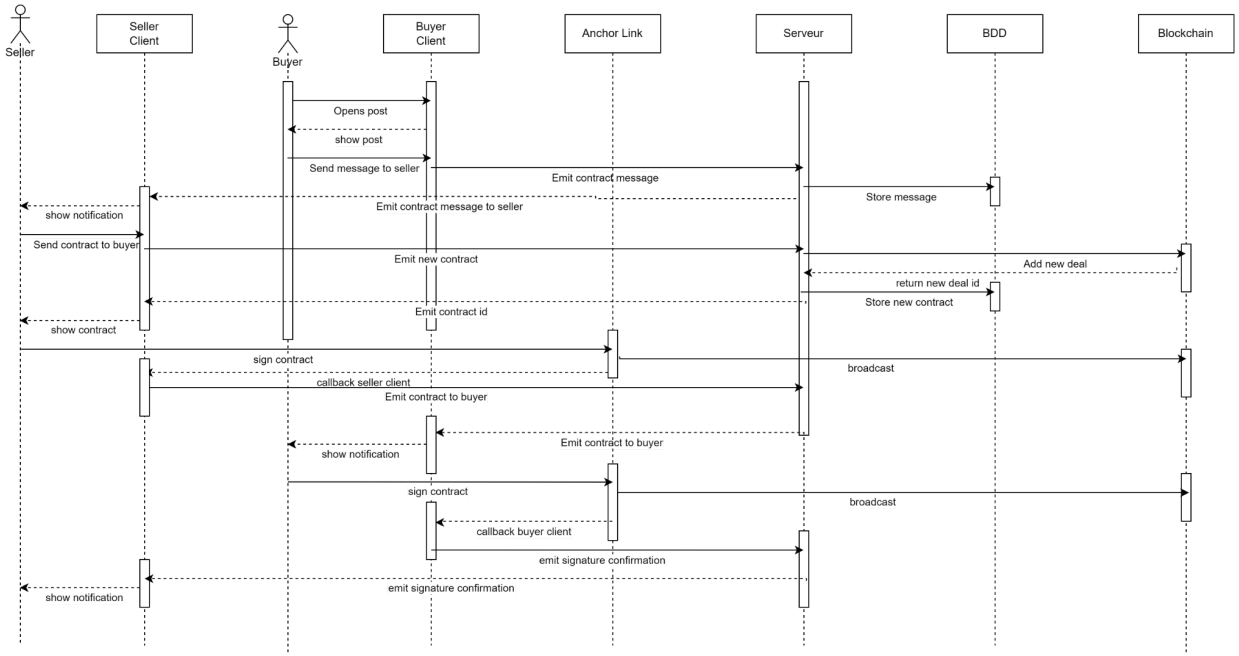
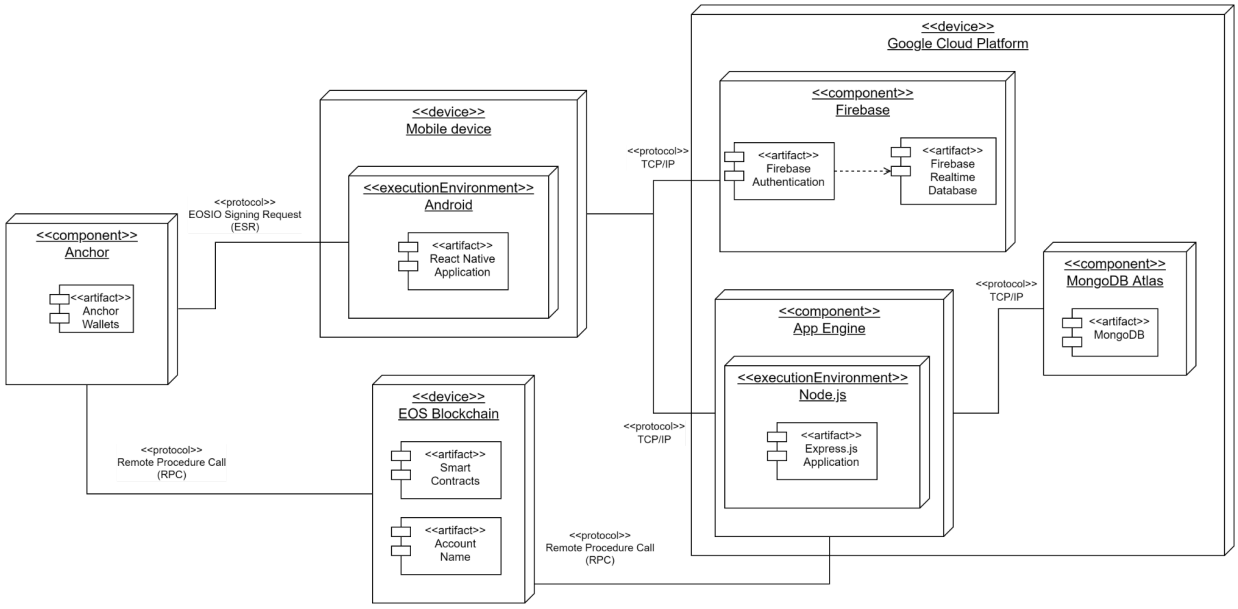


Figure - Diagramme de séquence de la création d'un contrat

6. Vue de déploiement



L'application React Native sur appareil android permet aux utilisateurs de s'inscrire/se connecter à travers le service de Firebase Authentication sur Firebase, service offert à travers la plateforme Google Cloud. Le service de Firebase Authentication utilise dans le background une base de données offerte par le service Firebase Realtime Database.
L'application communique aussi au backend, qui est une application Express.js déployée sur Google Cloud par le service App Engine. La branche master du backend est automatiquement déployée sur Google Cloud à chaque changement introduit afin que les clients utilisent toujours la dernière version disponible du serveur. Le frontend communique aussi avec Anchor afin de fournir une signature à la blockchain d'EOS, pour ensuiteeffectuer des transactions (Smart Contracts) par le protocole EOSIO Signing Request.

Le backend communique à une base de données MongoDB qui est déployée sur Google Cloud à travers le service MongoDB Atlas. La base de données de Firebase ne contient que les informations de connexion d'un utilisateur (e-mail et mot de passe par exemple). La base de données MongoDB contiendra toute information supplémentaire sur les utilisateurs (nom, certifications, etc.) ainsi que les annonces. Chaque utilisateur est associé à un id unique partagé par les deux bases de données.

7. Taille et performance

Plusieurs exigences non fonctionnelles décrites dans le SRS représentent une caractéristique de taille et de performance. Notamment, chaque action autre que la confirmation d'une transaction doit avoir un délai maximal de 3 secondes et un délai moyen de 500 millisecondes. Puisque l'application est prévue pour appareils mobiles, la performance de l'appareil doit être optimisée pour rouler l'application et les tâches plus coûteuses, comme l'écriture dans la base de données, sont exécutées sur le serveur. Il faut donc une connexion rapide entre le client, le serveur et la base de données. Utiliser le même service d'hébergement (Google Cloud), ainsi que des serveurs de la même région, pour nos différents services permet d'avoir une meilleure compatibilité et performance.

Le service de Google Cloud nous donne aussi un bon contrôle sur les ressources associées au serveur, permettant de les ajuster selon nos besoins de performance.