

## **Projet Microservices**

Ary MOTHIEU, Suthan THAIYAPARAN, Léopold MAURICE,  
Fardeen POOREEA, Janik JIANG



## Table des matières

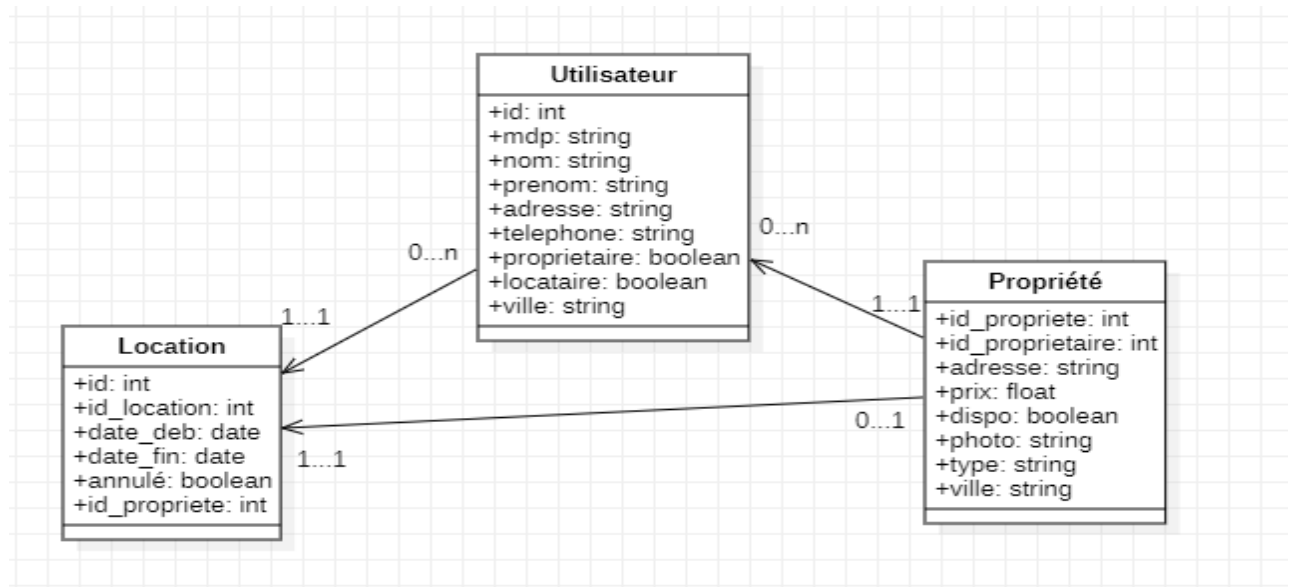
Introduction.....	3
Conception du projet .....	4
Découpage des services .....	6
Description de chacun des services.....	7
Service de recherche de biens immobiliers .....	7
Service de gestion des propriétés .....	8
Service de réservations de maisons .....	9
Les technologies utilisées.....	10
ReactJS pour le front-end.....	10
Pour le back-end.....	10
Gestion de déploiement du projet.....	11
Utilisation de docker .....	11
Déploiement de notre application .....	11
Démonstration .....	12

## Introduction

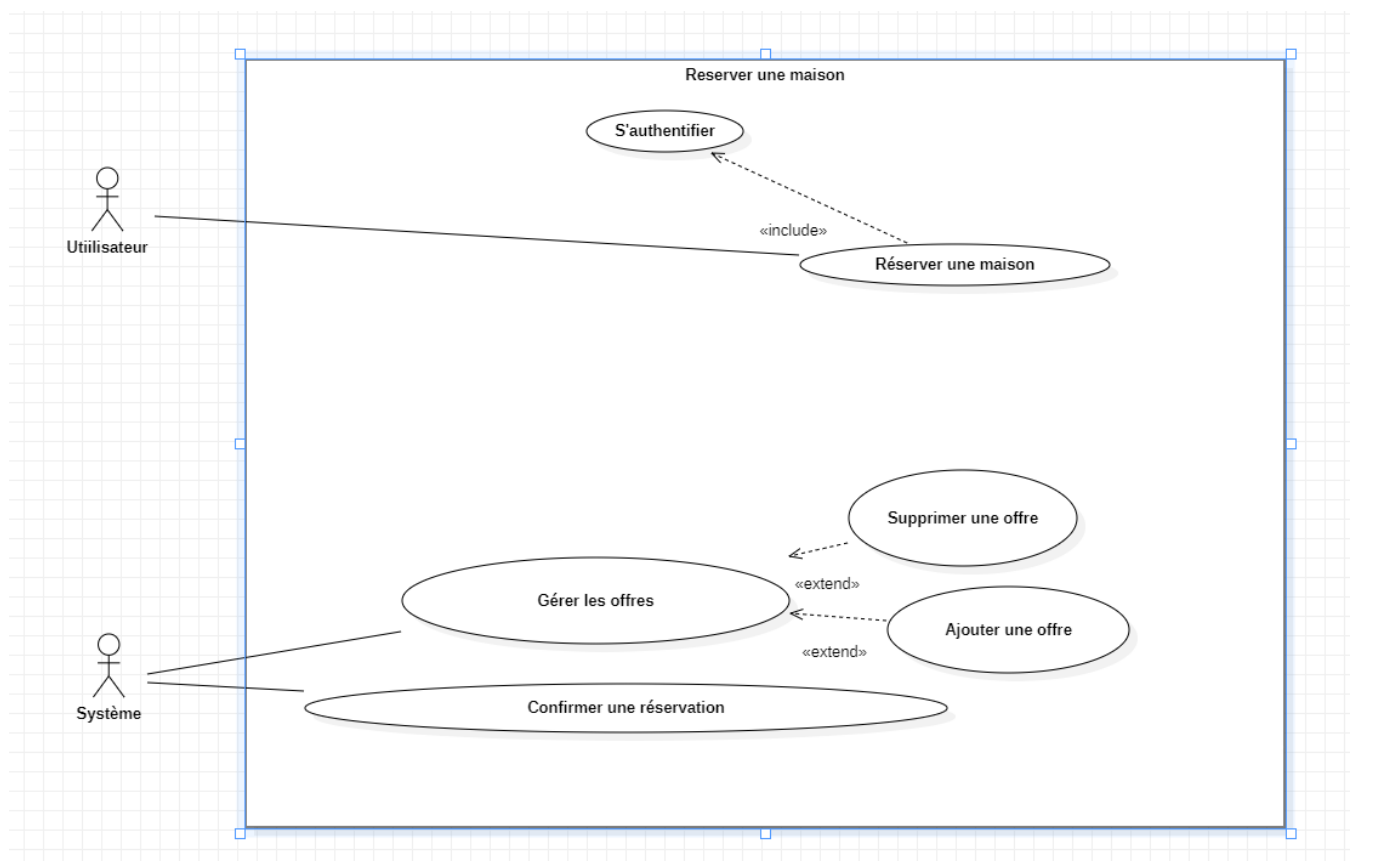
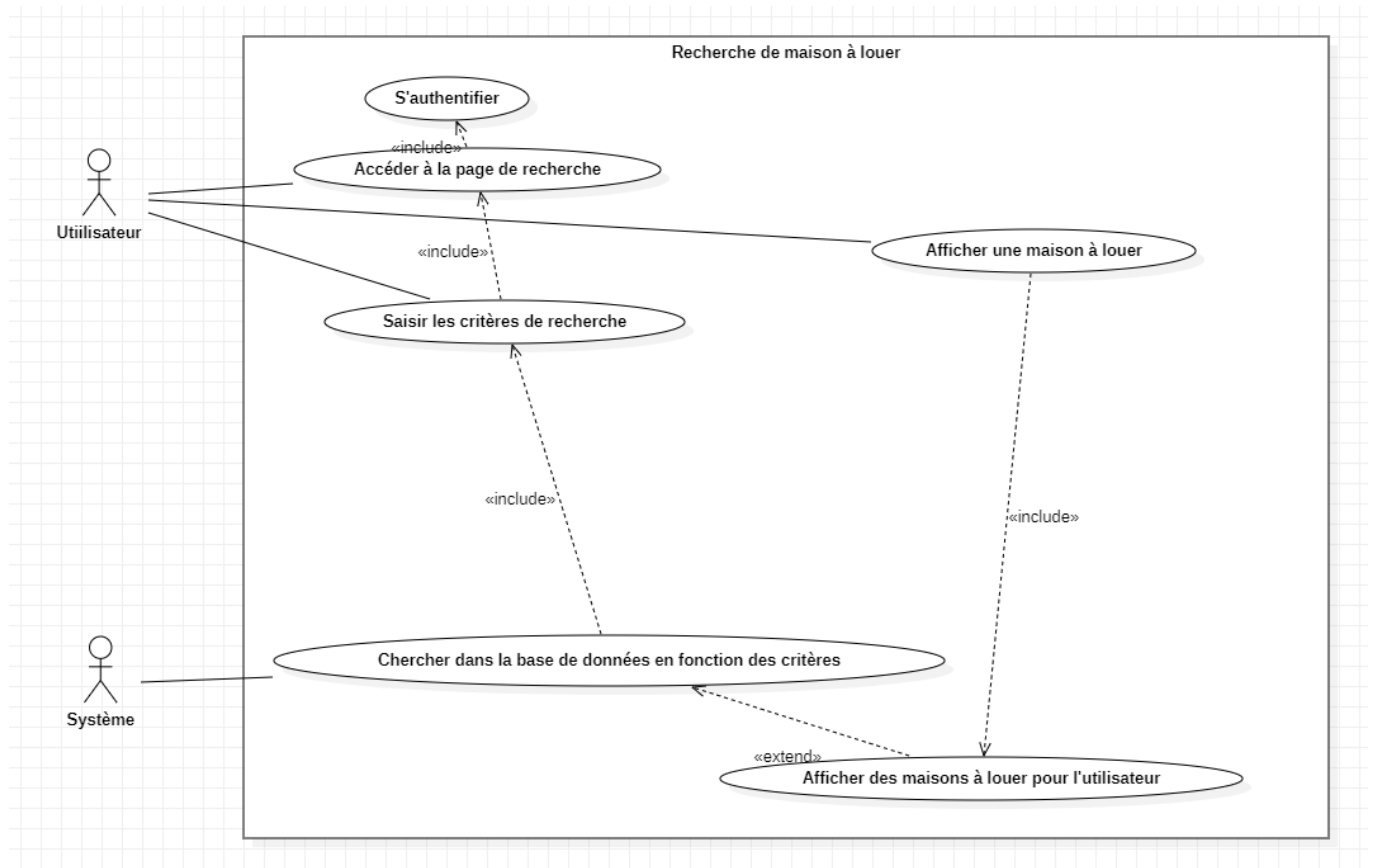
Bienvenue dans ce rapport consacré à la création d'une application de location de logement basée sur l'architecture des microservices. L'objectif principal de ce projet est de proposer une solution facile à utiliser et offrant une expérience fluide pour les locataires et les propriétaires. Nous allons présenter les fonctionnalités que nous avons prévues pour cette application, notamment la personnalisation de l'interface utilisateur, la recherche de logements disponibles par type, emplacement et budget, la gestion des demandes d'inscription et des offres de location, la réservation de logement, ainsi que le suivi des demandes et des processus de location. Nous avons la liberté de choisir les technologies à utiliser pour la réalisation de cette application, que ce soit pour le stockage de données, le développement (Front, Back) et le déploiement. Nous allons également préciser les livrables attendus à la fin du projet.

## Conception du projet

L'objectif premier a été de faire un diagramme des classes pour modéliser notre application, voici le diagramme des classes que nous avons établis pour cette application :



Voici les différents cas d'utilisations pour nos microservices :



## Découpage des services

Tout d'abord, il est important de comprendre ce qu'est une architecture de microservices. Dans une architecture de microservices, une application est divisée en plusieurs petits services autonomes qui travaillent ensemble pour fournir une fonctionnalité complète. Chaque service est responsable d'une tâche spécifique et communique avec les autres services par le biais d'API. Dans notre cas, nous avons découpé notre application comme ceci :

- Un service de recherche de biens immobiliers : ce service serait responsable de la recherche de biens immobiliers disponibles à la vente ou à la location. Il pourrait se connecter à une base de données de biens immobiliers et offrir une interface de recherche aux utilisateurs de l'application.
- Un service de gestion des propriétés : ce service serait responsable de la gestion des propriétés que l'agence a en gestion. Il pourrait offrir une interface de gestion des contrats de location, des informations sur les locataires et des détails sur les propriétés.
- Un service de réservation de maison : ce service serait responsable de la réservation de visites pour les propriétés. Les utilisateurs pourraient utiliser l'interface de réservation pour planifier une visite d'un bien immobilier qui les intéresse.

Chacun de ces services serait développé indépendamment et fonctionnerait comme une application autonome, avec son propre système de base de données et son propre code source. Les services communiqueraient entre eux par le biais d'API, ce qui permettrait une intégration facile et une évolutivité maximale.

# Description de chacun des services

## Service de recherche de biens immobiliers

Le service de recherche de biens immobiliers serait responsable de permettre aux utilisateurs de l'application de rechercher des biens immobiliers disponibles à la vente ou à la location. Pour ce faire, il utiliserait un système de base de données qui stockerait les informations sur les biens immobiliers disponibles, telles que l'emplacement, la taille, le prix et les caractéristiques.

Le service de recherche de biens immobiliers serait conçu pour être évolutif et résilient, ce qui signifie qu'il serait capable de gérer une charge élevée d'utilisateurs et de continuer à fonctionner même en cas de panne d'un des composants du système.

Voici quelques éléments clés que le service de recherche de biens immobiliers pourrait inclure :

**Interface utilisateur :** Le service de recherche de biens immobiliers offrirait une interface utilisateur permettant aux utilisateurs de rechercher des biens immobiliers en fonction de critères tels que l'emplacement, la taille et le prix. Cette interface utilisateur serait facile à utiliser et permettrait aux utilisateurs de filtrer les résultats de recherche en fonction de leurs besoins.

**Système de recherche :** Le service de recherche de biens immobiliers serait équipé d'un système de recherche performant, qui serait capable de rechercher rapidement et efficacement les biens immobiliers en fonction des critères de recherche spécifiés par l'utilisateur. Ce système de recherche serait conçu pour être évolutif et capable de gérer une charge importante.

**Base de données :** Le service de recherche de biens immobiliers serait connecté à une base de données qui stockerait les informations sur les biens immobiliers disponibles à la vente ou à la location. Cette base de données serait conçue pour être évolutive, de manière à pouvoir stocker un grand nombre de biens immobiliers et à gérer une charge importante d'accès.

**Système de cache :** Pour améliorer les performances et réduire les temps de réponse, le service de recherche de biens immobiliers pourrait utiliser un système de cache pour stocker les résultats de recherche les plus fréquemment consultés. Cela permettrait aux utilisateurs de récupérer rapidement les informations sur les biens immobiliers et de réduire la charge sur le système de recherche.

**API :** Le service de recherche de biens immobiliers serait conçu pour offrir une API qui permettrait aux autres services de l'application d'accéder aux informations sur les biens

immobiliers disponibles. Cette API serait documentée et facile à utiliser, ce qui permettrait une intégration facile avec les autres services de l'application.

## Service de gestion des propriétés

Le service de gestion des propriétés serait responsable de permettre à l'agence immobilière de gérer les propriétés qu'elle a en gestion. Pour ce faire, il utiliserait un système de base de données qui stockerait les informations sur les propriétés, telles que les contrats de location, les informations sur les locataires et les détails sur les propriétés.

Le service de gestion des propriétés serait conçu pour être évolutif et résilient, ce qui signifie qu'il serait capable de gérer une charge élevée d'utilisateurs et de continuer à fonctionner même en cas de panne d'un des composants du système.

Voici quelques éléments clés que le service de gestion des propriétés pourrait inclure :

**Interface utilisateur :** Le service de gestion des propriétés offrirait une interface utilisateur permettant aux agents immobiliers de l'agence de gérer les propriétés qu'ils ont en gestion. Cette interface utilisateur serait facile à utiliser et permettrait aux agents immobiliers de mettre à jour les informations sur les propriétés, de gérer les contrats de location et de communiquer avec les locataires.

**Système de gestion des contrats :** Le service de gestion des propriétés serait équipé d'un système de gestion des contrats de location, qui permettrait aux agents immobiliers de gérer les contrats de location en cours et de créer de nouveaux contrats de location. Ce système de gestion des contrats serait conçu pour être facile à utiliser et à mettre à jour.

**Système de gestion des locataires :** Le service de gestion des propriétés serait équipé d'un système de gestion des locataires, qui permettrait aux agents immobiliers de gérer les informations sur les locataires, telles que les coordonnées, les paiements et les demandes de réparation. Ce système de gestion des locataires serait conçu pour être facile à utiliser et à mettre à jour.

**Base de données :** Le service de gestion des propriétés serait connecté à une base de données qui stockerait les informations sur les propriétés, les contrats de location et les informations sur les locataires. Cette base de données serait conçue pour être évolutive, de manière à pouvoir stocker un grand nombre de propriétés et de contrats de location et à gérer une charge importante d'accès.

**API :** Le service de gestion des propriétés serait conçu pour offrir une API qui permettrait aux autres services de l'application d'accéder aux informations sur les propriétés, les contrats de location et les informations sur les locataires. Cette API serait documentée et facile à utiliser, ce qui permettrait une intégration facile avec les autres services de l'application.



## Service de réservations de maisons

Le système de réservation de maisons serait responsable de permettre aux utilisateurs de l'application de réserver des visites pour les maisons qui les intéressent. Pour ce faire, il utiliserait un système de base de données qui stockerait les informations sur les maisons disponibles, ainsi que les disponibilités des visites.

Le système de réservation de maisons serait conçu pour être évolutif et résilient, ce qui signifie qu'il serait capable de gérer une charge élevée d'utilisateurs et de continuer à fonctionner même en cas de panne d'un des composants du système.

Voici quelques éléments clés que le système de réservation de maisons pourrait inclure :

**Interface utilisateur :** Le système de réservation de maisons offrirait une interface utilisateur permettant aux utilisateurs de réserver des visites pour les maisons qui les intéressent. Cette interface utilisateur serait facile à utiliser et permettrait aux utilisateurs de filtrer les résultats de recherche en fonction de leurs besoins.

**Système de réservation :** Le système de réservation de maisons serait équipé d'un système de réservation performant, qui serait capable de réserver rapidement et efficacement des visites pour les maisons en fonction des disponibilités. Ce système de réservation serait conçu pour être évolutif et capable de gérer une charge importante.

**Base de données :** Le système de réservation de maisons serait connecté à une base de données qui stockerait les informations sur les maisons disponibles à la visite, ainsi que les disponibilités des visites. Cette base de données serait conçue pour être évolutive, de manière à pouvoir stocker un grand nombre de maisons et de disponibilités de visite et à gérer une charge importante d'accès.

**Système de notification :** Le système de réservation de maisons pourrait inclure un système de notification pour informer les utilisateurs des confirmations de réservation, des changements de disponibilité et des annulations de visite. Ce système de notification serait conçu pour être rapide et fiable, afin que les utilisateurs soient informés en temps réel des informations importantes.

**API :** Le système de réservation de maisons serait conçu pour offrir une API qui permettrait aux autres services de l'application d'accéder aux informations sur les maisons disponibles et les disponibilités de visite. Cette API serait documentée et facile à utiliser, ce qui permettrait une intégration facile avec les autres services de l'application.

# Les technologies utilisées

## ReactJS pour le front-end

ReactJS est une bibliothèque JavaScript open source qui permet de construire des interfaces utilisateur interactives et réactives pour le web. Elle est souvent utilisée pour le développement de front-end dans les applications web modernes.

L'un des avantages de ReactJS est son approche basée sur les composants. Les développeurs peuvent créer des composants réutilisables pour représenter des parties spécifiques de l'interface utilisateur, et les combiner pour construire des interfaces plus complexes. ReactJS utilise un DOM virtuel pour améliorer les performances et réduire les temps de chargement de la page.

Axios, quant à lui, est une bibliothèque JavaScript populaire qui permet de faire des requêtes HTTP depuis un navigateur ou depuis Node.js. Elle est souvent utilisée pour récupérer des données à partir de serveurs distants, pour communiquer avec des APIs ou pour effectuer des opérations CRUD (créer, lire, mettre à jour et supprimer) sur des bases de données.

Axios simplifie la gestion des requêtes HTTP en offrant une syntaxe simple et intuitive pour créer des requêtes, gérer les réponses et les erreurs. Elle permet également de configurer des intercepteurs pour ajouter des en-têtes, des jetons d'authentification ou des fonctions de traitement de la réponse.

En utilisant ReactJS avec Axios, les développeurs peuvent créer des interfaces utilisateur interactives qui communiquent avec des serveurs distants de manière efficace et sécurisée. Les composants React peuvent inclure des appels Axios pour récupérer des données en temps réel, ou pour envoyer des données à un serveur pour effectuer des opérations de mise à jour ou de suppression. Dans notre application, nous avons donc utilisés ces deux technologies pour développer notre partie front-end.

## Pour le back-end

L'utilisation de microservices est intéressante car elle permet de développer une application en utilisant plusieurs technologies différentes. Chaque microservice peut être développé dans un langage de programmation différent, en fonction de ses besoins spécifiques.

Cette approche offre une grande flexibilité et permet de choisir les meilleures technologies pour chaque microservice, sans être limité par les choix faits pour le reste de l'application. Par exemple, un microservice de traitement de données peut être développé en utilisant Python, tandis qu'un microservice de gestion d'utilisateurs peut être développé en utilisant Java.

Dans notre cas, deux de nos services utilisent du nodeJS pendant qu'un autre utilise du spring.

# Gestion de déploiement du projet

## Utilisation de docker

Docker est une plateforme de virtualisation légère qui permet d'isoler les applications dans des conteneurs, ce qui permet de les exécuter de manière fiable et cohérente, indépendamment de l'environnement d'exécution.

Dans un environnement de développement traditionnel, il est souvent difficile de garantir que l'application fonctionnera correctement sur différents systèmes d'exploitation, versions de langages de programmation, bibliothèques tierces, etc. Cela peut conduire à des problèmes de compatibilité, de performances, de sécurité et de maintenance.

Docker résout ce problème en fournissant une solution de virtualisation légère qui encapsule l'application et toutes ses dépendances dans un conteneur. Un conteneur est une unité logique qui contient tout ce dont l'application a besoin pour s'exécuter, y compris le code, les bibliothèques, les outils système, les fichiers de configuration, etc. Les conteneurs peuvent être exécutés sur n'importe quel système d'exploitation ou infrastructure de cloud qui prend en charge Docker, ce qui permet de garantir la portabilité et la fiabilité de l'application.

En plus de l'isolation des applications, Docker offre également des avantages en termes de performances, de sécurité et de gestion des ressources. Les conteneurs Docker sont légers et nécessitent peu de ressources système, ce qui permet d'exécuter plusieurs applications sur une même machine sans impact significatif sur les performances. Les conteneurs sont également isolés les uns des autres, ce qui renforce la sécurité et permet de limiter les risques de compromission.

Enfin, Docker simplifie également la gestion des applications, en permettant de créer, de déployer et de mettre à jour les conteneurs de manière automatisée. Les images Docker, qui sont des modèles pour la création de conteneurs, peuvent être versionnées et partagées entre les développeurs, ce qui facilite la collaboration et le déploiement continu.

## Déploiement de notre application

Pour déployer le projet, vous devez tout d'abord récupérer le code source à partir du dépôt Git situé à l'adresse [https://github.com/B-Mahdj/Project\\_MicroServices.git](https://github.com/B-Mahdj/Project_MicroServices.git). Vous pouvez le faire en copiant le projet à partir du terminal en utilisant la commande "git clone [https://github.com/B-Mahdj/Project\\_MicroServices.git](https://github.com/B-Mahdj/Project_MicroServices.git)".

Une fois le projet récupéré, vous devez vous rendre dans le répertoire principal de l'application, qui se trouve dans le dossier nommé "Project\_MicroServices". Vous pouvez utiliser la commande "cd Project\_MicroServices" pour accéder à ce répertoire.

Ensuite, pour lancer l'application, vous devez exécuter la commande "**docker compose up**" dans le terminal. Cette commande permettra de créer les images Docker nécessaires et de lancer les containers pour exécuter l'application. Veuillez noter que le processus de création des images Docker peut prendre un certain temps.

Une fois que toutes les images sont créées et que les containers sont lancés, vous pouvez accéder à l'application en ouvrant votre navigateur web et en saisissant l'adresse "<http://localhost:3003/>". Cette adresse correspond à l'emplacement de l'application sur votre machine locale. À partir de là, vous pourrez interagir avec l'application et utiliser ses fonctionnalités.

## Démonstration

Page d'accueil de notre application :



## Gestion de l'authentification :

Bienvenue, veuillez-vous connecter !


Se connecter

Adresse e-mail :

Mot de passe :


Connexion


## Visualisation des biens présents :

 Les agents tous risques


Propriété

Ajouter un bien






Maison  
500000€  
Rue des Lilas, Paris




Appartement  
300000€  
Avenue des Roses, Lyon



Villa  
800000€  
Avenue des Pins, Nice

## Détails d'un bien immobilier :


 Les agents tous risques

Propriété

Ajouter un bien

Maison  
500000€  
Rue des Lilas, Paris

## Insertion d'une nouvelle propriété :

 Les agents tous risques

Propriété

Ajouter un bien

Type de bien :

Adresse :


Ville :

Prix de vente :


Choisir une image du bien immobilier




Enregistrer le bien




**Maison**  
**5000000€**  
Rue des Lilas, Paris



**Appartement**  
**3000000€**  
Avenue des Roses, Lyon




**Villa**  
**8000000€**  
Avenue des Pins, Nice



**maison**  
**12340000€**  
12 rue du ventoux, Dijon

## Profil utilisateur :



**Informations utilisateur**  
Nom : Dupont  
Prénom : Jean  
Adresse : Rue des Lilas  
Ville : Paris  
Email : 0123456789  
Téléphone : j.dupont@email.com  
Statut : Propriétaire

Déconnexion