2025/2026

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

Rapport

Dev-Mobile

Renting platform

Prepared By:
Ghalbi mohamed reda
ABOUNAY aymane

Résumé

Avec l'évolution rapide des technologies mobiles et la généralisation de l'usage des smartphones, les applications mobiles sont devenues des outils essentiels dans de nombreux domaines, notamment dans le secteur de l'immobilier et de la location de biens. Les méthodes traditionnelles de gestion des locations, souvent basées sur des démarches manuelles, des appels téléphoniques ou des échanges physiques, présentent plusieurs limites telles que le manque de transparence, la lenteur des processus et la difficulté de centralisation des informations.

Ce projet consiste à concevoir et développer une application mobile de location de biens immobiliers, permettant de mettre en relation des propriétaires et des locataires à travers une plateforme numérique moderne, sécurisée et performante. L'application offre aux utilisateurs la possibilité de consulter des annonces, d'ajouter des propriétés, de gérer des demandes de location, de recevoir des notifications et de publier des avis, le tout à partir d'un appareil mobile.

La solution proposée repose sur une architecture client-serveur bien structurée. La partie frontend est développée en React Native avec Expo, ce qui permet de créer une application mobile multiplateforme compatible avec Android et iOS. L'authentification des utilisateurs est assurée par Firebase Authentication, garantissant un haut niveau de sécurité et une gestion fiable des identités. La partie backend est implémentée à l'aide de ASP.NET Core, exposant des API REST responsables de la logique métier. Enfin, les données relatives aux propriétés, aux locations, aux images, aux avis et aux notifications sont stockées dans une base de données MongoDB, choisie pour sa flexibilité et sa capacité à gérer des données non structurées.

L'objectif principal de ce projet est de proposer une solution numérique complète qui simplifie la gestion des locations immobilières, améliore l'expérience utilisateur et met en œuvre des technologies modernes largement utilisées dans le monde professionnel. Ce travail permet également d'appliquer concrètement les connaissances acquises en développement mobile, en backend web, en bases de données NoSQL et en architectures logicielles.

Introduction

1. Contexte général

Le secteur de la location immobilière connaît une transformation importante grâce à la digitalisation. Aujourd'hui, de plus en plus d'utilisateurs recherchent des solutions rapides, accessibles et fiables pour trouver un logement ou proposer un bien à la location. Les plateformes numériques jouent un rôle clé dans cette évolution en facilitant l'accès à l'information, la communication entre les parties et la gestion des transactions.

Cependant, malgré l'existence de nombreuses plateformes, plusieurs problèmes persistent. Les utilisateurs peuvent être confrontés à des interfaces complexes, à des systèmes peu sécurisés ou à un manque de flexibilité dans la gestion des données. De plus, certaines solutions ne sont pas adaptées aux appareils mobiles ou ne proposent pas une expérience utilisateur fluide et intuitive.

C'est dans ce contexte que s'inscrit ce projet, qui vise à concevoir une application mobile moderne dédiée à la location de biens immobiliers, en mettant l'accent sur la simplicité d'utilisation, la sécurité et la performance.

2. Problématique

Les méthodes classiques de gestion de la location immobilière présentent plusieurs limites, parmi lesquelles :

- La difficulté de centraliser les informations relatives aux biens, aux locataires et aux propriétaires
- Le manque de transparence dans les échanges entre les utilisateurs
- L'absence de notifications en temps réel concernant les demandes de location ou les mises à jour
- Des systèmes d'authentification parfois peu sécurisés
- Une mauvaise adaptation aux usages mobiles

Ces contraintes rendent le processus de location plus lent et moins efficace, aussi bien pour les propriétaires que pour les locataires.
La problématique principale de ce projet peut donc être formulée comme suit :
Comment concevoir une application mobile sécurisée, performante et facile à utiliser permettant de gérer efficacement la location de biens immobiliers tout en assurant une séparation claire entre l'authentification des utilisateurs et la gestion des données métier ?

3. Choix technologiques

Les technologies utilisées dans ce projet ont été choisies en fonction de leur popularité, de leur performance et de leur adéquation avec les besoins de l'application :

- React Native avec Expo pour le développement de l'application mobile, permettant une rapidité de développement et une compatibilité multiplateforme
- Firebase Authentication pour la gestion sécurisée des utilisateurs
- ASP.NET Core pour le développement des API backend, offrant performance et modularité
- MongoDB comme base de données NoSQL, adaptée au stockage de données flexibles et évolutives

Ces choix permettent de mettre en place une solution moderne, conforme aux standards actuels du développement logiciel.

4.Objectifs du projet

L'objectif principal de ce projet est de développer une application mobile complète permettant de digitaliser et d'optimiser le processus de location immobilière. Cet objectif général se décline en plusieurs objectifs spécifiques :

- Concevoir une application mobile multiplateforme accessible depuis Android et iOS
- Mettre en place un système d'authentification sécurisé basé sur Firebase Authentication
- Développer un backend robuste en ASP.NET Core pour gérer la logique métier
- Utiliser MongoDB pour stocker et gérer les données liées aux propriétés, locations, images, avis et notifications
- Permettre aux utilisateurs d'ajouter, consulter et gérer des propriétés
- Gérer les images des propriétés via un système de stockage séparé
- Offrir un système de notifications informant les utilisateurs des événements importants
- Assurer une architecture claire et évolutive facilitant la maintenance et les futures améliorations

5. Organisation du rapport

Ce rapport est structuré de manière à présenter progressivement les différentes composantes du projet. Après cette introduction, le document abordera l'analyse des besoins, l'architecture générale du système, la conception de la partie frontend et backend, la gestion de la base de données, ainsi que les aspects liés à la sécurité, aux tests et aux perspectives d'amélioration.

Analyse des besoins

1. Introduction à l'analyse des besoins

L'analyse des besoins constitue une étape essentielle dans tout projet informatique, car elle permet de définir précisément les fonctionnalités attendues du système ainsi que les contraintes auxquelles il doit répondre. Une bonne analyse garantit que la solution développée correspond réellement aux attentes des utilisateurs finaux et aux objectifs fixés.
Dans le cadre de ce projet, l'analyse des besoins a été réalisée en tenant compte des différents profils d'utilisateurs de l'application de location immobilière, notamment les propriétaires, les locataires et les administrateurs. Cette analyse a permis d'identifier les fonctionnalités principales nécessaires au bon fonctionnement de l'application ainsi que les exigences non fonctionnelles liées à la sécurité, à la performance et à l'ergonomie.

2. Identification des acteurs

Le système proposé fait intervenir plusieurs acteurs, chacun ayant des rôles et des responsabilités spécifiques :

- Utilisateur (locataire) : personne cherchant à louer un bien immobilier à travers l'application.
- Utilisateur (propriétaire) : personne proposant un ou plusieurs biens immobiliers à la location.
- Administrateur : responsable de la gestion globale de la plateforme, de la modération du contenu et du suivi des activités.
- Système d'authentification (Firebase) : service externe assurant la gestion sécurisée des identités des utilisateurs.

Chaque acteur interagit avec le système selon ses droits et ses besoins, ce qui nécessite une gestion claire des rôles et des autorisations.

3. Besoins fonctionnels

Les besoins fonctionnels décrivent les différentes actions que le système doit permettre de réaliser. Ils définissent le comportement attendu de l'application du point de vue des utilisateurs.

3.1 Authentification et gestion des utilisateurs

- Permettre aux utilisateurs de s'inscrire et de se connecter à l'application
- Utiliser un système d'authentification sécurisé basé sur Firebase Authentication
- Gérer les sessions utilisateurs et la persistance de la connexion
- Associer chaque utilisateur authentifié à un identifiant unique (Firebase UID)
- Protéger l'accès aux fonctionnalités sensibles de l'application

3.2 Gestion des propriétés immobilières

- Permettre aux propriétaires d'ajouter de nouvelles propriétés
- Modifier ou supprimer des propriétés existantes
- Consulter la liste des propriétés disponibles
- Afficher les détails d'une propriété (description, prix, localisation, images)
- Associer chaque propriété à son propriétaire via l'identifiant utilisateur

3.3 Gestion des images

- Permettre l'ajout d'images pour chaque propriété
- Stocker les images dans une collection distincte de celle des propriétés
- Associer les images à une propriété via un identifiant de référence
- Assurer une bonne organisation et une récupération rapide des images
- Gérer les erreurs liées à l'envoi ou à l'affichage des images

3.4 Gestion des locations

- Permettre aux locataires d'envoyer des demandes de location
- Associer une demande de location à un utilisateur et à une propriété
- Permettre aux propriétaires d'accepter ou de refuser une demande
- Consulter l'historique des locations
- Mettre à jour le statut des locations (en attente, acceptée, refusée)

3.5 Notifications

- Informer les utilisateurs des événements importants
- Envoyer des notifications lors :
  - d'une nouvelle demande de location
  - d'une acceptation ou d'un refus
  - d'une mise à jour importante
- Afficher les notifications dans l'application mobile
- Stocker les notifications dans la base de données MongoDB

3.6 Avis et évaluations

- Permettre aux utilisateurs de laisser des avis après une location
- Associer un avis à une propriété et à un utilisateur
- Afficher les avis et les notes sur les propriétés
- Améliorer la transparence et la confiance entre utilisateurs

3.7 Administration

- Gérer les utilisateurs et les propriétés
- Modérer le contenu publié
- Accéder à une vue globale des données
- Assurer le bon fonctionnement général de la plateforme

4. Besoins non fonctionnels

Les besoins non fonctionnels décrivent les contraintes et les critères de qualité que le système doit respecter.

4.1 Sécurité

- Utilisation de Firebase Authentication pour sécuriser l'accès
- Validation des tokens Firebase côté backend
- Protection des routes API sensibles
- Séparation claire entre l'authentification et les données métier
- Prévention des accès non autorisés

4.2 Performance

- Temps de réponse rapide des API
- Chargement fluide des données sur l'application mobile
- Gestion efficace des requêtes vers la base MongoDB
- Optimisation de la gestion des images

4.3 Scalabilité

- Possibilité d'ajouter de nouveaux utilisateurs sans dégradation des performances
- Architecture évolutive permettant l'ajout de nouvelles fonctionnalités
- Utilisation de MongoDB pour supporter une croissance des données

4.4 Disponibilité et fiabilité

- Application accessible à tout moment
- Gestion des erreurs réseau
- Messages d'erreur clairs pour l'utilisateur
- Continuité du service en cas de forte utilisation

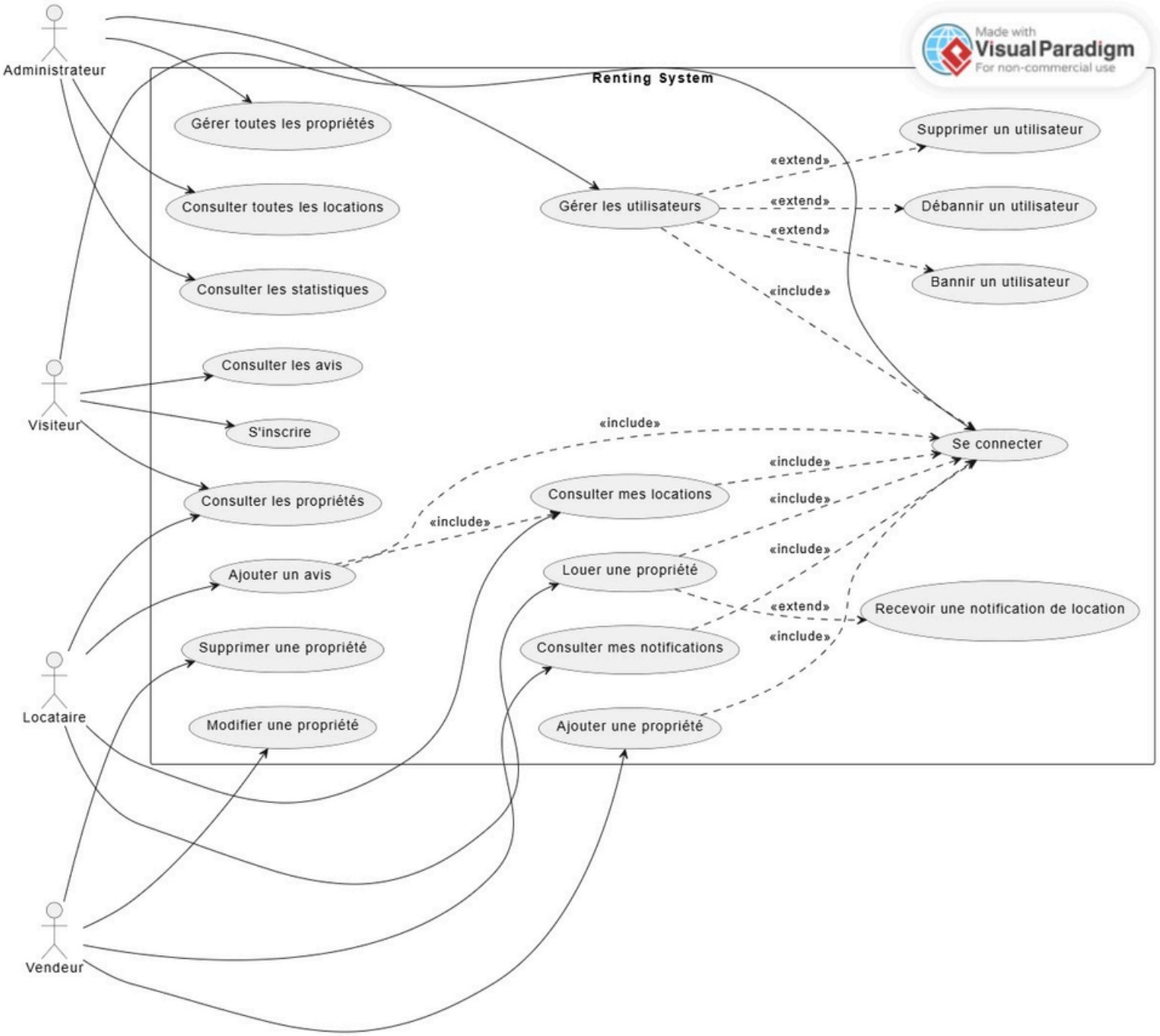
4.5 Ergonomie et expérience utilisateur

- Interface simple et intuitive
- Navigation fluide entre les écrans
- Design adapté aux appareils mobiles
- Réduction du nombre d'actions nécessaires pour effectuer une tâche

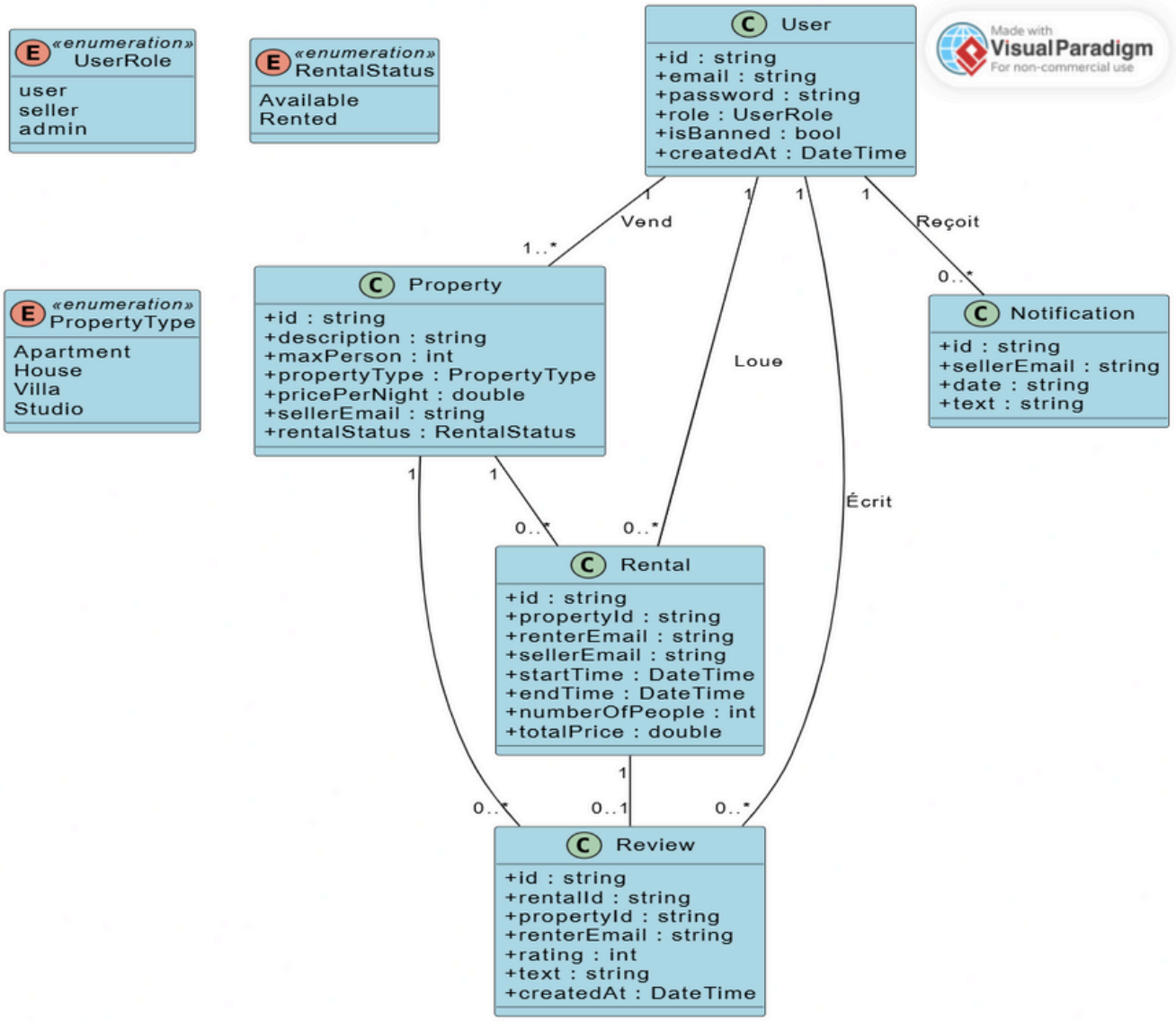
5. Conclusion de l’analyse des besoins

Cette analyse des besoins a permis de définir clairement les fonctionnalités attendues de l’application ainsi que les contraintes techniques et qualitatives à respecter. Elle constitue une base solide pour la conception et le développement du système. Les besoins identifiés guideront les choix architecturaux et technologiques présentés dans la suite de ce rapport.

use case


Administrateur
Visiteur
Locataire
Vendeur
Renting System
Gérer toutes les propriétés
Consulter toutes les locations
Consulter les statistiques
Consulter les avis
S'inscrire
Consulter les propriétés
Ajouter un avis
Supprimer une propriété
Modifier une propriété
Gérer les utilisateurs
Supprimer un utilisateur
Débannir un utilisateur
Bannir un utilisateur
Se connecter
Consulter mes locations
Louer une propriété
Consulter mes notifications
Ajouter une propriété
Recevoir une notification de location
«extend»
«include»
Made with VisualParadigm
For non-commercial use

class diagram



Architecture générale du système

1. Présentation générale de l'architecture

L'architecture du système joue un rôle fondamental dans la qualité, la maintenabilité et l'évolutivité d'une application. Dans le cadre de ce projet, une architecture client-serveur a été adoptée afin de séparer clairement les responsabilités entre la partie frontend, la gestion de l'authentification et la logique métier.

Cette architecture repose sur trois couches principales :

- une application mobile destinée aux utilisateurs finaux,
- un backend web chargé de traiter la logique métier,
- une base de données assurant la persistance des données.

Cette séparation permet de faciliter la maintenance du système, d'améliorer la sécurité et de rendre l'application évolutive.

2. Architecture globale du système

Le système est basé sur une architecture orientée services, dans laquelle l'application mobile communique avec le backend via des API REST. L'authentification des utilisateurs est déléguée à un service externe, Firebase Authentication, tandis que les données métier sont stockées dans une base de données MongoDB.

Le fonctionnement global peut être résumé comme suit :

1. L'utilisateur interagit avec l'application mobile développée en React Native (Expo).
2. L'utilisateur s'authentifie via Firebase Authentication.
3. Firebase génère un jeton d'authentification (Firebase ID Token).
4. L'application mobile envoie ce jeton dans les requêtes HTTP vers le backend.
5. Le backend ASP.NET Core valide le jeton et traite la requête.
6. Les données sont lues ou écrites dans la base MongoDB.

Cette architecture garantit une séparation claire entre l'identité des utilisateurs et la gestion des données métier.

3. Architecture de la partie frontend

La partie frontend est une application mobile développée en React Native en utilisant le framework Expo. Ce choix permet de développer une application multiplateforme à partir d'un seul code source, compatible avec les systèmes Android et iOS.
L'application mobile est structurée autour de plusieurs composants réutilisables et écrans, facilitant la navigation et l'expérience utilisateur. La gestion de l'état global, notamment l'état d'authentification, est assurée par le Context API de React.
Le frontend est responsable des tâches suivantes :

- Affichage des interfaces utilisateur
- Gestion des interactions utilisateurs
- Communication avec Firebase pour l'authentification
- Envoi des requêtes vers le backend via des API REST

4. Architecture de l'authentification

L'authentification est assurée par Firebase Authentication, un service cloud fiable et sécurisé. Ce choix permet d'éviter la gestion manuelle des mots de passe et de réduire les risques liés à la sécurité.
Firebase prend en charge :

- L'inscription des utilisateurs
- La connexion et la déconnexion
- La gestion des sessions
- La génération de jetons sécurisés

Le backend ne stocke aucune information sensible liée à l'authentification. Il se contente de vérifier la validité du jeton fourni par le client, garantissant ainsi un haut niveau de sécurité.

5. Architecture de la partie backend

Le backend est développé en ASP.NET Core, un framework moderne et performant permettant de créer des services web robustes. Il expose un ensemble d'API REST organisées autour de contrôleurs, chacun responsable d'un domaine fonctionnel spécifique.
Les principales responsabilités du backend sont :

- La validation des requêtes entrantes
- La vérification des jetons Firebase
- La gestion de la logique métier
- La communication avec la base de données MongoDB
- La gestion des erreurs et des exceptions

Cette approche modulaire permet une meilleure organisation du code et facilite les évolutions futures.

6. Architecture de la base de données

La persistance des données est assurée par MongoDB, une base de données NoSQL orientée documents. Ce choix est particulièrement adapté à ce projet en raison de la diversité et de la flexibilité des données manipulées.
Les données sont organisées en collections distinctes, telles que :

- Propriétés
- Images
- Locations
- Avis
- Notifications

Chaque collection est indépendante mais liée aux autres par des identifiants, notamment l'identifiant utilisateur fourni par Firebase Authentication.

7. Avantages de l’architecture adoptée

L’architecture choisie présente plusieurs avantages :

- Séparation claire des responsabilités
- Sécurité renforcée grâce à Firebase Authentication
- Facilité de maintenance et d’évolution
- Scalabilité du système
- Compatibilité avec des applications mobiles modernes

8. Conclusion de l’architecture générale

Cette architecture générale constitue une base solide pour le développement de l’application de location immobilière. Elle répond aux besoins identifiés tout en garantissant performance, sécurité et évolutivité. Les sections suivantes du rapport détailleront les aspects de conception, d’implémentation et de validation du système.

Conception de la base de données (MongoDB)

1. Introduction à la conception de la base de données

La base de données constitue un élément central de toute application informatique, car elle assure la persistance et la cohérence des données. Dans ce projet, le choix s'est porté sur MongoDB, une base de données NoSQL orientée documents, particulièrement adaptée aux applications modernes nécessitant flexibilité, performance et évolutivité.

La conception de la base de données a été réalisée en tenant compte de l'architecture globale du système, de la séparation entre l'authentification et les données métier, ainsi que des besoins fonctionnels identifiés précédemment. Contrairement aux bases de données relationnelles, MongoDB permet de stocker des données sous forme de documents JSON, ce qui facilite la gestion de structures complexes et évolutives.

2. Justification du choix de MongoDB

Le choix de MongoDB repose sur plusieurs critères techniques et fonctionnels :

- Flexibilité du schéma : les documents peuvent évoluer sans modification stricte du schéma
- Adaptation aux données non structurées : images, notifications, avis, etc.
- Performance élevée pour les opérations de lecture et d'écriture
- Scalabilité horizontale, permettant de gérer un grand volume de données
- Bonne intégration avec ASP.NET Core

MongoDB est donc particulièrement adapté à une application de location immobilière où les données peuvent varier d'un utilisateur à un autre.

3. Organisation générale de la base de données

La base de données est organisée en collections, chacune représentant un domaine fonctionnel spécifique de l'application. Chaque collection est indépendante, mais des relations logiques sont établies à l'aide d'identifiants, notamment l'identifiant utilisateur fourni par Firebase Authentication.

Il est important de souligner que :

- Les informations d'authentification ne sont pas stockées dans MongoDB
- L'identification des utilisateurs repose uniquement sur le Firebase UID

4. Description des principales collections

4.1 Collection « Properties »

Cette collection stocke les informations relatives aux biens immobiliers proposés à la location.

Principaux attributs :

- Identifiant de la propriété
- Titre et description
- Prix de location
- Localisation
- Identifiant du propriétaire (Firebase UID)
- Date de création
- Statut de disponibilité

Chaque propriété est associée à un utilisateur propriétaire via son identifiant Firebase.

4.2 Collection « Images »

Les images des propriétés sont stockées dans une collection séparée afin de garantir une meilleure organisation des données et une plus grande flexibilité.

Principaux attributs :

- Identifiant de l'image
- URL ou chemin de l'image
- Identifiant de la propriété associée
- Date d'ajout

4.3 Collection « Rentals »

La collection des locations permet de gérer les demandes et l'historique des locations.

Principaux attributs :

- Identifiant de la location
- Identifiant de la propriété
- Identifiant du locataire (Firebase UID)
- Statut de la location (en attente, acceptée, refusée)
- Dates de début et de fin
- Date de création de la demande

Cette collection joue un rôle clé dans la gestion du processus de location.

4.4 Collection « Reviews »

Les avis et évaluations sont stockés dans une collection dédiée afin d'améliorer la transparence et la confiance entre utilisateurs.

Principaux attributs :

- Identifiant de l'avis
- Identifiant de la propriété
- Identifiant de l'utilisateur ayant laissé l'avis
- Note attribuée
- Commentaire
- Date de publication

Les avis permettent aux futurs locataires de se faire une idée sur la qualité des biens proposés.

4.5 Collection « Notifications »

La collection des notifications permet d'informer les utilisateurs des événements importants liés à leurs activités.

Principaux attributs :

- Identifiant de la notification
- Identifiant de l'utilisateur concerné
- Type de notification
- Message
- Statut (lu / non lu)
- Date de création

Les notifications améliorent l'expérience utilisateur en fournissant des informations en temps réel.

5. Relations entre les collections

Même si MongoDB ne repose pas sur des relations strictes comme dans les bases de données relationnelles, des relations logiques existent entre les collections :

- Un utilisateur peut posséder plusieurs propriétés
- Une propriété peut avoir plusieurs images
- Une propriété peut être associée à plusieurs locations
- Un utilisateur peut recevoir plusieurs notifications
- Une propriété peut recevoir plusieurs avis

Ces relations sont gérées à l'aide d'identifiants partagés, principalement le Firebase UID et les identifiants des documents MongoDB.

6. Sécurité et intégrité des données

La sécurité des données est assurée à plusieurs niveaux :

- Accès à la base de données contrôlé par le backend
- Validation des requêtes côté serveur
- Association stricte des données à l'utilisateur authentifié
- Absence de stockage des mots de passe ou données sensibles

Cette approche garantit l'intégrité des données et limite les risques d'accès non autorisé.

7. Conclusion sur la conception de la base de données

La conception de la base de données MongoDB répond efficacement aux besoins du projet. Elle offre une structure flexible, évolutive et sécurisée, parfaitement adaptée à une application mobile moderne. Cette conception facilite également l'intégration avec le backend ASP.NET Core et permet une gestion efficace des données métier.

Conception de la partie frontend

1. Introduction à la conception du frontend

La partie frontend représente l'interface entre l'utilisateur et le système. Elle joue un rôle essentiel dans la qualité de l'expérience utilisateur, car elle est responsable de l'affichage des informations, de la navigation et de l'interaction avec les différentes fonctionnalités de l'application. Dans ce projet, le frontend est implémenté sous la forme d'une application mobile développée avec React Native et le framework Expo.

Le choix d'une application mobile répond aux besoins actuels des utilisateurs, qui privilégient l'utilisation des smartphones pour accéder aux services numériques. L'objectif principal du frontend est de proposer une interface intuitive, fluide et adaptée aux contraintes des appareils mobiles, tout en assurant une communication efficace avec les services backend.

2. Choix de React Native et Expo

React Native permet de développer des applications mobiles multiplateformes à partir d'un seul code source JavaScript. Ce choix présente plusieurs avantages :

- Développement rapide et efficace
- Partage du code entre Android et iOS
- Large communauté et écosystème riche
- Performances proches des applications natives

Expo a été utilisé afin de simplifier la configuration du projet et l'accès aux fonctionnalités natives du téléphone, telles que la gestion des images, les notifications et l'accès aux ressources du système.

3. Architecture générale de l’application mobile

L’application mobile est structurée de manière modulaire afin de faciliter la maintenance et l’évolution du code. Elle est composée de plusieurs éléments principaux :

- Écrans (Screens) : représentent les différentes pages de l’application
- Composants (Components) : éléments réutilisables de l’interface utilisateur
- Context API : gestion de l’état global, notamment l’état d’authentification
- Services : gestion des appels API vers le backend
- Configuration : paramètres globaux de l’application

Cette organisation permet une séparation claire des responsabilités au sein du frontend.

4. Gestion de l’authentification côté frontend

L’authentification des utilisateurs est entièrement gérée côté frontend à l’aide de Firebase Authentication. Ce choix permet de garantir un haut niveau de sécurité tout en simplifiant la gestion des utilisateurs.

Le processus d’authentification comprend :

- L’inscription des nouveaux utilisateurs
- La connexion des utilisateurs existants
- La gestion de la session utilisateur
- La récupération du jeton d’authentification (Firebase ID Token)

Le jeton généré par Firebase est ensuite inclus dans les en-têtes des requêtes HTTP envoyées vers le backend, afin de sécuriser l’accès aux ressources.

5. Gestion de l’état et navigation

La gestion de l’état global de l’application est assurée par le Context API de React. Cette approche permet de partager facilement les informations liées à l’utilisateur authentifié entre les différents composants de l’application.
La navigation entre les écrans est conçue de manière à offrir une expérience fluide et intuitive. Les utilisateurs peuvent naviguer facilement entre :

- L’écran d’accueil
- La liste des propriétés
- Les détails d’une propriété
- Le formulaire d’ajout de propriété
- Les notifications
- Le profil utilisateur

Cette organisation favorise une prise en main rapide de l’application.

6. Conception des principales interfaces utilisateur

6.1 Écran de consultation des propriétés
Cet écran permet aux utilisateurs de visualiser la liste des biens disponibles à la location. Chaque propriété est affichée sous forme de carte contenant les informations essentielles telles que le titre, le prix et une image représentative.
6.2 Ajout d’une propriété
L’ajout d’une propriété se fait via un formulaire accessible depuis l’application. L’utilisateur peut saisir les informations nécessaires et ajouter des images. Les images sont envoyées séparément vers le backend, puis associées à la propriété.
6.3 Gestion des notifications
Un écran ou une fenêtre modale permet à l’utilisateur de consulter les notifications reçues. Ces notifications informent l’utilisateur des événements importants liés à ses activités, tels que les demandes de location ou les mises à jour de statut.

7. Communication avec le backend

La communication entre le frontend et le backend se fait via des requêtes HTTP utilisant des API REST. Chaque requête est sécurisée par l'ajout du jeton Firebase dans l'en-tête d'autorisation.

Le frontend est responsable de :

- L'envoi des requêtes
- Le traitement des réponses
- La gestion des erreurs
- L'affichage des messages à l'utilisateur

Cette communication permet d'assurer une synchronisation efficace entre l'interface utilisateur et les données stockées dans la base MongoDB.

8. Ergonomie et expérience utilisateur

Une attention particulière a été portée à l'ergonomie de l'application. L'interface est conçue pour être :

- Simple et intuitive
- Adaptée aux écrans mobiles
- Facile à naviguer
- Réactive et fluide

L'objectif est d'améliorer l'expérience utilisateur et de rendre l'application accessible à un large public.

9. Conclusion sur la conception du frontend

La conception de la partie frontend permet de proposer une application mobile moderne, performante et intuitive. Grâce à React Native et Expo, l'application est multiplateforme et facilement évolutive. Cette partie constitue un élément clé du projet et contribue fortement à la satisfaction des utilisateurs.

Conception de la partie backend

1. Introduction à la conception du backend

La partie backend constitue le cœur logique de l'application. Elle est responsable du traitement des requêtes envoyées par l'application mobile, de l'application des règles métier, de la communication avec la base de données et de la sécurisation des accès aux ressources. Dans ce projet, le backend est développé en ASP.NET Core, un framework moderne, performant et largement utilisé dans le développement d'API web.

Le backend joue un rôle d'intermédiaire entre le frontend mobile et la base de données MongoDB. Il garantit la cohérence des données, applique les règles de gestion et assure la sécurité du système.

2. Choix de ASP.NET Core

Le choix de ASP.NET Core repose sur plusieurs avantages techniques :

- Framework multiplateforme et open source
- Excellentes performances pour les API REST
- Architecture modulaire et extensible
- Bonne intégration avec MongoDB
- Gestion efficace de la sécurité et des middlewares

ASP.NET Core est donc particulièrement adapté au développement d'un backend robuste pour une application mobile moderne.

3. Architecture générale du backend

Le backend est structuré selon une architecture claire et organisée, favorisant la séparation des responsabilités. Il est composé principalement des éléments suivants :

- Controllers : exposent les points d'accès (endpoints) de l'API
- Models : représentent les entités métier de l'application
- Services : gèrent l'accès à la base de données et la logique technique
- Configuration : fichiers de paramétrage (appsettings)

Cette organisation permet une meilleure lisibilité du code et facilite sa maintenance.

4. Les contrôleurs (Controllers)

4.1 AuthController

Bien que l'authentification soit assurée par Firebase, ce contrôleur est utilisé pour :

- Valider les jetons Firebase envoyés par le client
- Gérer les accès sécurisés aux ressources
- Centraliser la logique de sécurité côté backend

4.2 PropertyController

Ce contrôleur gère toutes les opérations liées aux propriétés immobilières :

- Ajout de nouvelles propriétés
- Modification et suppression
- Consultation de la liste des propriétés
- Récupération des détails d'une propriété

Il assure également l'association des propriétés à leurs propriétaires via l'identifiant Firebase.

4.3 ImageController

Le contrôleur des images est responsable de la gestion des images associées aux propriétés. Les images sont stockées séparément afin d'optimiser la structure des données.

Ses principales fonctions sont :

- Téléversement des images
- Association des images à une propriété
- Récupération des images

4.4 RentalController

Ce contrôleur gère le processus de location :

- Création des demandes de location
- Acceptation ou refus par le propriétaire
- Mise à jour du statut des locations
- Consultation de l'historique des locations

Il joue un rôle central dans la logique métier de l'application.

4.5 ReviewController

Le contrôleur des avis permet :

- L'ajout d'avis et de notes
- La consultation des avis d'une propriété
- L'amélioration de la transparence entre utilisateurs

4.6 NotificationsController

Ce contrôleur est chargé de :

- Créer des notifications lors d'événements importants
- Récupérer les notifications d'un utilisateur
- Mettre à jour le statut des notifications (lues / non lues)

4.7 AdminController

Le contrôleur administrateur permet :

- La gestion globale des données
- La modération du contenu
- Le suivi de l'activité de la plateforme

5. Les modèles (Models)

Les modèles représentent les entités principales du système, telles que :

- Propriété
- Image
- Location
- Avis
- Notification

Chaque modèle correspond à une collection MongoDB et définit la structure des documents stockés dans la base de données.

6. Les services et l'accès à la base de données

L'accès à MongoDB est centralisé au sein d'un service dédié, permettant :

- La gestion de la connexion à la base de données
- L'accès aux différentes collections
- La réutilisation du code

Cette approche améliore la maintenabilité et réduit la duplication du code

7. Sécurité côté backend

La sécurité est un aspect essentiel du backend. Elle est assurée par :

- La validation des jetons Firebase
- La protection des endpoints sensibles
- Le contrôle des autorisations selon l'utilisateur
- La gestion des erreurs et des exceptions

Le backend ne stocke aucune donnée sensible liée à l'authentification, ce qui renforce la sécurité globale du système.

8. Gestion des erreurs et des réponses

Le backend fournit des réponses claires et structurées au frontend :

- Codes HTTP appropriés
- Messages d'erreur explicites
- Gestion des exceptions

Cette approche permet au frontend de gérer efficacement les erreurs et d'informer l'utilisateur en cas de problème.

9. Conclusion sur la conception du backend

La conception de la partie backend repose sur une architecture robuste, sécurisée et évolutive. ASP.NET Core permet de répondre efficacement aux besoins du projet tout en garantissant de bonnes performances et une intégration fluide avec MongoDB et Firebase Authentication.

Flux de l'Application

1. Flux d'authentification utilisateur

- Ouverture de l'application : Vérification du token stocké dans AsyncStorage
- Si non authentifié :
- Redirection vers l'écran de connexion
- L'utilisateur saisit email et mot de passe
- Envoi des credentials à Firebase Authentication
- Réception du token JWT
- Stockage du token localement
- Récupération du profil utilisateur depuis l'API
- Si authentifié : Redirection directe vers l'écran d'accueil

2. Processus d'ajout de propriété

- Navigation : Le propriétaire accède à "Ajouter une annonce"
- Formulaire : Remplissage des informations (titre, description, prix, adresse, type, capacité)
- Upload d'images :
- Sélection des photos depuis la galerie ou l'appareil photo
- Prévisualisation des images sélectionnées
- Upload vers le serveur
- Validation : Vérification des champs obligatoires côté client
- Envoi : Requête POST vers /api/properties
- Confirmation : Message de succès et redirection vers la liste des annonces

3.Flux upload d'images puis création de propriété

- Ce workflow en deux temps optimise l'expérience utilisateur :
- Phase 1 - Upload des images :
- Upload immédiat des images vers le serveur
- Récupération des IDs des images uploadées
- Phase 2 - Création de la propriété :
- Soumission du formulaire avec les IDs d'images
- Création du document Property avec références aux images
- Liaison automatique des images à la propriété

4.Cycle de vie d'une demande de location

- Recherche : Le locataire filtre et trouve une propriété
- Consultation : Consultation des détails, photos, avis
- Réservation :
- Sélection des dates de début et fin
- Calcul automatique du prix total
- Vérification de la disponibilité
- Confirmation de la réservation
- Notification propriétaire : Notification push envoyée au propriétaire
- Validation : Le propriétaire accepte ou refuse
- Notification locataire : Confirmation de statut envoyée
- Post-location : Possibilité de laisser un avis après le séjour

5.Système de livraison des notifications

- Trigger : Événements (nouvelle réservation, validation, message)
- Backend : Création d'un document Notification dans MongoDB
- Firebase Cloud Messaging : Envoi de la notification push
- Réception mobile : Affichage de la notification sur l'appareil
- Stockage local : Sauvegarde dans l'application pour consultation ultérieure
- Badge : Indicateur visuel du nombre de notifications non lues

Limites et Défis

1. Synchronisation image-propriété

- Un des principaux défis techniques rencontrés concerne la synchronisation entre les images et les propriétés :
- Problème : En cas d'échec lors de la création d'une propriété, les images peuvent rester orphelines dans la base
- Impact : Accumulation de données inutiles et gaspillage d'espace de stockage
- Solution partielle : Mise en place d'un système de nettoyage périodique (cron job) pour supprimer les images non liées
- Amélioration future : Implémenter un système de transaction distribuée ou un mécanisme de rollback

2.Complexité de validation des tokens

- Malgré nos efforts, certaines limitations persistent :
- Performance : Temps de chargement sur connexions lentes (3G)
- Galerie d'images : Pas de zoom avancé ou de visualisation 360°
- Filtres : Options de filtrage limitées (pas de fourchette de prix, distance exacte)
- Accessibilité : Support limité des lecteurs d'écran et modes d'accessibilité
- Langues : Application uniquement en français

Améliorations Futures

1.Intégration de systèmes de paiement

- Prochaine étape majeure pour finaliser l'expérience de location :
- Stripe/PayPal : Intégration d'APIs de paiement sécurisées
- Paiement fractionné : Possibilité de payer en plusieurs fois
- Caution : Système de dépôt de garantie avec déblocage automatique
- Reçus automatiques : Génération et envoi de factures par email
- Multi-devises : Support de plusieurs monnaies pour locations internationales

2.Cartes et géolocalisation

- Amélioration de l'expérience de recherche avec des fonctionnalités géographiques :
- Google Maps / Mapbox : Intégration d'une carte interactive
- Vue carte : Affichage des propriétés sur une carte avec markers
- Recherche par zone : Dessin d'une zone de recherche sur la carte
- Calcul de distance : Distance entre la propriété et des points d'intérêt
- Itinéraires : Génération d'itinéraires vers la propriété

3.Système de messagerie instantanée

- Communication directe entre locataires et propriétaires :
- Chat en temps réel : Utilisation de Socket.io ou Firebase Realtime Database
- Historique : Conservation des conversations
- Notifications : Alertes pour nouveaux messages
- Partage de fichiers : Envoi de documents (contrats, pièces d'identité)
- Traduction automatique : Pour faciliter les échanges internationaux

Conclusion

1.Synthèse des réalisations

- Ce projet de plateforme de location immobilière mobile représente l'aboutissement de plusieurs mois de travail et d'apprentissage. Nous avons réussi à créer une application React Native complète et fonctionnelle qui répond aux besoins réels des utilisateurs dans le secteur de la location.
- Les fonctionnalités principales ont été implémentées avec succès :
- Un système d'authentification sécurisé avec Firebase
- Une gestion complète des propriétés avec upload d'images
- Un processus de réservation fluide et intuitif
- Un système d'avis pour garantir la transparence
- Une interface administrateur pour la modération
- L'architecture technique choisie (React Native + ASP.NET Core + MongoDB) s'est avérée robuste et scalable, nous permettant d'itérer rapidement tout en maintenant la qualité du code.

2.Compétences acquises

- Ce projet nous a permis de développer et consolider de nombreuses compétences techniques et transversales :
- Compétences techniques :
- Maîtrise de React Native et de son écosystème (Navigation, Hooks, Context API)
- Développement d'APIs RESTful avec ASP.NET Core
- Gestion de bases de données NoSQL avec MongoDB
- Intégration de services Firebase (Authentication, Cloud Messaging)
- Gestion d'état et optimisation des performances mobile
- Manipulation d'images et gestion du stockage

- Compétences méthodologiques :
- Architecture logicielle et design patterns
- Conception UX/UI adaptée au mobile
- Gestion de projet agile et versioning (Git)
- Tests et débogage d'applications mobiles
- Documentation technique
- Soft skills :
- Travail en équipe et communication
- Résolution de problèmes complexes
- Autonomie et recherche de solutions
- Gestion du temps et respect des deadlines

3.Impact du projet

- Au-delà de l'aspect technique, ce projet a un impact concret :
- Pour les utilisateurs :
- Facilite la recherche et la location de biens immobiliers
- Offre une transparence totale grâce aux avis
- Simplifie la gestion pour les propriétaires
- Expérience mobile moderne et intuitive
- Pour notre formation :
- Application pratique des concepts théoriques appris
- Préparation aux exigences du monde professionnel
- Constitution d'un portfolio démontrant nos compétences
- Expérience de développement full-stack mobile

4.Perspective professionnelle

- Cette plateforme pourrait évoluer vers un produit commercial viable. Les prochaines étapes incluent l'intégration de paiements, l'ajout de fonctionnalités avancées et potentiellement une phase de test utilisateur réel pour valider le concept sur le marché.
- En conclusion, ce projet nous a permis de transformer une idée en une application fonctionnelle, tout en développant des compétences essentielles pour notre carrière dans le développement mobile.

Références

Firebase Documentation

- Firebase Authentication Guide: https://firebase.google.com/docs/auth
- Firebase Cloud Messaging (FCM): https://firebase.google.com/docs/cloud-messaging
- Firebase Console: https://console.firebase.google.com
- Firebase pour React Native: https://rnfirebase.io

ASP.NET Core Documentation

- ASP.NET Core Web API: https://docs.microsoft.com/aspnet/core/web-api
- Entity Framework Core: https://docs.microsoft.com/ef/core
- ASP.NET Core Security: https://docs.microsoft.com/aspnet/core/security
- Middleware ASP.NET Core: https://docs.microsoft.com/aspnet/core/fundamentals/middleware

MongoDB Documentation

- MongoDB Manual: https://docs.mongodb.com/manual
- MongoDB with .NET Driver: https://docs.mongodb.com/drivers/csharp
- MongoDB Schema Design: https://docs.mongodb.com/manual/data-modeling

React Native & Expo Documentation

- React Native Official Docs: https://reactnative.dev/docs/getting-started
- MongoDB Atlas (Cloud): https://docs.atlas.mongodb.co
- Expo Documentation: https://docs.expo.dev
- React Navigation: https://reactnavigation.org/docs/getting-started
- Expo Image Picker: https://docs.expo.dev/versions/latest/sdk/imagepicker
- AsyncStorage: https://react-native-async-storage.github.io/async-storage

Bibliothèques et outils utilisés

- React Native Paper (UI Components): https://callstack.github.io/react-native-paper
- Axios (HTTP Client): https://axios-http.com/docs/intro
- React Hook Form: https://react-hook-form.com
- Date-fns (Date manipulation): https://date-fns.org

Ressources d'apprentissage

- React Native School: https://www.reactnativeschool.com
- Microsoft Learn - ASP.NET Core: https://learn.microsoft.com/training/aspnetcore
- MongoDB University: https://university.mongodb.com
- Firebase Tutorials: https://firebase.google.com/codelabs

Articles et tutoriels de référence

- "Building RESTful APIs with ASP.NET Core" - Microsoft Docs
- "React Native Performance Optimization" - React Native Blog
- "NoSQL Data Modeling" - MongoDB Blog
- "Mobile Authentication Best Practices" - Firebase Blog
- "Image Optimization for Mobile Apps" - Medium

Outils de développement

- Visual Studio Code: https://code.visualstudio.com
- Visual Studio 2022: https://visualstudio.microsoft.com
- MongoDB Compass: https://www.mongodb.com/products/compass
- Postman (API Testing): https://www.postman.com
- Android Studio: https://developer.android.com/studio
- Xcode (pour iOS): https://developer.apple.com/xcode

Ressources complémentaires

- Stack Overflow: https://stackoverflow.com
- GitHub Repositories (projets similaires pour inspiration)
- React Native Community: https://github.com/react-native-community
- MDN Web Docs: https://developer.mozilla.org