DÉPARTEMENT INFORMATIQUE

Rapport du projet Application IOS

Filière :

« Cyber Security »

Conception et développement d'une application MoussAid

MoussAid

V1.0

Table des matières :

Réalisé par : Ezzamouri Houssam / Er-rafiy Salma

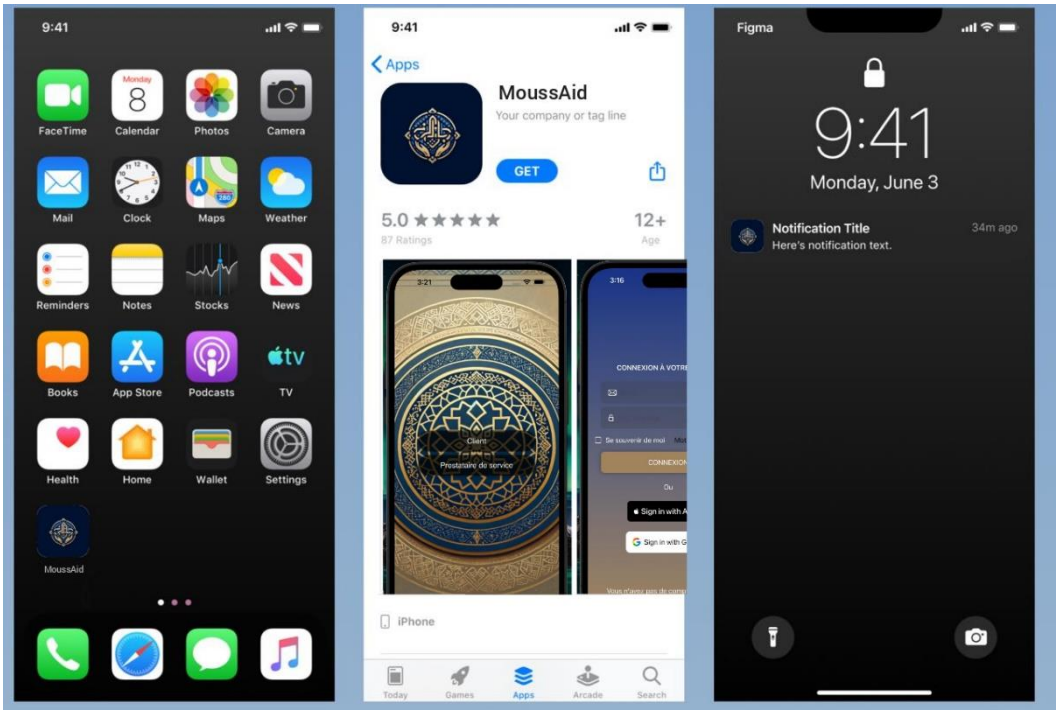
Encadré par : Pr.Jabrane Jihane

I. Introduction

Dans un monde où la digitalisation transforme radicalement les modes de consommation et les interactions sociales, l'émergence d'applications innovantes devient un pilier essentiel pour répondre aux besoins croissants des consommateurs et des entreprises. Au cœur du Maroc, une nouvelle application nommée MoussAid s'inscrit dans cette révolution numérique en proposant une plateforme polyvalente qui facilite l'accès à divers services essentiels. Conçue pour soutenir aussi bien les petites entreprises que les entrepreneurs indépendants, MoussAid offre un pont entre ces prestataires de services et les clients à la recherche de solutions spécifiques à leurs besoins quotidiens.

MoussAid se distingue par sa structure organisée en cinq grandes catégories de services, chacune ciblant un aspect vital de la vie quotidienne des Marocains : la Santé, les Services à domicile, le Transport, la Réparation, et l'Alimentation, en plus d'une catégorie supplémentaire nommée Autres Services pour englober les offres plus diversifiées. Cette segmentation permet à l'application de couvrir un large éventail de demandes, facilitant ainsi la navigation et l'amélioration de l'expérience utilisateur, tout en stimulant l'activité économique locale par la mise en valeur des compétences régionales.

L'introduction de MoussAid sur le marché marocain représente non seulement une avancée technologique, mais aussi une opportunité d'empowerment économique pour les petits entrepreneurs et une amélioration significative de l'accès aux services nécessaires pour les consommateurs. Ce rapport détaillera les fonctionnalités de l'application, explorera son impact sur le marché local et envisagera les perspectives de son évolution future.



II. Cahier des charges

Contexte du Projet :

Le Maroc, à l'instar de nombreux pays, connaît une transformation numérique rapide qui touche divers secteurs de l'économie. Cette digitalisation présente une opportunité unique pour les petites entreprises et les entrepreneurs de développer leurs activités. **MoussAid** a été conçu pour répondre à ce besoin en facilitant la connexion entre les prestataires de services locaux et les consommateurs à travers une plateforme intuitive et accessible. Ce contexte de digitalisation et de besoin accru d'efficacité dans les services quotidiens a posé les bases de la création de **MoussAid**

- **Problématique :**

Les petites entreprises et les travailleurs indépendants au Maroc font souvent face à des défis majeurs en termes de visibilité et d'accès au marché. D'autre part, les consommateurs recherchent des moyens plus simples et rapides pour accéder à des services de qualité dans des domaines aussi essentiels que la santé, l'alimentation, ou les réparations. L'application MoussAid vise à résoudre ces problèmes en offrant une plateforme qui non seulement améliore la visibilité des prestataires, mais assure également une accessibilité et une fiabilité accrues pour les consommateurs.

Objectifs :

- **Améliorer la visibilité des petites entreprises et entrepreneurs** en leur fournissant une plateforme pour présenter leurs services.

- **Faciliter l'accès aux services nécessaires** pour les consommateurs, en réduisant le temps et l'effort nécessaires pour trouver des prestataires de confiance.

- **Stimuler l'économie locale** en dynamisant le secteur des services à travers l'utilisation des technologies numériques.

- **Améliorer l'expérience utilisateur** grâce à une interface simple et une navigation intuitive.

Outils Utilisés :

- **1. Langages et Technologies :**

Swift :

Langage principal utilisé pour le développement de l'application iOS. Swift est reconnu pour sa performance et sa sécurité de type, ce qui est essentiel pour créer une application robuste et efficace.

SwiftUI :

Un framework moderne conçu par Apple pour créer des interfaces utilisateur avec une syntaxe concise et déclarative. Il facilite la création d'applications multiplateformes sur l'écosystème Apple avec un code plus simple et maintenable, offrant également une intégration poussée avec les fonctionnalités de Swift.

FireBase :

Une plateforme de développement d'applications de Google qui fournit des outils pour suivre les analytics, rapporter les crashs, créer des bases de données en temps réel, authentifier les utilisateurs et faciliter le déploiement rapide des applications. Elle est choisie pour sa facilité d'utilisation, sa scalabilité et son intégration transparente avec iOS.

- **2. Logiciels et Outils de Développement :**

Xcode :

Environnement de développement intégré (IDE) pour macOS, utilisé pour développer des applications pour iOS, macOS, watchOS et tvOS. Xcode offre des outils de développement puissants tels que l'éditeur de code, le gestionnaire de projets, et le simulateur iOS.

III. Analyse et Conception

I Modélisation Merise :

○ Modèle conceptuel de données :

Le modèle conceptuel des données (MCD) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information (figure 1). Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

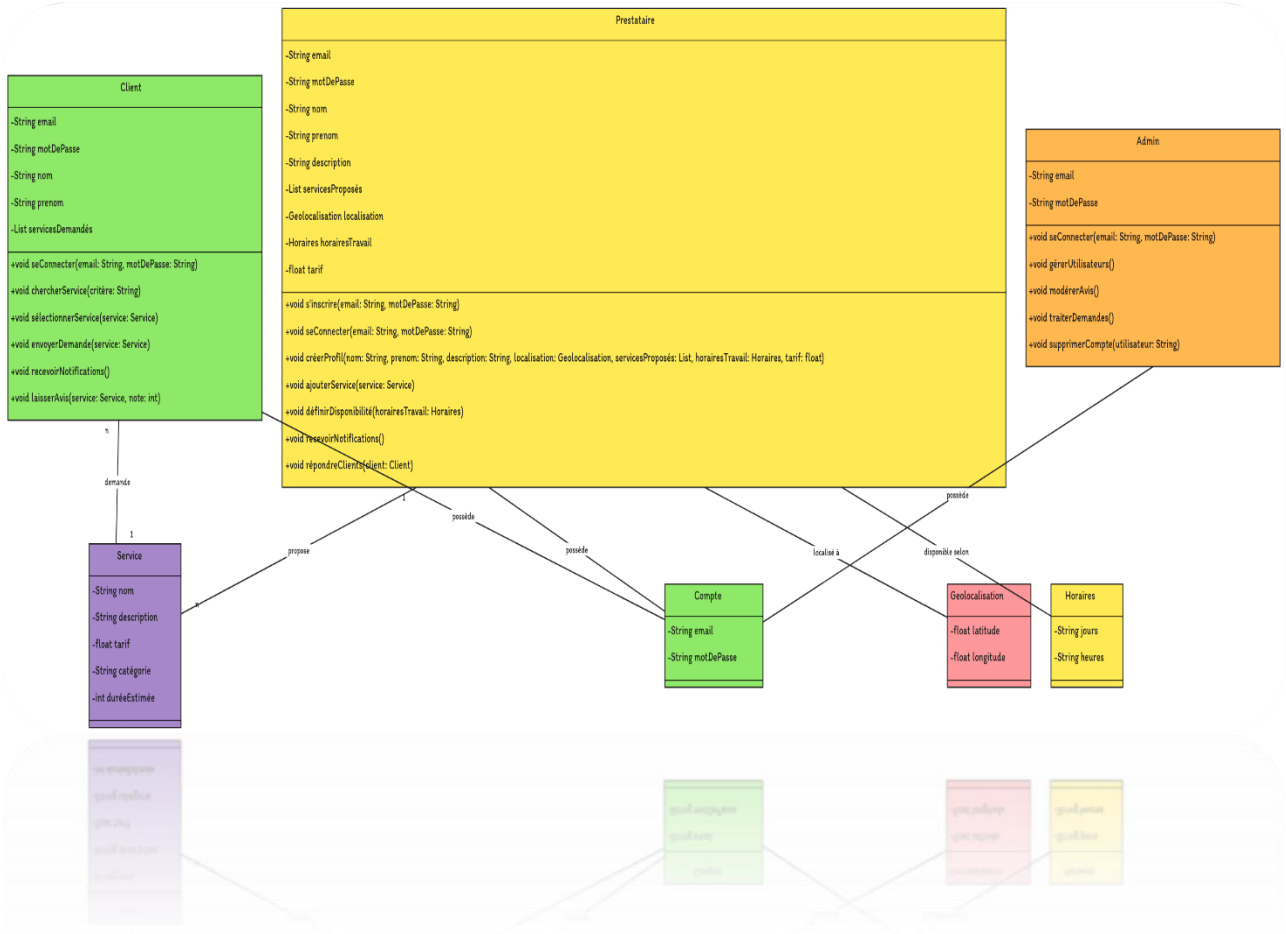


Figure 2 : le diagramme de classe

Description :

Il y a quatre classes principales représentées : `Client`, `Prestataire`, `Admin`, et `Service`. En plus, il y a trois autres classes qui semblent être des classes d'association ou des entités liées : `Compte`, `Geolocalisation`, et `Horaires`.

Voici la description de chaque classe et leurs relations :

- **Client** :
  - Attributs : Informations personnelles et détails du compte, tels que `email`, `mot de Passe`, `nom`, `prénom`, et une liste des services demandés (`services Demandés`).

- Méthodes : Fonctionnalités telles que se connecter, chercher un service, sélectionner un service, envoyer une demande, recevoir des notifications, et laisser un avis.
- Relations : Peut demander (`demande`) un ou plusieurs `Service`.

- **Prestataire** :
- Attributs : Similaires au `Client`, mais avec des `services Proposés` plutôt que des services demandés, ainsi que la `Géolocalisation` et les `Horaires` de travail.
- Méthodes : Incluent la gestion du profil, l'ajout de services, la définition de la disponibilité, etc.
- Relations : Propose (`propose`) un ou plusieurs `Service` et possède un `Compte`.

- **Admin** :
- Attributs : Détails du compte comme `email` et `mot de Passe`.
- Méthodes : Incluent la gestion des utilisateurs, la modération des avis, le traitement des demandes, et la suppression des comptes.
- Relations : Pas de relation directe avec les `Service`, mais gère l'application dans son ensemble.

- **Service** :
- Attributs : Détails du service proposé comme `nom`, `description`, `tarif`, `catégorie`, et `durée Estimée`.
- Relations : Est demandé par (`demande`) le `Client` et proposé par (`propose`) le `Prestataire`.

- **Compte** :
- Attributs : Détails de connexion comme `email` et `mot de Passe`.
- Relations : Est possédé par (`possède`) le `Client` et le `Prestataire`.

Géolocalisation :

- Attributs : `latitude` et `longitude` pour la position géographique.
- Relations : Localise (`localisé à`) le `Prestataire`.

- Horaires :

- Attributs : `jours` et `heures` de travail.
- Relations : Spécifie quand le `Prestataire` est disponible (`disponible selon`).
-

Les relations entre les classes sont représentées par des lignes reliant les boîtes. Par exemple, la ligne qui relie `Client` à `Service` indique que le client peut demander plusieurs services. Des losanges sur les lignes indiquent des relations de composition, signifiant que les objets de la classe `Compte`, `Géolocalisation`, et `Horaires` font partie intégrante des objets de la classe `Prestataire`.

Diagrammes de Cas d'Utilisation et de Séquence :

1- Le rôle du diagramme de séquences dans la conception du projet :

Les diagrammes de séquence sont des outils de modélisation visuelle qui décrivent de manière séquentielle l'interaction entre différents objets d'un système au fil du temps. Ils sont largement utilisés dans l'ingénierie logicielle pour représenter le flux des interactions entre les acteurs, les entités et les composants d'un système pendant l'exécution d'un scénario spécifique. Voici quelques points clés sur leur rôle et leur importance dans la conception logicielle : Représentation des Interactions : Les diagrammes de séquence permettent de représenter graphiquement les interactions entre les objets du système, mettant en lumière l'ordre chronologique dans lequel ces interactions se produisent. Visualisation du Flux Temporel : Ils offrent une représentation chronologique des événements, montrant comment les objets interagissent les uns avec les autres au fil du temps. Cela est particulièrement utile pour comprendre la séquence d'exécution des opérations dans un scénario donné. Analyse des Scénarios d'Utilisation : Les diagrammes de séquence sont souvent utilisés pour modéliser des scénarios d'utilisation spécifiques, tels que des cas d'utilisation particuliers ou des opérations système. Cela aide à visualiser et à comprendre le comportement attendu du système. Identification des Acteurs : Ils permettent d'identifier les acteurs (entités ou composants) impliqués dans un scénario donné, ainsi que leurs rôles respectifs dans les interactions. Validation de la Logique de Conception : Ils servent de support pour valider la logique de conception en illustrant comment les objets collaborent pour atteindre un objectif particulier. Cela facilite la détection des erreurs conceptuelles potentielles dans la conception.

Communication et Documentation : Les diagrammes de séquence fournissent un moyen visuel et intuitif de communiquer des détails techniques du système à différentes parties prenantes, y compris les développeurs, les concepteurs et les responsables de projet. Ils servent également de documentation visuelle pour le système.

Le diagramme de séquence :

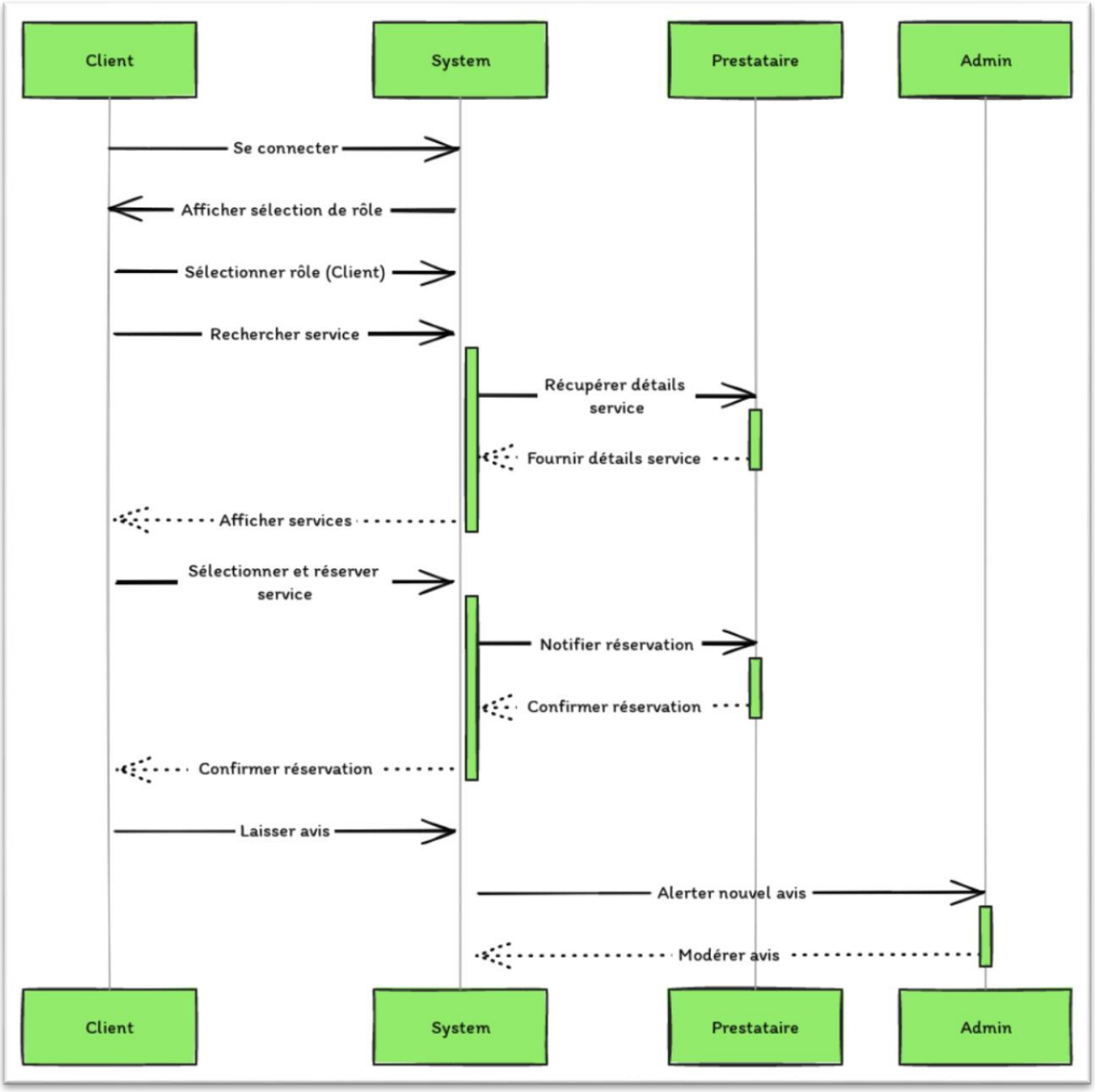


Figure 2 : le diagramme de séquence

Description :

Le diagramme de séquence illustre les interactions entre un client, le système, un prestataire de services et un administrateur dans le contexte d'une plateforme de services. Les interactions se produisent dans l'ordre séquentiel suivant :

1. **Se connecter** : - Le client initie le processus en se connectant au système.
2. **Afficher sélection de rôle :** - Le système présente les options de rôles à l'utilisateur, qui peut choisir entre différents rôles tels que Client, Prestataire ou Admin.

3. **Sélectionner rôle (Client) :** - Le client sélectionne spécifiquement le rôle de "Client" dans l'interface du système.
4. **Rechercher service :** - Le client demande à rechercher un service. Cela déclenche une interaction avec le système pour retrouver les informations correspondantes.
5. **Récupérer détails service :** - Le système demande au prestataire de fournir les détails du service recherché par le client.
6. **Fournir détails service :** - Le prestataire envoie les informations demandées au système. Cette interaction est souvent asynchrone, indiquée par les pointillés
7. **Afficher services :** - Le système affiche les services et les détails fournis par le prestataire au client.
8. **Sélectionner et réserver service :** -Le client choisit un service et fait une réservation à travers le système.
9. **Notifier réservation :** - Le système informe le prestataire de la nouvelle réservation effectuée par le client.
10. **Confirmer réservation :** - Le prestataire confirme la réservation et le système relaye cette confirmation au client.
11. **Laisser avis** : - Après la prestation du service, le client laisse un avis via le système.
12. **Alerter nouvel avis** : - Le système alerte l'administrateur d'un nouvel avis posté.
13. **Modérer avis** : - L'administrateur, à son tour, peut modérer cet avis. Cela pourrait inclure la vérification de la conformité de l'avis avec les politiques de la plateforme.

Les flèches représentent les messages (ou interactions) entre les acteurs et le système. Les flèches pleines représentent les messages synchrones où le demandeur attend une réponse immédiate, tandis que les flèches en pointillés représentent les messages asynchrones où une réponse n'est pas immédiatement attendue.

2. Diagramme de Cas d'utilisation :

Un diagramme de cas d'utilisation est un diagramme UML qui représente le comportement fonctionnel d'un système. Il est utilisé pour modéliser les interactions entre les acteurs du système et les fonctionnalités du système.

Les éléments d'un diagramme de cas d'utilisation sont les suivants

● Les acteurs représentent les utilisateurs du système. Ils peuvent être des humains, des machines ou d'autres systèmes.

● Les cas d'utilisation représentent les fonctionnalités du système. Ils décrivent ce que le système doit faire pour répondre aux besoins des acteurs

● Les relations entre les acteurs et les cas d'utilisation représentent les interactions entre les acteurs et les fonctionnalités du système.

Description :

Ce diagramme de cas d'utilisation présente les interactions entre trois acteurs principaux dans une application : l'Admin, le Client et le Prestataire. Voici les détails de leurs interactions :

- **Admin** :

- Peut **gérer les comptes,** ce qui inclut probablement la création, la mise à jour, et la suppression des comptes utilisateurs.

- A la capacité de **valider les inscriptions** des nouveaux prestataires ou clients, ce qui implique un contrôle ou une vérification avant que les comptes ne soient activés.

- A la responsabilité de **modérer les avis** laissés par les clients, assurant que les commentaires respectent les lignes directrices de la plateforme.

- Peut **superviser les transactions** pour maintenir l'intégrité financière de la plateforme.

- Peut **résoudre les litiges**, agissant en tant que médiateur entre les prestataires et les clients en cas de conflit.

- **Client :**

- Peut **chercher des prestataires** selon divers critères comme la localisation, le type de service, etc.

- Peut **consulter les profils** des prestataires pour voir leur évaluations, commentaires, et historique de services.

- A la possibilité de **réserver un service** une fois qu'un prestataire a été choisi.

- Après la prestation du service, le client peut **laisser un avis** qui peut être modéré par l'admin.

- Peut **suivre ses commandes** ou réservations et leur statut actuel.

- **Prestataire :**

- Doit d'abord **s'inscrire** pour offrir ses services sur la plateforme.

- Peut **gérer son profil**, qui inclut ses informations personnelles, sa présentation et ses qualifications.

- Est capable de **proposer des services** et décrire en détail ce qu'il offre.

- Doit **définir sa disponibilité** pour que les clients sachent quand les services peuvent être réservés.

- Interagit directement avec les clients lorsqu'il reçoit des demandes de services ou lors de la fourniture du service.

Les lignes pleines indiquent les interactions directes et standard entre un acteur et ses cas d'utilisation, tandis que les lignes en pointillés peuvent représenter des interactions optionnelles ou des extensions de cas d'utilisation standard. Par exemple, l'interaction en pointillés entre le Client et l'Admin pour « Modérer les avis » indique que cette action est une extension du cas d'utilisation « Laisser avis » effectué par le Client.

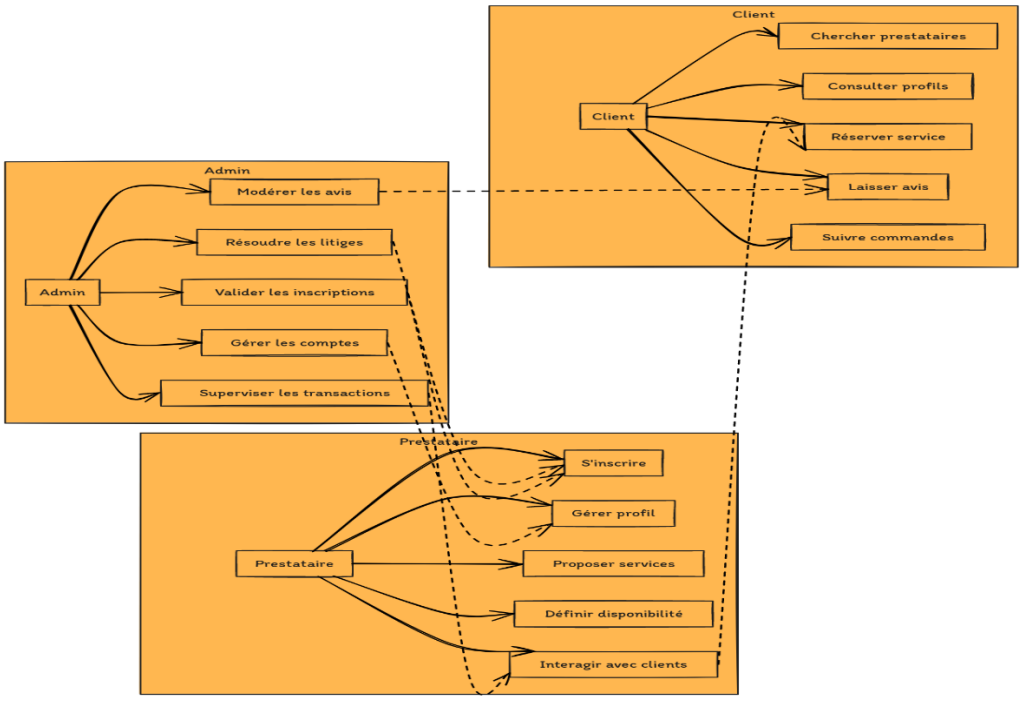
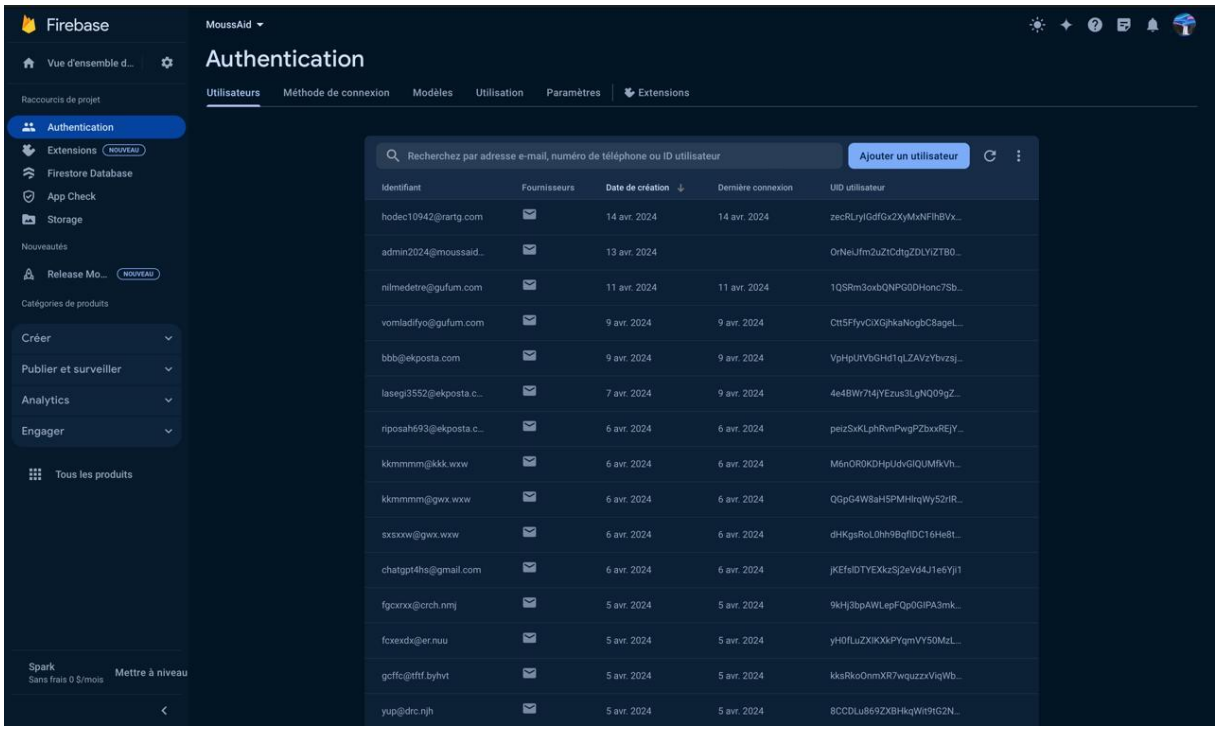


Figure 4 : l e diagramme de cas d'utilisation

IV. Implémentation de la base de données

- **l'interface de la section « Authentification »**



on trouve une liste d'utilisateurs avec plusieurs colonnes :

- **« Identifiant »** : des adresses e-mail des utilisateurs, masquées pour des raisons de confidentialité.

- **« Fournisseurs »** : cette colonne indique la méthode d'inscription ou de connexion utilisée par l'utilisateur (par exemple, e-mail/mot de passe, Google, Facebook, etc.).

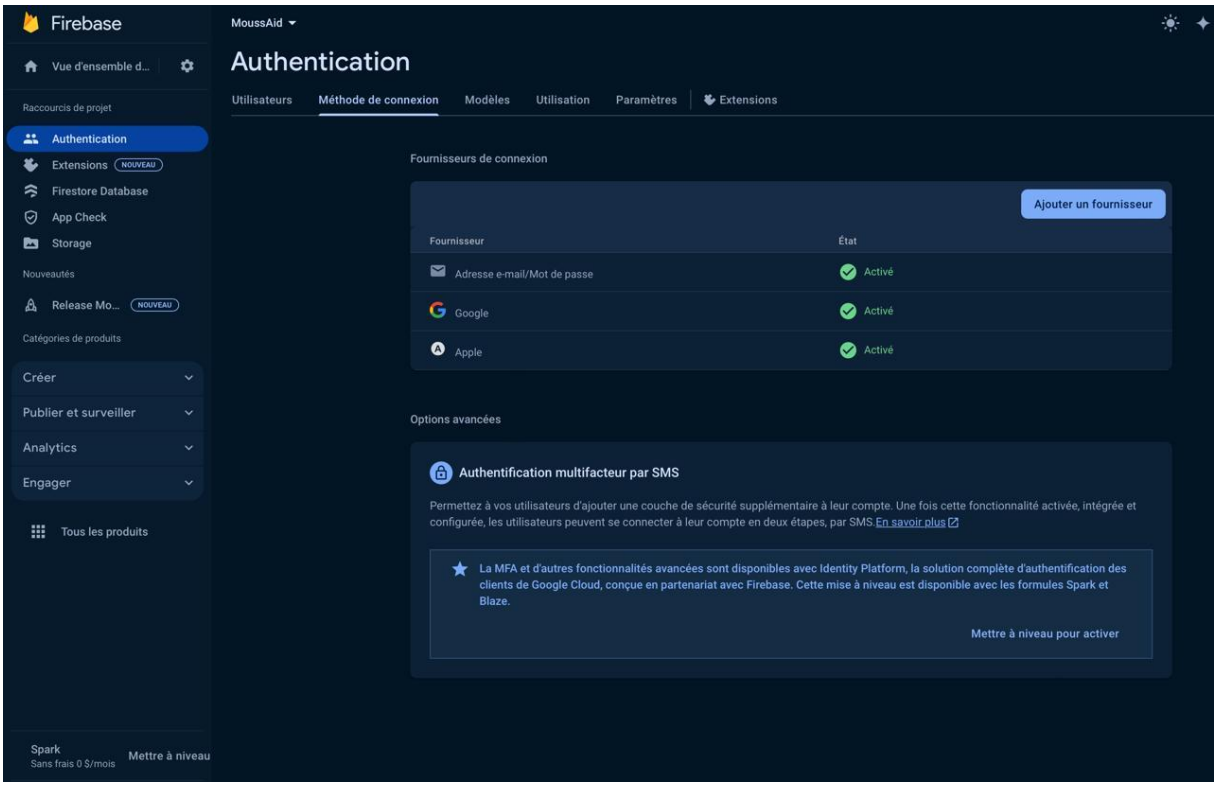
- « Date de création » : montre les dates auxquelles les comptes utilisateurs ont été créés.

- « Dernière connexion » : affiche quand l'utilisateur s'est connecté pour la dernière fois.

- « UID utilisateur » : identifiant unique attribué à chaque utilisateur, partiellement masqué ici.

Dans le coin supérieur droit, il y a un bouton « Ajouter un utilisateur », qui permettrait d'enregistrer manuellement un nouvel utilisateur.

- **Méthode de connexion :**



Description :

On voit trois méthodes de connexion actives pour les utilisateurs de l'application :

1. **« Adresse e-mail/Mot de passe »** : Permet aux utilisateurs de créer un compte et de se connecter à l'application en utilisant une adresse e-mail et un mot de passe.

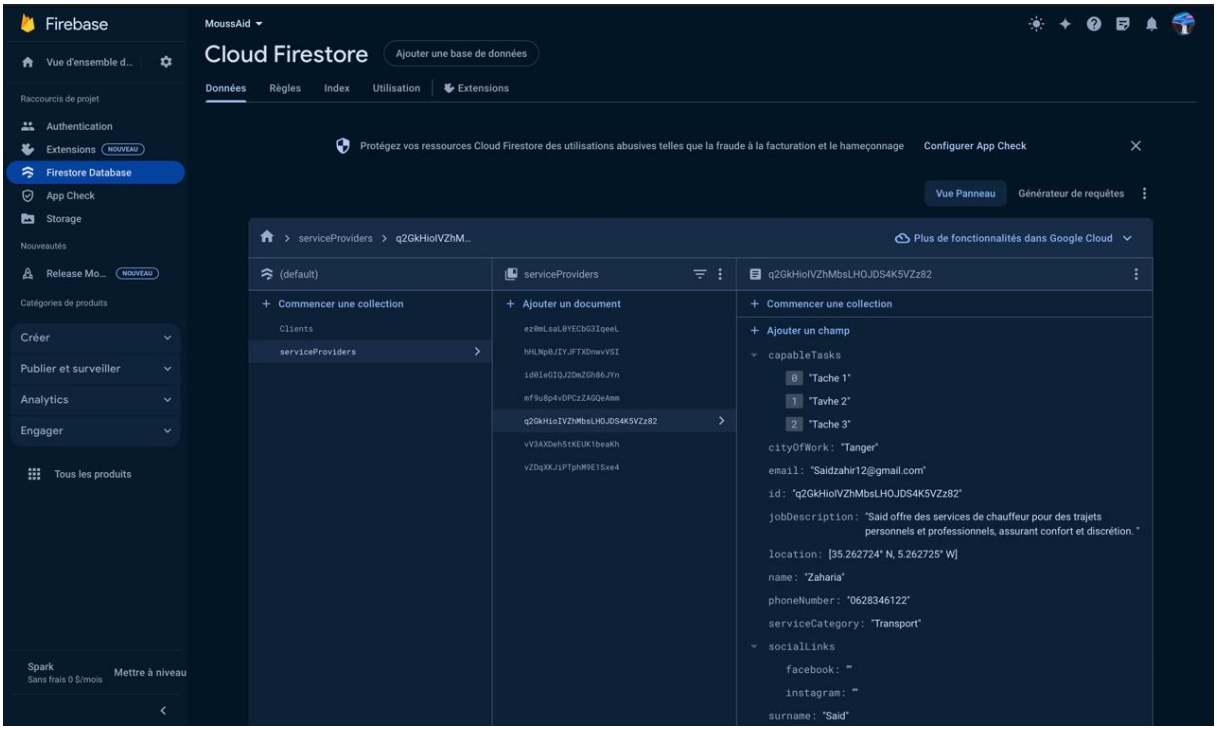
2. **« Google »** : Permet l'authentification via un compte Google existant, facilitant ainsi la connexion sans avoir besoin de créer un nouveau compte.

3. **« Apple »** : Offre une méthode de connexion via un identifiant Apple, ce qui est souvent utilisé pour les applications IOS pour une intégration transparente avec les services de l'écosystème Apple.

À côté de chaque méthode, il y a une icône en forme de coche sur un fond vert, indiquant que ces méthodes sont actuellement actives et disponibles pour les utilisateurs.

Dans le volet « Options avancées » en dessous, il y a une section intitulée **« Authentification multi factor par SMS ».** Cela suggère que l'application peut être configurée pour demander une étape supplémentaire de vérification, où les utilisateurs reçoivent un code par SMS pour confirmer leur identité. Ceci est une mesure de sécurité additionnelle pour s'assurer que l'accès au compte est bien légitime.

- **L’interface de la console :**



<u>Description :</u>

Deux volets principaux :

Le volet de gauche liste les collections dans la base de données Firestore avec deux collections apparentes : « **Clients** » et « **serviceProviders** ». (Une collection dans Firestore est un ensemble de documents, qui contiennent les données de la base.)

Le volet de droite montre les détails d'un document spécifique de la collection « **serviceProviders** ». Ce document contient plusieurs champs qui représentent les attributs du fournisseur de services :

- **`capableTasks`:** Liste des tâches que le fournisseur de services peut exécuter, divisées en sous-champs comme `Tache 1`, `Tache 2`, et `Tache 3`.

- **`cityOfWork`:** La ville où le fournisseur de services travaille, ex :"Tanger".

- **`email`:** L'adresse e-mail du fournisseur de services,

- **`id`:** L'identifiant unique du document dans la collection, ici masqué partiellement.

- **`jobDescription`:** Une description des services offerts par le fournisseur .

- **`location`:** Coordonnées géographiques du fournisseur de services.

- **`name`:** Le prénom du fournisseur de services

- **`phoneNumber`:** Le numéro de téléphone du fournisseur, partiellement masqué.

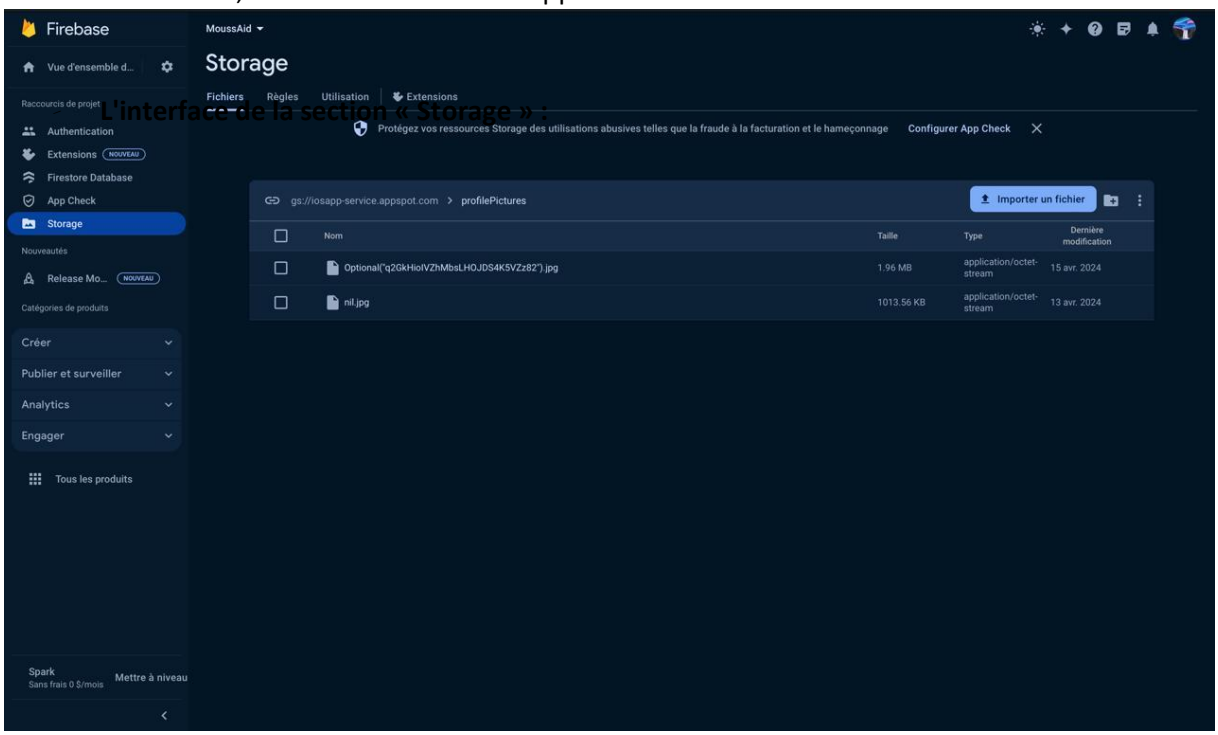
- **`serviceCategory`:** La catégorie de service .

- **`socialLinks`:** Liens vers des profils de médias sociaux, avec des champs pour Facebook et Instagram.

- **`surname`:** Le nom de famille du fournisseur de services.

Dans le coin supérieur droit du volet du document, il y a un menu qui permet probablement d'effectuer d'autres opérations sur le document, comme l'édition ou la suppression.

L'interface de la section « Storage » :



Ce screen représente l'interface de la section « Storage » de la console Firebase, spécifiquement le volet de gestion des fichiers. (Firebase Storage est un service qui offre un stockage d'objets sécurisé, puissant et facile à utiliser pour les développeurs d'applications.)

Cet écran est principalement utilisé pour gérer les ressources média ou autres fichiers qui sont utilisés par l'application, permettant aux développeurs d'uploader, de télécharger, et de gérer l'accès aux fichiers stockés

On remarque une liste des fichiers stockés :

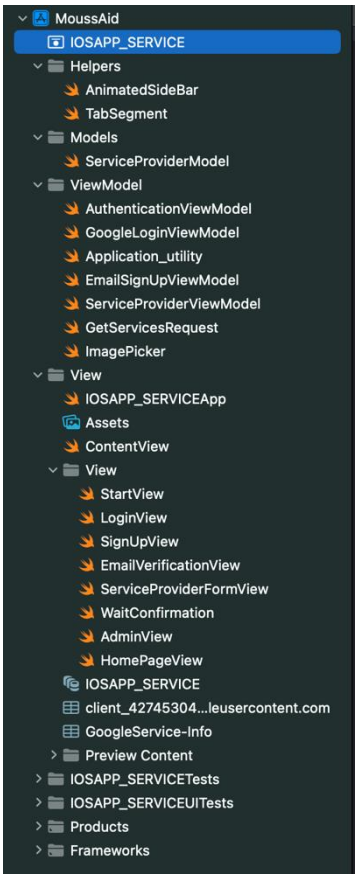
- Le premier fichier, qui est composé d'une `.jpg`, a une taille de 1,96 MB, est de type `application/octet-stream`.

- Le deuxième fichier, intitulé `nil.jpg`, est plus petit avec une taille de 1013,56 KB, est également de type `application/octet-Stream'. Un bouton « Importer un fichier », indiquant que l'utilisateur peut télécharger de nouveaux fichiers vers ce Storage.

V. Interface Graphique

Structure Code :

Dans la figure ci-jointe, nous observons l'architecture du projet de l'application MoussAid telle qu'elle est organisée dans l'environnement de développement Xcode. Cette représentation visuelle révèle la structure hiérarchique du code et la manière dont les différentes composantes de l'application sont agencées pour une clarté optimale et une maintenance aisée.



- **Le dossier Helpers** englobe les outils et composants fonctionnels qui supportent les fonctionnalités de l'application, tels que l'animation de la barre latérale et la gestion des segments d'onglets pour une navigation intuitive.

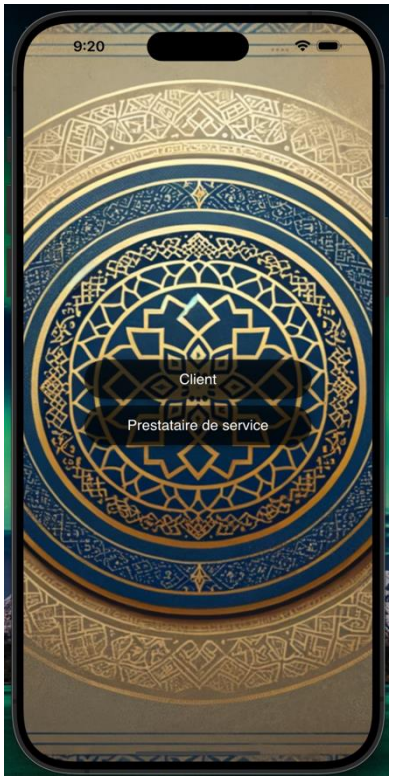
- **Models** est dédié aux structures de données qui définissent les entités clés de l'application, comme les modèles des prestataires de services, permettant une manipulation et un stockage efficaces des informations.

- **Dans ViewModel**, nous trouvons les classes responsables de la logique métier et de la communication entre les modèles et les vues, incluant des éléments tels que l'authentification, la connexion via Google, et l'inscription par e-mail.

- **Le dossier View** contient l'ensemble des vues de l'application, représentant les interfaces utilisateur par lesquelles les interactions se font, allant de l'écran de démarrage aux formulaires de service et à la page d'accueil.

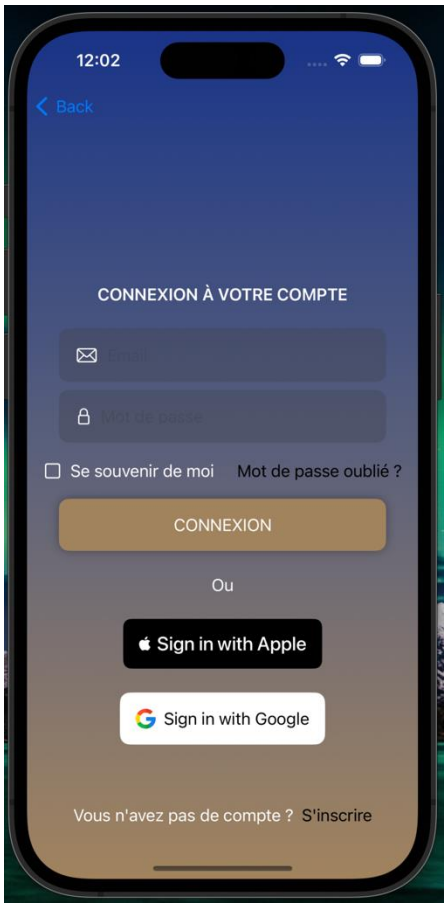
Chacune de ces catégories joue un rôle crucial dans le fonctionnement de l'application et, ensemble, elles constituent le squelette sur lequel repose MoussAid. Cette structure systématique est le fruit d'une conception méticuleuse visant à séparer les responsabilités de chaque couche de l'application, favorisant ainsi une approche de développement modulaire et scalable.

Écran de Sélection de Rôle



- **Objectif**: Cet écran sert de point de départ où l'utilisateur choisit son parcours en tant que "Client" ou "Prestataire de service". Il offre une introduction esthétique et fonctionnelle à l'application MoussAid, tout en orientant les utilisateurs vers les fonctionnalités qui leur sont pertinentes.
- **Design Visuel**: L'arrière-plan affiche un motif traditionnel marocain, ajoutant une touche culturelle qui favorise l'identification locale à l'application. Les options de sélection sont présentées de manière épurée pour une expérience utilisateur intuitive.
- **Code Source Principal**: `StartView.swift`
- **Explication du Code**:
  - Le fichier `StartView.swift` contient le code SwiftUI qui structure cet écran. La mise en page utilise une `VStack` pour l'alignement vertical des éléments.
  - Les boutons de sélection font usage de modificateurs pour correspondre au thème visuel de l'application, avec des effets de transparence et d'échelle pour mettre en évidence l'interaction de l'utilisateur.
  - La logique de navigation est encapsulée dans des gestionnaires d'événements associés aux boutons, permettant de diriger les utilisateurs vers les parcours spécifiques en fonction de leur sélection de rôle.

Écran de Connexion



- **Objectif:** Cet écran est crucial car il sécurise l'accès utilisateur et sert de point d'entrée à l'expérience personnalisée dans MoussAid. Il permet aux utilisateurs de se connecter à leur compte en utilisant leur email et mot de passe ou via des méthodes de connexion tierces pour une commodité accrue.
- **Design Visuel:** La conception de l'écran met l'accent sur la clarté et la facilité d'utilisation. Les champs de texte pour l'adresse e-mail et le mot de passe sont directement identifiables, et les boutons pour les actions de connexion sont mis en valeur de manière à guider l'utilisateur naturellement à travers le processus de connexion.
- **Code Source Principal: LoginView.swift**
- **Explication du Code:**

LoginView.swift contient la définition de l'interface de connexion et la logique de présentation.

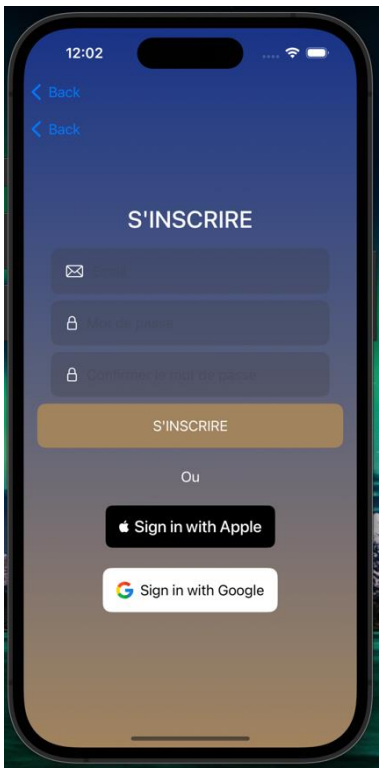
Les champs de saisie utilisent des indicateurs visuels comme des icônes pour une identification rapide de chaque champ de texte.

Des options telles que "Se souvenir de moi" et "Mot de passe oublié ?"(seront implementez en next version) sont incluses pour améliorer l'expérience utilisateur en apportant de la souplesse dans la gestion de la session.

Les boutons "Sign in with Apple" et "Sign in with Google" offrent des alternatives de connexion rapide, utilisant les API respectives pour une authentification sécurisée et fiable.

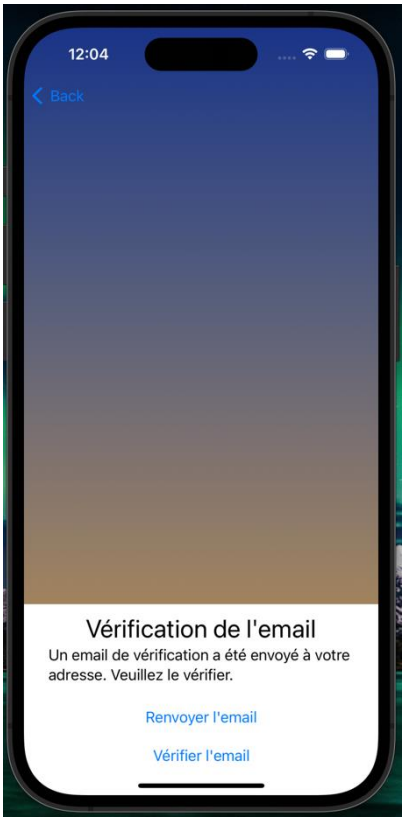
En bas, un lien invite les utilisateurs qui n'ont pas de compte à s'inscrire, favorisant ainsi l'acquisition d'utilisateurs.

Écran d'Inscription



- **Objectif**: Cet écran permet aux nouveaux utilisateurs de créer un compte pour accéder aux services de MoussAid. Il est conçu pour faciliter l'inscription avec des champs pour l'email et le mot de passe, ainsi que des options d'inscription rapide via Apple et Google.
- **Design Visuel**: L'écran est centré sur la simplicité et l'efficacité. Les champs de saisie pour l'email et la confirmation de mot de passe sont bien visibles, permettant une expérience d'inscription sans tracas. Les boutons d'inscription tiers sont également bien positionnés pour offrir des alternatives d'accès rapide.
- **Code Source Principal**: **SignUpView.swift**
- **Explication du Code**:
  - Le fichier **SignUpView.swift** orchestre l'affichage des champs de texte et des boutons d'action.
  - L'interaction utilisateur avec les champs de texte est gérée de façon sécurisée, en veillant à ce que les mots de passe soient confirmés correctement avant de procéder.
  - Les options d'inscription avec Apple et Google utilisent les frameworks d'authentification correspondants pour une intégration sécurisée au sein de l'écosystème iOS.
  - Des validations de formulaire garantissent que les informations fournies sont formatées correctement et que tous les champs requis sont remplis avant de permettre à l'utilisateur de soumettre le formulaire.

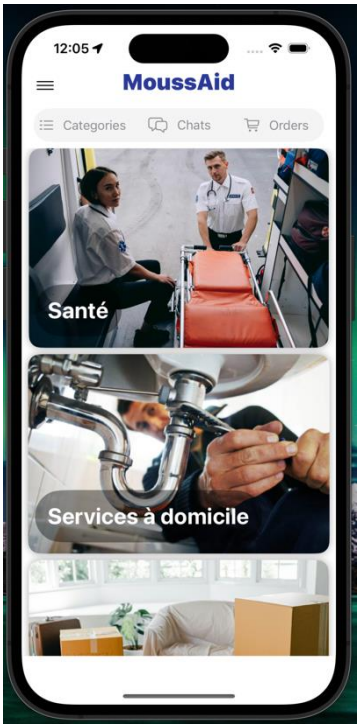
Écran de Vérification de l'Email



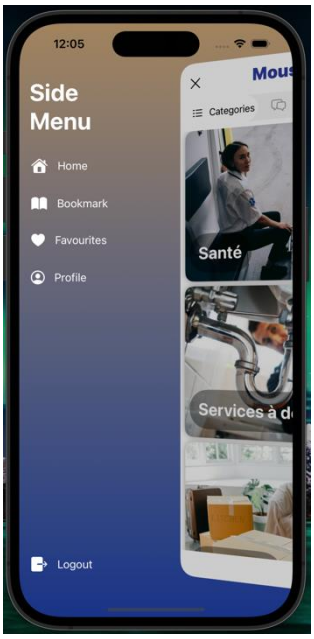
- **Objectif**: Cette vue est dédiée à la vérification de l'adresse email de l'utilisateur. Après l'inscription, elle assure que l'utilisateur possède l'accès à l'email fourni, renforçant ainsi la sécurité des comptes utilisateurs.
- **Capture d'Écran**: [Insérer la capture d'écran ici]
- **Design Visuel**: Un message clair informe l'utilisateur qu'un email de vérification a été envoyé. Deux options sont présentées : renvoyer l'email de vérification ou vérifier l'email après l'avoir consulté, rendant le processus de vérification direct et accessible.
- **Code Source Principal**: **EmailVerificationView.swift**
- **Explication du Code**:
  - Le fichier **EmailVerificationView.swift** implémente l'interface utilisateur affichée ici et contrôle les interactions telles que le renvoi d'email de vérification et la confirmation de vérification.

- Des boutons interactifs offrent à l'utilisateur les actions requises pour terminer le processus d'inscription ou pour résoudre les problèmes de non-réception de l'email.
- Le processus de vérification est intégré de manière sécurisée pour s'assurer que l'utilisateur finalise son inscription avec une adresse email validée, établissant ainsi une base sûre pour la gestion du compte utilisateur.

Écran Principal et Menu Latéral de l'Application MoussAid



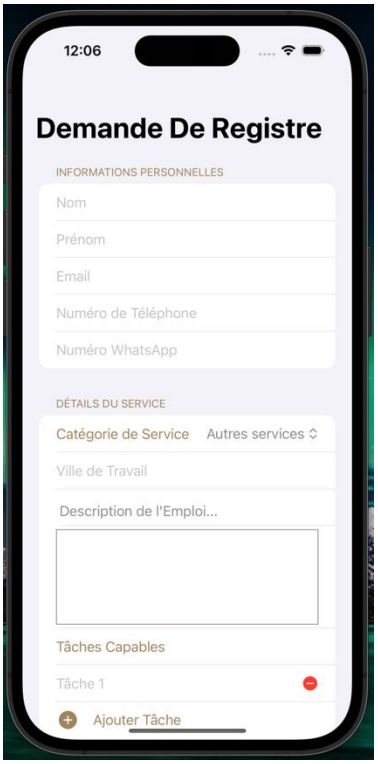
- **Objectif de l'Écran Principal**: Cet écran offre une vue d'ensemble des catégories principales de services disponibles sur MoussAid. En cliquant sur une catégorie, l'utilisateur est dirigé vers une liste détaillée de services au sein de cette catégorie, facilitant ainsi la recherche de services spécifiques.
- **Code Source Principal de l'Écran**: **HomePageView.swift**
- **Explication du Code de l'Écran Principal**:
- **HomePageView.swift** sert à construire la grille des catégories de services, chaque cellule étant une représentation visuelle interactive d'une catégorie.
- Utilise des **ScrollView** et des **Grid** pour un affichage fluide et réactif des catégories sur différents appareils et orientations.
- Les images et les titres des services sont chargés dynamiquement, ce qui permet de mettre facilement à jour le contenu si nécessaire.



- **Objectif du Menu Latéral**: Le menu latéral, accessible via l'icône hamburger, offre un accès rapide aux différentes sections de l'application, telles que le profil utilisateur, les signets, les favoris et une option de déconnexion.
- **Code Source du Menu Latéral**: **SideMenuView.swift**
- **Explication du Code du Menu Latéral**:
  - **SideMenuView.swift** organise les options de navigation dans une **List** ou une **VStack**, chacune avec un identifiant et une icône correspondante pour une reconnaissance intuitive.
  - Comprend des mécanismes de navigation qui lient chaque élément de menu à sa vue correspondante, permettant une navigation sans friction dans l'application.
  - Intègre des actions telles que la déconnexion, qui sont gérées par des gestionnaires d'événements sécurisés pour préserver l'intégrité du compte utilisateur.

Écran de Formulaire de Demande de Prestataire de Service

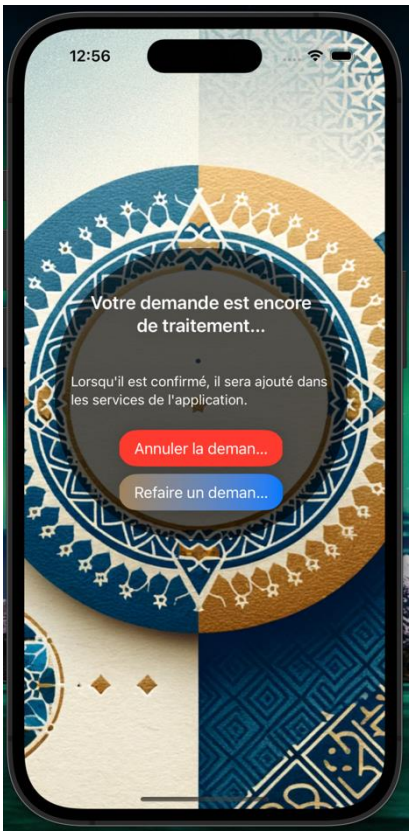
- **Objectif**: Cet écran est la première étape pour un prestataire de service souhaitant s'inscrire sur MoussAid. Il recueille des informations détaillées pour construire un profil complet, permettant aux clients de choisir les prestataires les plus appropriés à leurs besoins.
- **Capture d'Écran**: [Insérer les captures d'écran ici]
- **Code Source Principal**: **ServiceProviderFormView.swift**
- **Explication du Code**:
    - **ServiceProviderFormView.swift** est responsable de présenter le formulaire et de collecter les données saisies par l'utilisateur.
    - Il comprend des **TextField** pour les données personnelles, un **Picker** pour la sélection de la catégorie de service, et des **TextEditor** pour une description détaillée des services offerts.
    - Le formulaire permet également d'ajouter et de supprimer des tâches dynamiquement, offrant une flexibilité pour détailler les compétences spécifiques du prestataire.
    - La section de téléchargement de photo du profil facilite l'ajout d'une touche personnelle, tandis que les champs de localisation captent les coordonnées GPS pour faciliter la mise en relation avec les clients locaux.
    - Les boutons "Enregistrer la localisation", "Annuler", et "Soumettre" permettent de valider, de rejeter ou de procéder avec l'enregistrement des informations fournies, assurant un processus d'inscription cohérent et efficace.



Lorsqu'un prestataire de service entre pour la première fois, il est guidé à travers ce formulaire détaillé, conçu pour assurer que toutes les informations nécessaires sont capturées pour permettre aux clients de faire des choix éclairés sur la plateforme MoussAid.

Écran de Confirmation de Traitement de Demande

- **Objectif**: Cet écran informe le prestataire de service que sa demande d'inscription est en cours de traitement. Il fournit un feedback visuel après la soumission du formulaire, indiquant que les informations sont en phase de vérification par l'équipe de MoussAid.



- **Code Source Principal**: **WaitConfirmation.swift**
- **Explication du Code**:
  - **WaitConfirmation.swift.** affiche un message de statut et offre des options pour annuler ou refaire la demande si nécessaire.
  - Des **Alert** ou des **Modal** pourraient être utilisés pour confirmer l'annulation ou la resoumission de la demande, s'assurant que l'utilisateur a l'intention de réaliser ces actions.
  - Le design met l'accent sur la transparence du processus, avec des messages clairs et des boutons d'action qui reflètent les possibilités suivantes pour le prestataire.

L'écran joue un rôle important dans la gestion des attentes des prestataires, leur assurant que leurs efforts pour rejoindre la plateforme sont reconnus et traités avec attention.

.....

Accéder aux codes sources de l'application sur Google Drive

Remarque Importante :
Nous souhaitons souligner que la durée allouée à ce projet était relativement brève, ce qui a restreint notre capacité à achever pleinement et à affiner la conception de notre application. Malgré ce défi temporel, nous avons réussi à implémenter les fonctionnalités clés et essentielles. Nous sommes reconnaissants envers notre professeur pour nous avoir confié ce projet enrichissant, qui nous a permis de renforcer nos compétences en programmation avec Swift et de mieux comprendre le développement sous iOS.

Nous tenons à exprimer notre gratitude pour l'opportunité d'apprendre et de progresser à travers ce projet. Nous envisageons de continuer le développement de l'application et de peaufiner ses fonctionnalités. Dès que la version finale sera prête, nous la ferons parvenir à votre adresse e-mail.

Merci pour votre soutien et votre passion qui inspirent notre engagement continu.