

CASTANO ORTIZ Diana Carolina

31/07/2023

CASTANO ORTIZ Diana Carolina

# Dossier Projet

CDA 2022 - 2023 / MNS / IFA

<b>I. Liste des compétences du référentiel qui sont couvertes par le projet</b>	<b>2</b>
<b>II. Résumé du projet en anglais</b>	<b>3</b>
<b>III. Cahier des charges ou expression des besoins du projet</b>	<b>4</b>
Présentation de l'entreprise	4
Présentation du projet	4
Fonctionnalités	5
Possibles restrictions du projet d'application web immobilière	7
<b>IV. Gestion de projet (planning et suivi, environnement humain et technique, objectifs de qualité)</b>	<b>8</b>
Outils utilisés	8
Certaines des outils que j'ai utilisés sont :	8
Changements de vision du client pendant le projet et adaptations	9
<b>V. Spécifications fonctionnelles du projet</b>	<b>10</b>
Cas d'utilisation	10
Diagramme de classes	14
Dictionnaire de données	14
Méthodologie de modélisation MERISE.	17
Niveau Conceptuel MCD	18
Niveau Logique MLD	19
Modèle UML	19
Maquette ou conception UX	20
<b>VI. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité</b>	<b>22</b>
Choix des technologies	22
Choix de la base de données	22
Solution Back-end	22
Solution Front-End	23
Comportement intégral de Spring Boot et Angular	24
<b>VII. Réalisations du candidat comportant les extraits de code les plus significatifs et en les argumentant, y compris pour la sécurité</b>	<b>26</b>
a. Back-end	26
Création d'une entité dans Spring Boot	26
Création d'un DAO (Data Access Object)	28
Création d'un contrôleur dans Spring Boot	29
Description algorithme de connexion pour les utilisateurs et les administrateurs	31
Sécurité: Spring Boot Security et des tokens JWT	32
b. Front-end	33
Création d'une modal sous Angular	33
Création d'un menu de navigation sous Angular	38
Sécurisé: Authentification par JWT	41
Inscription d'un utilisateur	44
<b>VIII. Présentation des jeux d'essai</b>	<b>47</b>
Tests sur l'API Spring Boot	47

Tests sur la partie Front-end	49
<b>IX. Liste des Figures</b>	<b>52</b>

## **I. Liste des compétences du référentiel qui sont couvertes par le projet**

1. Maquetter une application
2. Développer une interface utilisateur
3. Développer des composants d'accès aux données
4. Développer la partie front-end d'une interface utilisateur
5. Développer la partie back-end d'une interface utilisateur
6. Concevoir une base de données
7. Mettre en place une base de données
8. Développer des composants dans le langage d'une base de données
9. Concevoir une application
10. Développer des composants métiers
11. Construire une application organisée en couches
12. Développer une application mobile
13. Préparer et exécuter les plans de tests d'une application
14. Préparer et exécuter le déploiement d'une application

## II. Résumé du projet en anglais

SCI Cansell Real Estate Web Application offers features such as personalized search queries for buying, selling, or renting properties, as well as the ability for users to upload photos of their properties. For this application, I have implemented a form for accurate property valuation. Additionally, there is a dedicated space to showcase renovations done on previous properties.

I have created a mockup with five pages: Home, Search, Renovation, Estimation, and Image Upload. Each page serves a specific function.

For database modeling, I have used the MERISE approach, creating six fundamental tables, including roles, property photos, registered users, property valuations, categories, and property owners' data.

The use case diagram illustrates the interactions between application actors such as users, real estate agents, and administrators. Each actor plays an essential role in the functioning and interaction with the database.

Regarding the application development, I have utilized the Spring Boot framework for the backend, leveraging its features like dependency injection, transaction management, and integrated security. For the frontend, I have employed Angular, a TypeScript-based framework that allows for robust architecture and dynamic user interface creation.

I have implemented Spring Security dependencies and configured URL access, restricting certain functionalities based on user roles. In Angular, I have used routes and guards to control component access and implemented CRUD functionality to keep user and property data up-to-date.

### **III. Cahier des charges ou expression des besoins du projet**

#### **Présentation de l'entreprise**

SCI Cansell est une entreprise familiale leader dans le secteur immobilier avec une remarquable expérience de plus de 50 ans, au cours desquels elle s'est distinguée en offrant des services de qualité et de confiance. Son principal engagement réside dans la satisfaction des besoins de ses clients en leur fournissant des solutions efficaces et adaptées à leurs exigences sur le marché immobilier dynamique.

L'entreprise s'est adaptée aux changements technologiques et cherche constamment à favoriser la rencontre entre propriétaires et acheteurs via des canaux numériques. Ils reconnaissent qu'à l'ère numérique, il est essentiel d'offrir aux clients le confort et l'accessibilité que procurent les plateformes en ligne. C'est pourquoi ils ont souhaité développer une application Web immobilière permettant aux utilisateurs d'explorer et de sélectionner des propriétés de manière pratique et efficace.

Ils ne se limitent pas seulement à présenter des propriétés aux clients, mais ils jouent également un rôle de conseil complet. Ils comprennent que le processus d'achat ou de vente d'une propriété implique des aspects légaux, financiers, fiscaux et contractuels qui peuvent être complexes et déroutants pour les parties impliquées. Ils s'engagent donc à fournir à leurs clients le soutien nécessaire dans ces domaines, en collaborant étroitement avec des professionnels du secteur tels que des notaires, des institutions financières et des experts en fiscalité et en contrats. De cette manière, ils garantissent que les clients disposent de toutes les informations et du soutien nécessaires pour prendre des décisions éclairées et sûres dans leurs transactions immobilières.

En ce qui concerne les clients, SCI Cansell se consacre à écouter attentivement leurs besoins et leurs exigences. Ils reconnaissent que chaque client a des exigences et des préférences uniques lorsqu'il recherche une propriété. C'est pourquoi ils s'efforcent de comprendre en profondeur les attentes de leurs clients et de leur présenter des propriétés qui correspondent précisément à leur profil de recherche. De plus, pour offrir encore plus de confiance et de sécurité, ils effectuent des évaluations et des inspections légales avant l'achat. Cela permet aux clients d'avoir une vision claire et complète des propriétés qu'ils envisagent, en s'assurant que leur investissement est judicieux et étayé par des informations fiables.

#### **Présentation du projet**

L'idée principale est de développer une application web immobilière offrant aux utilisateurs une expérience interactive et participative concernant leurs biens immobiliers. L'objectif est que les utilisateurs se sentent à l'aise et puissent avoir un meilleur contrôle sur l'information et la présentation visuelle de leurs propriétés.

Avec cette application, les utilisateurs pourront interagir plus activement avec les données de leurs biens immobiliers. Ils auront non seulement accès à des informations détaillées telles que les caractéristiques, l'emplacement et la description, mais pourront également contribuer avec leurs propres images des espaces qu'ils souhaitent montrer. Cependant, afin de garantir la qualité et la cohérence des images, celles-ci devront toutes passer par un processus de validation de la part de l'agence immobilière. Cela assurera que les photographies présentées soient attrayantes et précises, offrant une représentation fidèle de chaque propriété.

De plus, un service d'évaluation en ligne des biens immobiliers a été intégré. Ce service permettra aux utilisateurs d'obtenir une estimation du prix de leurs propriétés sans avoir besoin d'interagir directement avec un agent immobilier. Grâce à un agent immobilier spécialisé, les utilisateurs pourront fournir des informations pertinentes sur leurs biens et recevoir une évaluation précise et fiable. Cette fonctionnalité leur offrira une perspective claire de la valeur de leurs propriétés et les aidera à prendre des décisions éclairées concernant la vente, la location ou l'achat de biens immobiliers.

Il est important de souligner que pour accéder tant au service d'évaluation qu'à l'option de téléchargement d'images, les utilisateurs devront être enregistrés sur l'application web. L'inscription leur permettra de profiter pleinement de ces avantages et leur offrira une expérience personnalisée. De plus, dans un souci de sécurité et de protection des données, des mesures de sécurité ont été mises en place pour garantir que l'accès et l'utilisation de la plateforme soient sécurisés et confidentiels.

Cette application web immobilière vise à offrir aux utilisateurs une plateforme interactive et participative, où ils pourront interagir avec les informations de leurs biens immobiliers et jouer un rôle actif dans leur présentation visuelle. Avec des services d'évaluation en ligne et la possibilité de télécharger des images validées, l'idée est de fournir aux utilisateurs des outils qui les aideront à prendre des décisions intelligentes et bien fondées sur le marché immobilier. Tout cela, dans un environnement sûr et fiable qui favorise la confiance et la satisfaction des clients/utilisateurs.

## **Fonctionnalités**

L'application web de l'agence immobilière a pour objectif principal de faciliter aux utilisateurs la recherche, la publication et l'évaluation de biens immobiliers de manière efficace et pratique.

Voici les principales fonctionnalités de l'application :

**Recherche de biens immobiliers :** Les utilisateurs peuvent effectuer des recherches en fonction de leurs besoins spécifiques, que ce soit pour l'achat, la vente ou la location de

propriétés. L'application propose des options de filtrage par catégorie (type de propriété, emplacement, prix, etc.) ainsi que par ID de propriété, ce qui permet aux utilisateurs de trouver facilement les biens qui correspondent à leurs préférences.

**Publication d'images de biens immobiliers :** Les utilisateurs ont la possibilité de publier leurs propres photos de manière simple et efficace. Ils peuvent télécharger des photographies de la propriété en suivant un processus rigoureux d'approbation de la part de l'agence immobilière. Cela permet aux utilisateurs de participer activement à la présentation visuelle de leur propriété, contribuant ainsi à la présenter de manière attrayante et précise.

**Évaluation de biens immobiliers :** L'application propose un service d'évaluation en ligne qui permet aux utilisateurs de connaître la valeur estimée de leurs biens immobiliers. Les utilisateurs peuvent saisir les informations pertinentes sur la propriété et recevoir une analyse détaillée et précise de son évaluation. Cette fonctionnalité offre aux utilisateurs un outil utile pour prendre des décisions éclairées concernant l'achat, la vente ou la location de biens immobiliers.

**Rénovations de propriétés :** L'application dispose d'un espace dédié pour présenter les rénovations réalisées sur différentes propriétés. Cela permet aux utilisateurs d'avoir une vision claire du potentiel d'amélioration et de la qualité des propositions immobilières. Cette fonctionnalité offre aux utilisateurs la possibilité d'explorer les transformations réalisées sur des biens immobiliers précédents, ce qui peut influencer leur décision d'achat ou de location.

**Gestion des utilisateurs :** L'application permet aux utilisateurs de s'inscrire et d'accéder à leur compte personnalisé. À travers leur compte, les utilisateurs peuvent effectuer des actions telles que télécharger des photos de biens immobiliers, demander des évaluations, mettre à jour des informations personnelles et gérer leurs propriétés dans l'application (il convient de préciser que les informations sur la propriété ne peuvent être mises à jour que par l'agence immobilière).

**Rôles des utilisateurs :** L'application différencie les rôles des utilisateurs, tels que l'administrateur de l'application. Chaque rôle a des privilèges et des fonctionnalités différentes dans l'application. Par exemple, les administrateurs peuvent publier des propriétés, les utilisateurs ne peuvent que rechercher et contacter un agent, tandis que l'utilisateur enregistré peut également télécharger des images d'une propriété qui lui appartient et/ou faire une évaluation.

**Interaction avec les agents immobiliers :** Les utilisateurs peuvent interagir avec les agents immobiliers à travers l'application pour obtenir des conseils personnalisés et des réponses à leurs questions. Cette fonctionnalité offre un canal de communication direct entre les utilisateurs et les experts immobiliers, améliorant ainsi l'expérience de l'utilisateur et offrant L'application web de l'agence immobilière offre aux utilisateurs une plateforme complète et conviviale pour rechercher, publier et évaluer. Elle permet aux utilisateurs de participer



activement à la présentation visuelle de leurs propriétés et d'obtenir des évaluations précises. Avec des fonctionnalités supplémentaires telles que la visualisation des rénovations et l'interaction avec les agents immobiliers, l'application offre une expérience globale qui facilite la prise de décisions sur le marché immobilier."

### **Possibles restrictions du projet d'application web immobilière**

**Contrainte de temps :** En raison d'une date limite, il est possible que le développement web ne puisse pas être entièrement achevé avant cette date.

**Contrainte des ressources humaines :** Les ressources humaines sont d'une importance capitale, car je ne dispose pas de la collaboration d'autres personnes pouvant m'aider en cas de doutes techniques ou ayant l'expérience des outils que j'utilise pour mener à bien le développement de l'application. Dans ce projet, je travaille de manière individuelle.

**Contraintes de portée :** Le projet a une portée définie avec des caractéristiques et des fonctionnalités spécifiques qui doivent être mises en œuvre dans l'application web.

**Contraintes de sécurité :** Étant donné que l'application stocke des informations sur les utilisateurs et les propriétés, il est impératif de respecter rigoureusement les normes de sécurité et de confidentialité. Des mesures doivent être prises pour protéger les informations confidentielles et se conformer aux réglementations en vigueur.

Ces restrictions sont des considérations importantes qui doivent être prises en compte lors du développement de l'application web immobilière.

#### **IV. Gestion de projet (planning et suivi, environnement humain et technique, objectifs de qualité)**

Étant donné que je travaille seule, j'applique les principes d'organisation et de suivi pour maximiser l'efficacité et m'assurer d'atteindre les objectifs fixés.

Pour commencer, j'ai établi les délais pour chaque tâche à réaliser, tels que la création du modèle de base de données, la conception et le développement. Fondamentalement, je les ai divisées en trois étapes ou tâches principales, et certaines de ces tâches ont été subdivisées en sous-tâches.

Dans cette optique, j'ai défini les principales fonctionnalités à mettre en œuvre, telles que la connexion et l'inscription de l'utilisateur, ainsi que la sécurité des données pour compléter l'interface utilisateur.

Certaines sous-tâches plus petites incluent le développement de la page d'accueil avec les informations de l'entreprise, la mise en place de la page de recherche et l'intégration des données pour charger toutes les informations de la page.

J'ai fixé des dates pour chaque tâche, comme la conception UX, le développement des pages, la connexion, les enregistrements, les validations, etc. Tout cela est planifié avec une date de début et une date de fin indiquées dans l'outil Trello, où chaque tâche a un temps spécifique pour être réalisée. Bien sûr, tout est sujet à des changements en cas d'imprévus. J'ai également défini mes propres horaires de travail, en travaillant en continu du lundi au vendredi.

##### **Outils utilisés**

**Certaines des outils que j'ai utilisés sont :**

**Trello :** J'utilise Trello pour suivre les tâches que je dois effectuer tout au long des semaines. J'attribue un temps estimé à chaque tâche pour planifier mon travail de manière efficace.

**GitHub :** J'utilise GitHub pour contrôler les versions de mes projets. Cela me permet de suivre les changements effectués pendant le développement du programme et de maintenir un historique actualisé des modifications.

**IntelliJ IDEA :** Travailler dans l'environnement IntelliJ IDEA a amélioré ma productivité dans le développement. Cet outil offre des fonctionnalités qui facilitent la création et la maintenance du projet, ce qui m'a permis d'optimiser mon flux de travail.

## Changements de vision du client pendant le projet et adaptations

Tout au long de tout projet, il est compréhensible que le client, dans ce cas, l'agence immobilière, puisse demander des changements à un moment donné. Ces changements doivent être pris en compte dans le temps estimé pour le développement, le cas échéant. Pour cette raison, des temps supplémentaires ont été alloués à certaines tâches afin de pouvoir répondre de manière appropriée aux besoins du client.

Si le client souhaite ajouter de nouvelles fonctionnalités, il sera nécessaire de mettre en place de nouveaux changements dans le plan et les exigences pour inclure ces fonctionnalités. Il est important de souligner que ces changements sont effectués en tenant compte des temps de travail déjà établis.

La flexibilité pour s'adapter aux changements est essentielle dans le processus de développement du projet. Je dois être prêt(e) à ajuster le plan et les tâches en fonction des exigences du client. Cela implique d'évaluer l'impact des changements proposés, tant en termes de temps que de ressources, et de s'assurer que le projet puisse continuer à avancer de manière efficace et en ligne avec les attentes du client.

Ci-dessous, je vous présente une image de ces tâches

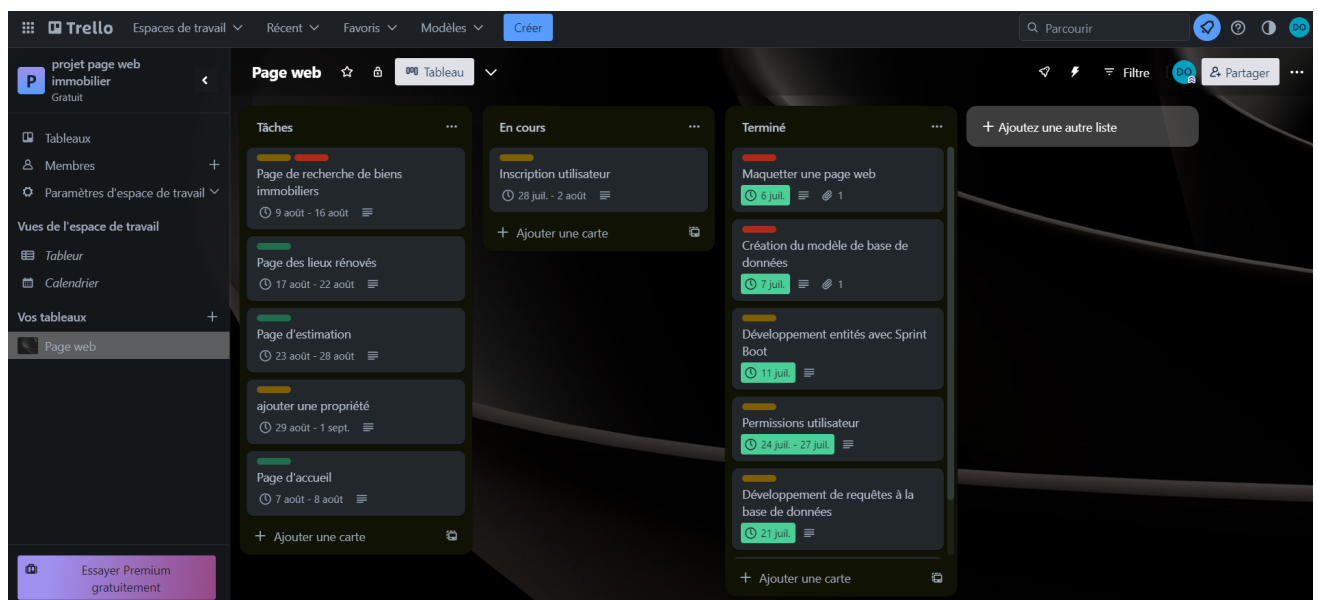


Figure #1 -> Gestion de projet

## V. Spécifications fonctionnelles du projet

Voici les différents cas d'utilisation qui seront mis en œuvre dans l'application web :

### ***Cas d'utilisation : Recherche de Biens Immobiliers***

L'utilisateur peut effectuer des recherches de biens immobiliers selon ses intérêts, que ce soit pour louer, acheter ou vendre. Il pourra utiliser des critères de recherche tels que la catégorie, l'ID du bien immobilier ou d'autres filtres pertinents.

Acteurs	Utilisateur	
Préconditions	accéder à l'application	
Scenarios	Étapes	Règles de gestion
	* Aller à la page de recherche. * Rechercher par son ID / catégorie. * Contacter un agent immobilier.	
Post conditions	Attendre réponse de l'agence immobilière	

### Cas d'utilisation

### ***Publication de Photos de Biens Immobiliers***

Les clients ont la possibilité de publier facilement et efficacement les photos de leurs biens immobiliers. L'application suivra un processus d'approbation rigoureux pour garantir une présentation visuelle attrayante et précise de chaque bien immobilier.

Acteurs	Utilisateur/Administrateur	
Préconditions	accéder à l'application s'inscrire	
Scenarios	Étapes	Règles de gestion
	* Aller à la page de recherche. * Sélectionner la propriété ou la rechercher par son ID. * Cliquer sur le bouton "Télécharger des photos". * Envoyer.	Vous devez être inscrit en tant qu'utilisateur
Post conditions	Attendre la validation des images par l'agence immobilière	

### **Visualisation des Rénovations de Propriétés**

L'application offrira un espace dédié pour montrer les rénovations réalisées dans différentes propriétés. Les utilisateurs pourront avoir une vision claire du potentiel d'amélioration et de la qualité de la proposition immobilière.

Acteurs	Utilisateur	
Préconditions	accéder à l'application	
Scenarios	Étapes	Règles de gestion
	* Aller à la page de rénovation de propriété * voir les photos. * Communiquer avec un conseiller immobilier	Vous devez être inscrit en tant qu'utilisateur
Post conditions		

### **Inscription des Utilisateurs**

Les utilisateurs auront la possibilité de s'inscrire sur le système en fournissant des informations telles que leur nom, leur adresse, leur adresse e-mail, entre autres. Cette inscription sera nécessaire pour effectuer certaines actions spécifiques sur l'application.

Acteurs	Utilisateur	
Préconditions	accéder à l'application, s'identifier	
Scenarios	Étapes	Règles de gestion
	* Accéder à la page d'inscription * Remplir le formulaire	Vous devez être inscrit en tant qu'utilisateur
Post conditions	Confirmer l'inscription	

### **Validation des Biens Immobiliers**

Les utilisateurs pourront valider les informations de leurs biens immobiliers afin qu'ils apparaissent correctement sur la plateforme. Cette validation impliquera la confirmation de données et de détails pertinents.

Acteurs	Utilisateur	
Préconditions	accéder à l'application s'inscrire	
Scenarios	Étapes	Règles de gestion
	* Aller sur la page de validation * Remplir les champs avec les informations requises * Envoyer	Vous devez être inscrit en tant qu'utilisateur
Post conditions	Attendre la réponse de l'agence immobilière par e-mail	

### ***Gestion des Rôles et des Permissions***

L'administrateur de l'application pourra gérer les rôles et les permissions des utilisateurs. Il attribuera différents niveaux d'accès et d'autorisation pour assurer la sécurité et le bon fonctionnement de la plateforme.

Acteurs	Administrateur	
Préconditions	accéder à l'application Connexion	
Scenarios	Étapes	Règles de gestion
	* Accéder a la page de connexion * Gérer les rôles et les permissions des utilisateurs	
Post conditions	Confirmer	

### ***Gestion des Biens Immobiliers et des Utilisateurs***

L'administrateur sera responsable de la gestion des informations des biens immobiliers et des utilisateurs. Il pourra insérer, supprimer ou mettre à jour des enregistrements pour maintenir la base de données à jour.

Acteurs	Administrateur	
Préconditions	accéder à l'application s'identifier	
Scenarios	Étapes	Règles de gestion
	<ul style="list-style-type: none"> <li>* Aller à la page de recherche.</li> <li>* Rechercher par son ID / catégorie</li> <li>* Sélectionner l'option "Insérer, supprimer, chercher".</li> <li>* Remplir les champs requis.</li> <li>* Insérer les informations.</li> </ul>	Avoir les permissions d'administrateur.
Post conditions	Notification d'insertion	

### ***Téléchargement de Photographies***

Les utilisateurs auront la capacité de télécharger leurs propres photographies liées aux biens immobiliers. Les images seront évaluées et approuvées par l'agent immobilier pour assurer la qualité et l'exactitude des informations.

Acteurs	Utilisateur	
Préconditions	accéder à l'application s'inscrire/connexion	
Scenarios	Étapes	Règles de gestion
	<ul style="list-style-type: none"> <li>* Accéder à la page de recherche</li> <li>* sélectionner la propriété immobilière</li> <li>* Télécharge les photos</li> </ul>	
Post conditions	Envoyer, Attendre la validation d'image pour l'agence immobilière	

Ces cas d'utilisation ont été conçus pour garantir une expérience optimale aux utilisateurs de l'application web immobilière SCI Cansell, en offrant des fonctionnalités essentielles et en facilitant l'interaction avec la plateforme.

## Diagramme de classes

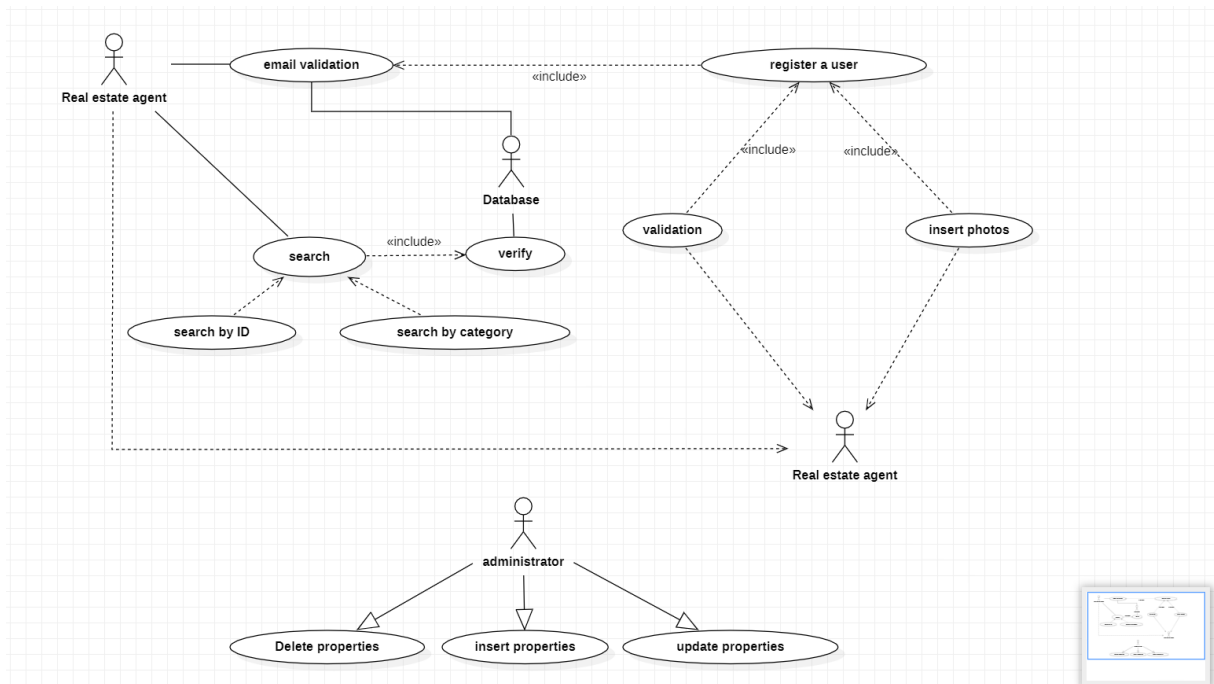


Figure #2 -> Diagramme de classe

## Dictionnaire de données

### Role

**role\_id:** Identifiant unique du rôle. (Type de données : chaîne de caractères de longueur 50)

**name\_role:** Nom du rôle. (Type de données : chaîne de caractères de longueur 50)

**PRIMARY KEY(role\_id):** Clé primaire pour la table Role.

### PropertyOwnerInformation

**owner\_id:** Identifiant unique du propriétaire. (Type de données : entier)

**first\_name:** Prénom du propriétaire. (Type de données : chaîne de caractères de longueur 50)

**last\_name:** Nom de famille du propriétaire. (Type de données : chaîne de caractères de longueur 50)

**email:** Adresse e-mail du propriétaire. (Type de données : chaîne de caractères de longueur 100)

**telephone:** Numéro de téléphone du propriétaire. (Type de données : chaîne de caractères de longueur 50)

**address:** Adresse du propriétaire. (Type de données : chaîne de caractères de longueur 50)

**registered\_owner:** Indicateur du propriétaire enregistré (Type de données : LOGIQUE)

**PRIMARY KEY(owner\_id) :** Clé primaire pour la table PropertyOwnerInformation.



### ***CategoryContract***

**category\_id:** Identifiant unique de la catégorie. (Type de données : entier)

**stateContract:** Indique si un contrat a été établi avec le client. (Type de données : booléen)

**PRIMARY KEY(category\_id):** Clé primaire pour la table Category.

### ***TypePhoto***

**type\_photo\_id:** Identifiant unique du type de photo. (Type de données : entier)

**type\_photo:** Indique s'il s'agit d'une photo de propriété ou d'une photo de propriété rénover (Type de données : VARCHAR(50)).

**PRIMARY KEY(type\_photo\_id):** Clé primaire pour la table TypePhoto.

### ***ValidationState***

**validation\_state\_id:** Identifiant unique de l'état de la propriété. (Type de données : entier)

**descriptionState:** Indique l'état de la propriété (en validation, approuvé) (Type de données : VARCHAR(100))

### ***Property***

**property\_id:** Identifiant unique de la propriété. (Type de données : entier)

**real\_estate\_type:** Type de bien immobilier. (Type de données : chaîne de caractères de longueur 50)

**real\_estate\_age:** Âge du bien immobilier. (Type de données : octet)

**number\_showers:** Nombre de douches. (Type de données : entier)

**number\_bedrooms:** Nombre de chambres. (Type de données : entier)

**number\_rooms:** Nombre total de chambres. (Type de données : entier)

**needs\_renovation:** Indicateur de la nécessité de rénovation (vrai/faux). (Type de données : booléen)

**land\_surface:** Superficie du terrain. (Type de données : décimal(15,2))

**living\_room\_surface:** Superficie du salon. (Type de données : décimal(15,2))

**house\_surface:** Superficie de la maison. (Type de données : décimal(15,2))

**parking:** Indicateur de présence d'un parking (vrai/faux). (Type de données : booléen)

**cellar:** Indicateur de présence d'une cave (vrai/faux). (Type de données : booléen)

**cellar\_surface:** Superficie de la cave. (Type de données : décimal(15,2))

**balcony:** Indicateur de présence d'un balcon (vrai/faux). (Type de données : booléen)

**balcony\_surface:** Superficie du balcon. (Type de données : décimal(15,2))

**pool:** Indicateur de présence d'une piscine (vrai/faux). (Type de données : booléen)

**pool\_exposure:** Exposition de la piscine. (Type de données : chaîne de caractères de longueur 50)

**house\_view:** Vue de la maison. (Type de données : chaîne de caractères de longueur 50)

**house\_quality:** Qualité de la maison. (Type de données : chaîne de caractères de longueur 50)

**semi\_detached:** Indicateur de maison mitoyenne (vrai/faux). (Type de données : booléen)

**environment:** Environnement de la propriété. (Type de données : chaîne de caractères de longueur 50)

**validation\_state:** État de validation. (Type de données : décimal(15,2))

**description:** Description de la propriété. (Type de données : chaîne de caractères de longueur 500)

**price:** Prix de la propriété. (Type de données : décimal(15,2))

**category\_id:** ID de catégorie liée (FK). (Type de données : entier, NON NULL)

**owner\_id:** ID du propriétaire lié (FK). (Type de données : entier, NON NULL)

**PRIMARY KEY(property\_id):** Clé primaire pour la table Property.

**FOREIGN KEY(contract\_id):** Clé étrangère faisant référence à la colonne contract\_id de la table CategoryContract.

**FOREIGN KEY(validation\_state\_id):** Clé étrangère faisant référence à la colonne validation\_state\_id de la table validationState.

### ***Users***

**user\_id:** Identifiant unique de l'utilisateur. (Type de données : entier)

**first\_name:** Prénom de l'utilisateur. (Type de données : chaîne de caractères de longueur 100)

**last\_name:** Nom de famille de l'utilisateur. (Type de données : chaîne de caractères de longueur 100)

**telephone:** Numéro de téléphone de l'utilisateur. (Type de données : chaîne de caractères de longueur 50)

**address:** Adresse de l'utilisateur. (Type de données : chaîne de caractères de longueur 250)

**email:** Adresse e-mail de l'utilisateur. (Type de données : chaîne de caractères de longueur 100)

**password:** Mot de passe de l'utilisateur. (Type de données : chaîne de caractères de longueur 100)

**property\_id:** ID de propriété liée (FK). (Type de données : entier, NON NULL)

**role\_id:** ID de rôle lié (FK). (Type de données : chaîne de caractères de longueur 50, NON NULL)

**PRIMARY KEY(user\_id):** Clé primaire pour la table Users.

**FOREIGN KEY(property\_id):** Clé étrangère faisant référence à la colonne property\_id de la table Property.

**FOREIGN KEY(role\_id):** Clé étrangère faisant référence à la colonne role\_id de la table Role.

### ***Photo***

**id\_photo:** Identifiant unique de la photo. (Type de données : entier)

**jpg\_photo:** Chemin de la photo au format JPG. (Type de données : chaîne de caractères de longueur 250)

**user\_id:** ID d'utilisateur lié (FK). (Type de données : entier, NON NULL)

**property\_id:** ID de propriété liée (FK). (Type de données : entier, NON NULL)

**PRIMARY KEY(id\_photo):** Clé primaire pour la table Photo.

**FOREIGN KEY(user\_id):** Clé étrangère faisant référence à la colonne user\_id de la table Users.

### ***Belongs***

**FOREIGN KEY(photo\_id):** Clé étrangère faisant référence à la colonne photo\_id de la table Photo.

**FOREIGN KEY(property\_id):** Clé étrangère faisant référence à la colonne property\_id de la table Property.

### **Méthodologie de modélisation MERISE.**

Pour commencer, toutes les demandes du client ont été analysées, qui consistent essentiellement à créer des requêtes pour la recherche de biens immobiliers, que ce soit pour l'achat, la vente ou la location, ainsi qu'à permettre aux clients de publier facilement et efficacement des photos de leurs biens immobiliers. De plus, une fonctionnalité d'évaluation a été mise en place pour garantir une précision et un professionnalisme de la part des agents immobiliers. Pour cela, j'ai développé un formulaire gratuit où les spécialistes en immobilier évaluent chaque propriété avec minutie et fournissent la valeur de leur bien au client, parmi d'autres fonctionnalités.

Après avoir analysé les demandes du client, j'ai créé une base de données simple pour recueillir toutes les informations nécessaires à l'application. Plusieurs tables ont été conçues et reliées entre elles pour établir la logique de fonctionnement de l'application.

La table "Role" stocke les différents rôles que peuvent avoir les utilisateurs de l'application, tels que "administrateur" et "utilisateur". Il y a une relation de un à plusieurs entre la table "Role" et la table "Users", car un utilisateur peut avoir un seul rôle, mais un rôle peut être associé à plusieurs utilisateurs.

La table "User" contient des informations à la fois sur les utilisateurs enregistrés et sur les clients. La différence entre un client et un utilisateur enregistré est que le client a un contrat avec l'agence et seul l'administrateur peut enregistrer des informations relatives à ce client. Sauf si le client souhaite insérer ses propres images, auquel cas il doit changer son statut pour devenir un "utilisateur enregistré" avec contrat.

La table "Property" enregistre des informations sur les biens immobiliers. Cette table remplit deux fonctions : enregistrer toutes les informations nécessaires sur les biens immobiliers des clients, et permettre à un utilisateur enregistré de demander une évaluation de la valeur d'un bien immobilier. Un agent immobilier expert répondra ensuite à l'utilisateur en lui indiquant la valeur du bien. Il y a une relation de un à plusieurs entre la table "Users"

et la table "Propertys", car un utilisateur peut avoir plusieurs propriétés, mais une propriété ne peut appartenir qu'à un seul utilisateur.

La table "Category\_contract" est liée à la table "Propertys". Cette relation est de un à plusieurs, ce qui signifie qu'une propriété peut avoir un état de catégorie, mais un état de catégorie peut être attribué à plusieurs propriétés.

La table "Photo" est liée à la table "Users" et à la table "Propertys". Il y a une relation de un à plusieurs entre la table "User" et la table "Photos", car un utilisateur peut télécharger plusieurs photos. De plus, il y a une relation de un à plusieurs entre la table "Photo" et la table "Propertys", car une propriété peut avoir plusieurs photos et une photo peut appartenir à une seule propriété.

La table "ValidationState" est directement liée à la table "Propertys", car elle permet de distinguer entre un client et un utilisateur enregistré qui valide sa propriété. Le client a déjà signé un contrat avec l'agence, tandis que l'utilisateur enregistré n'en a pas pour le moment.

## Niveau Conceptuel MCD

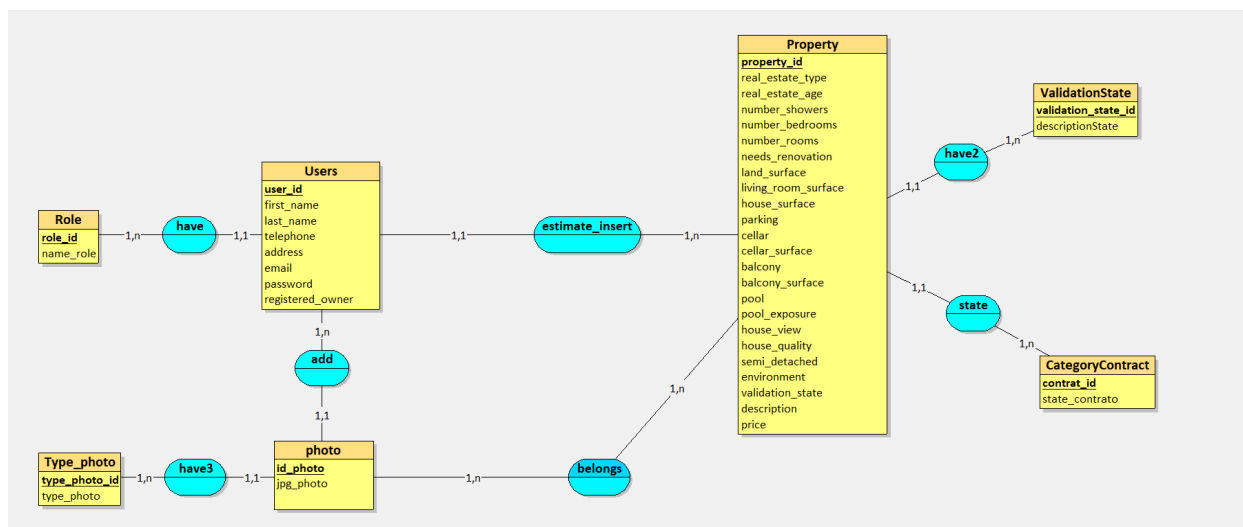


Figure #3 ->Modèle MCD

## Niveau Logique MLD

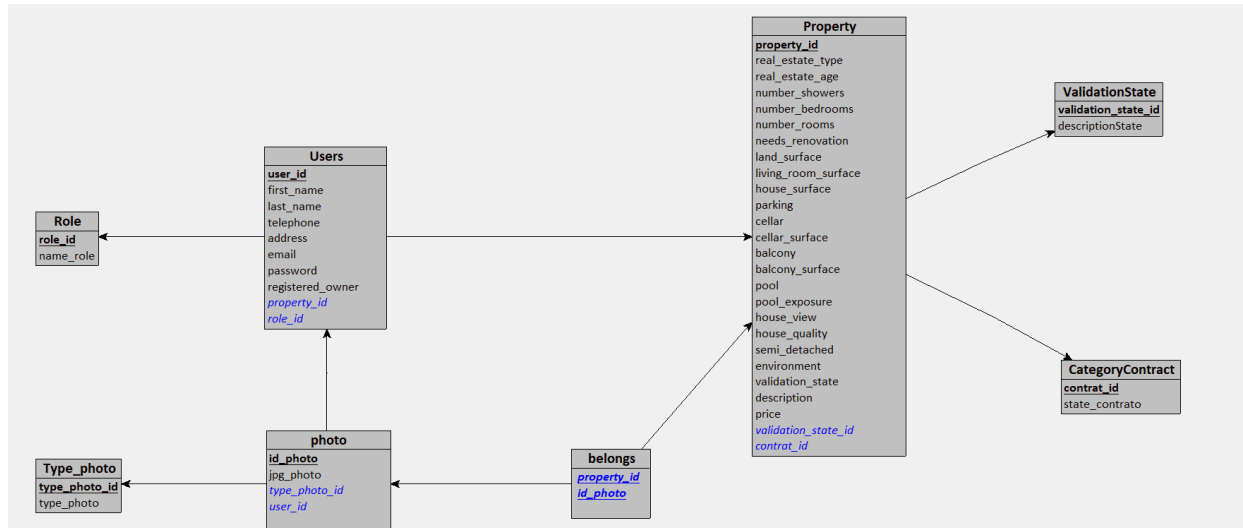


Figure #4 -> Modèle MLD

## Modèle UML

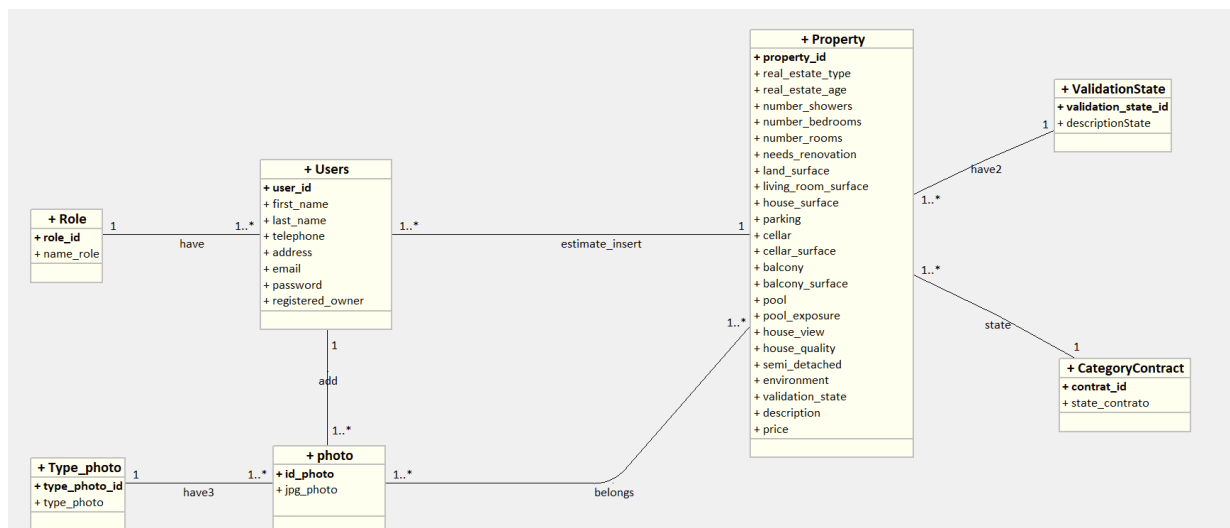


Figure #5 -> Modèle UML

## Maquette ou conception UX

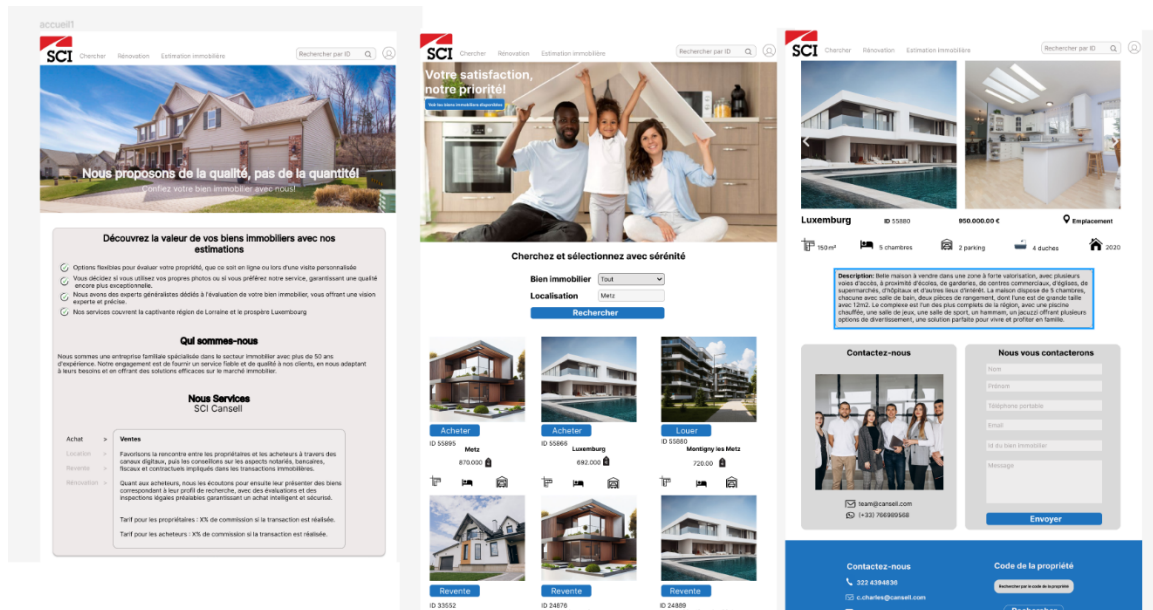


Figure #6 -> Maquette 1

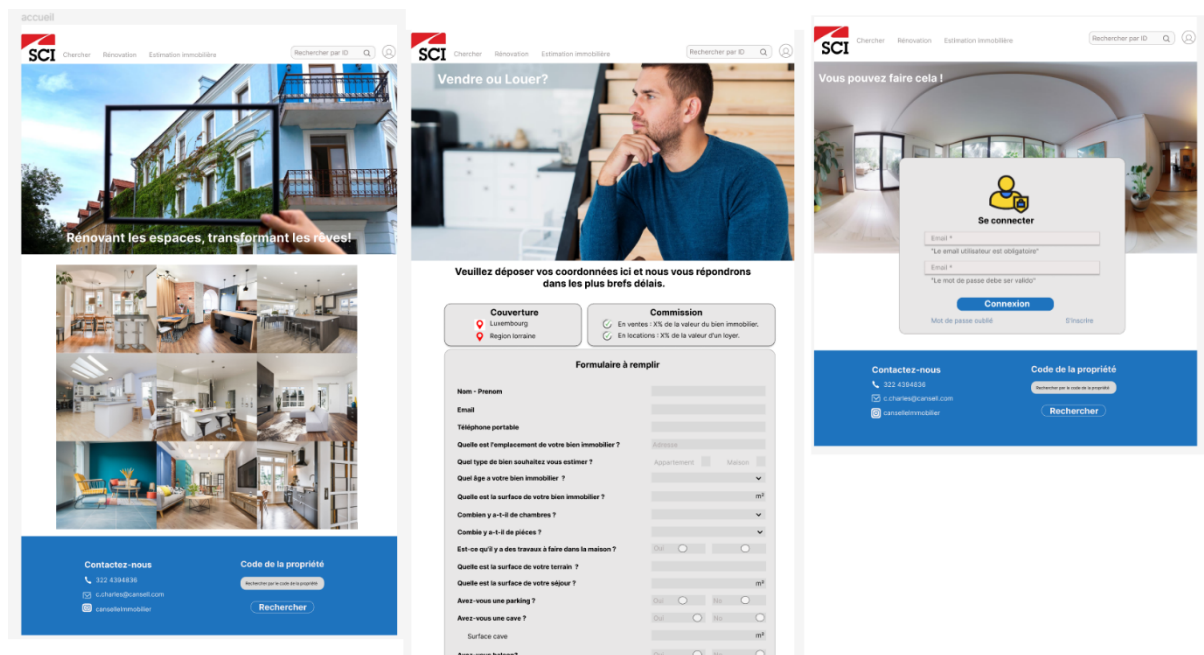


Figure #7 -> Maquette 2

Le design du site Web a été développé en tenant compte des besoins spécifiques du client. Ci-dessous, vous trouverez des informations plus détaillées sur chacune des pages principales:

**Page d'accueil :** Cette page vise à fournir aux visiteurs une vue d'ensemble de l'agence immobilière. Elle inclut des informations pertinentes telles que l'histoire de l'entreprise, son expérience sur le marché et les politiques régissant ses services. De plus, la mission et les valeurs de l'entreprise sont mises en avant pour transmettre confiance et crédibilité.

**Page de recherche :** Sur cette page, les utilisateurs peuvent explorer et rechercher parmi toutes les propriétés disponibles. Ils ont un espace de recherche où ils peuvent saisir l'identifiant de la propriété ou sélectionner un type d'opération, comme la vente, la location ou l'achat. De plus, il est possible de filtrer les résultats par emplacement géographique. Une fois qu'ils trouvent une propriété d'intérêt, ils peuvent accéder à une page dédiée à cette propriété, où des informations détaillées telles que des descriptions, des caractéristiques, des photos et des détails supplémentaires sont présentés.

**Page de rénovation :** Dans cette section, des exemples d'espaces qui ont été rénovés sont présentés. Des photos des propriétés après avoir été soumises à des processus de rénovation et d'amélioration sont montrées. Cela donne aux utilisateurs une idée claire de l'apparence des propriétés après la rénovation, ce qui peut être un facteur important dans leur prise de décision.

**Page d'estimation :** Cette page offre aux utilisateurs la possibilité d'accéder à un formulaire pour effectuer des estimations. Les champs du formulaire sont conçus pour recueillir des informations sur la propriété afin de fournir une réponse précise de sa valeur. Il convient de noter que les champs du formulaire ne sont activés que si l'utilisateur est inscrit sur la plateforme, permettant ainsi une plus grande personnalisation.

**Page d'inscription :** Sur cette page, un accès direct est fourni aux utilisateurs enregistrés pour se connecter à leurs comptes. Si un utilisateur n'est pas encore enregistré, des options claires et visibles lui sont proposées pour qu'il puisse créer un compte sur la plateforme. L'inscription implique de remplir un formulaire avec des informations de base telles que le nom, l'adresse, le téléphone et l'adresse e-mail. Une fois enregistré, l'utilisateur aura accès à des fonctionnalités telles que l'évaluation des biens immobiliers et l'insertion de photos.

Il est important de noter que toutes les pages de l'application Web comportent un pied de page contenant les coordonnées de l'agence immobilière, telles que le numéro de téléphone, l'adresse e-mail et les liens vers les réseaux sociaux. Cela permet aux utilisateurs de contacter facilement l'agence pour obtenir plus d'informations, résoudre des doutes ou poser des questions spécifiques.

En plus des espaces mentionnés ci-dessus, l'application Web inclut des fonctionnalités spécifiques pour les utilisateurs enregistrés et l'administrateur. Les utilisateurs enregistrés ont accès à des boutons visibles leur permettant d'insérer des photos. D'autre part, l'administrateur dispose d'un ensemble complet d'outils lui permettant de gérer et de contrôler les informations des propriétés et des utilisateurs. Cela inclut la capacité de réaliser des modifications, des mises à jour et des suppressions d'informations.

## VI. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité

### Choix des technologies

Pour ce projet, j'ai sélectionné différentes technologies pour répondre aux exigences de développement. Les voici :

**Spring Boot** : C'est un framework Java qui facilite le développement d'applications web en fournissant une configuration automatique et des fonctionnalités intégrées. Avec Spring Boot, j'ai pu simplifier la mise en œuvre de l'architecture MVC (Modèle-Vue-Contrôleur) et bénéficier d'une gestion efficace des requêtes HTTP, de la sécurité et des exceptions.

**ORM Hibernate et JPA** : Hibernate est un framework de mappage objet-relationnel qui facilite la manipulation des données dans une base de données relationnelle. JPA (Java Persistence API) est une spécification qui définit une interface commune pour interagir avec la base de données. Ensemble, Hibernate et JPA me permettent d'effectuer des opérations de création, lecture, mise à jour et suppression (CRUD) sur les entités de manière simplifiée.

**Angular** : Il me permet d'établir facilement une architecture Modèle-Vue-Contrôleur (MVC), ce qui facilite la séparation des composants et améliore la maintenabilité du code. De plus, Angular offre un large éventail de fonctionnalités et d'outils qui aident au développement d'applications web de haute qualité.

### Choix de la base de données

J'ai choisi d'utiliser une base de données relationnelle MariaDB pour ce projet. Ce choix est basé sur la structure des données et les besoins de stockage. Une base de données relationnelle permet de gérer efficacement les relations entre les entités et offre des fonctionnalités avancées pour l'intégrité des données et la gestion des transactions.

### Solution Back-end

La solution back-end repose sur l'utilisation de Spring Boot, Hibernate et JPA. Ces technologies me permettent de développer une couche de logique métier robuste et efficace pour l'application. Spring Boot facilite la configuration et le déploiement du serveur, tandis que Hibernate et JPA simplifient l'interaction avec la base de données et la manipulation des données.

**Respect de l'architecture MVC** : Dans la conception de l'application, j'ai suivi le modèle d'architecture MVC (Modèle-Vue-Contrôleur), où le modèle représente les données et la logique métier, la vue s'occupe de l'interface utilisateur et le contrôleur gère les interactions entre le modèle et la vue.



**Tester notre API :** Pendant le développement, il est important de réaliser des tests exhaustifs de l'application pour assurer son bon fonctionnement. Cela implique de réaliser des tests unitaires pour garantir que les fonctionnalités répondent aux exigences établies. À cette fin, j'ai créé quelques tests unitaires en utilisant des outils tels que POSTMAN.

## **Solution Front-End**

Pour la partie front-end, j'ai utilisé le framework Angular, Angular Material Bootstrap et CSS. Cela me permet de développer une interface utilisateur moderne et réactive pour l'application web. J'ai suivi une architecture spécifique pour organiser et modulariser le code, comme indiqué précédemment, et j'utiliserai des services pour implémenter la logique de l'application et effectuer des appels vers le back-end.

**Sécurité :** La sécurité est une considération importante dans le développement de toute application. J'ai mis en place des mesures de sécurité telles que l'utilisation de l'authentification basée sur les tokens JWT (JSON Web Tokens). J'ai également utilisé des gardes d'authentification (AuthGuard) côté front-end pour contrôler l'accès à certaines fonctionnalités en fonction des rôles et des permissions des utilisateurs.

J'ai défini l'environnement de développement en utilisant Spring Boot, ce qui m'a permis de mapper les entités des tables dans la base de données. Cela inclut la création de tables, la définition des noms, des colonnes et des relations entre les entités. Grâce à des fonctionnalités telles que l'ORM (Mapping Objet-Relationnel) et la bibliothèque Hibernate, je peux générer automatiquement des requêtes. Par exemple, la méthode `save()` génère une requête d'insertion et `findAll()` renvoie une requête pour obtenir tous les enregistrements d'une table, entre autres.

En ce qui concerne le développement du back-end, j'ai suivi une approche détaillée dans la modélisation de la base de données et de ses relations, telles que 'One-to-Many, Many-to-One et Many-to-Many', en fonction des besoins spécifiques de ma base de données relationnelle. En utilisant Spring Boot, j'ai réussi à créer la base de données dans son intégralité, car cet outil me permet non seulement de créer les tables, mais aussi d'intégrer le modèle de manière efficace.

Ensuite, j'ai créé des interfaces via des référentiels ou DAO (Data Access Object) qui facilitent la communication avec la base de données (en utilisant MariaDB). Ces référentiels contiennent les méthodes nécessaires pour effectuer des opérations de lecture, écriture, mise à jour et suppression (CRUD) dans la base de données. Cela m'a permis d'établir efficacement les requêtes et les transactions nécessaires dans des classes concrètes.

De plus, Spring Boot facilite les opérations de base du CRUD (Créer, Lire, Mettre à jour, Supprimer), simplifiant l'interaction avec la base de données et évitant la nécessité d'écrire manuellement des requêtes SQL. Certains des méthodes que j'ai utilisées sont `save()`, `findById()`, `findAll()`, `update()` et `delete()`.

Pour intégrer le client web de l'agence immobilière, j'ai utilisé la classe 'HttpClient' d'Angular. Cette classe m'a permis d'effectuer différentes requêtes HTTP telles que GET, POST, PUT et DELETE, facilitant ainsi la gestion des réponses du serveur. En utilisant cette fonctionnalité, j'ai pu établir la couche de communication nécessaire entre le frontend et le backend de l'application.

De plus, j'ai créé une page qui sert de vue principale de l'application. Dans cette vue, la logique de connexion et d'inscription des utilisateurs est mise en œuvre, ainsi que les pages de navigation. Pour améliorer la modularité et la réutilisation du code, j'ai utilisé des composants d'Angular pour mettre en œuvre la logique spécifique de chaque section de l'application.

D'autre part, j'ai utilisé un package de services dans Angular pour établir la communication avec le backend développé en Spring Boot. Ces services agissent comme des intermédiaires entre le frontend et le backend, permettant l'échange de données et la gestion des opérations nécessaires pour obtenir et envoyer des informations au serveur.

En ce qui concerne la conception, j'ai décidé d'utiliser CSS, Bootstrap et Angular Material. Ce sont des outils très populaires et faciles à utiliser qui offrent une large gamme de styles et de composants prédéfinis. Avec une grande quantité d'informations et de documentation disponibles, je peux rapidement mettre en œuvre des conceptions attrayantes et fonctionnelles dans mon application. De plus, ces outils me permettent de rendre l'application responsive, ce qui est essentiel de nos jours car les utilisateurs accèdent aux applications depuis une variété d'appareils.

### Comportement intégral de Spring Boot et Angular

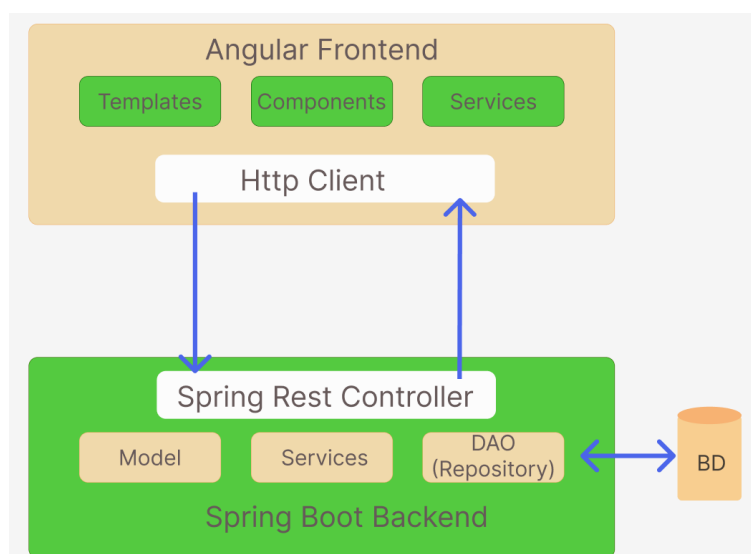


Figure #8 -> Comportement intégral de Spring Boot et Angular

Dans cette image, le comportement intégral de Spring Boot et Angular est représenté. Du côté du backend, nous observons le pattern MVC (Modèle-Vue-Contrôleur), qui se connecte

à la base de données via le DAO ou le Répertoire. Depuis Angular, un client se connecte au backend via le HttpClient.

Lorsqu'une opération est souhaitée, par exemple, insérer un utilisateur, une méthode dans le service du frontend est appelée en utilisant les méthodes HTTP telles que POST, GET ou DELETE. Le service appelle le contrôleur via le HttpClient, établissant ainsi une connexion. À son tour, le contrôleur s'appuie sur un service qui communique avec le DAO par injection de dépendances.

Ensuite, le DAO se connecte à la base de données et reçoit une réponse, qui est ensuite renvoyée au service. Le service renvoie ensuite les informations au frontend (Angular), et le résultat sont affiché dans un modèle.

Ainsi, le HttpClient permet d'établir une connexion entre le backend et le frontend, où le pattern MVC est utilisé pour organiser la logique de l'application et l'accès à la base de données.

## VII. Réalisations du candidat comportant les extraits de code les plus significatifs et en les argumentant, y compris pour la sécurité

### a. Back-end

#### Création d'une entité dans Spring Boot

##### *Classe User*

```
package com.cansell.web.model;

import lombok.Data;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;

@Entity
@Data
@Table(name = "users", uniqueConstraints =
@UniqueConstraint(columnNames = "email"))
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY,
generator = "native")
    @GenericGenerator(name = "native", strategy = "native")
    @Column(name = "user_id")
    private Long userId;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "phone")
    private String phone;

    @Column(name = "address")
    private String address;

    @Column(name = "email")
    private String email;

    @Column(name = "password")
    private String password;
```

```

@Column(name = "registered_owner")
private boolean registeredOwner;

@ManyToOne
@JoinColumn(name = "property_id")
private Propertys propertyId;

@ManyToOne
@JoinColumn(name = "role_id")
private Role role;
}

```

La relation entre les deux tables Users et Role est définie à l'aide de l'annotation @ManyToOne dans l'entité Users. Cela indique que chaque enregistrement de la table Users est associé à un enregistrement de la table Role.

### ***L'entité Users***

Dans cette entité, vous avez deux relations @ManyToOne, une avec l'entité Propertys et une autre avec l'entité Role.

@ManyToOne avec Propertys : Cela indique que chaque utilisateur (Users) est associé à une propriété (Propertys). L'annotation @JoinColumn(name = "property\_id") spécifie la colonne dans la table users qui sera utilisée comme clé étrangère pour établir cette relation avec la table property. En d'autres termes, dans la table users, la colonne property\_id contiendra l'ID de la propriété associée à chaque utilisateur.

@ManyToOne avec Role : Cela indique que chaque utilisateur (Users) est associé à un rôle (Role). Tout comme dans la relation précédente, l'annotation @JoinColumn(name = "role\_id") spécifie la colonne dans la table users qui sera utilisée comme clé étrangère pour établir cette relation avec la table role. Dans la table users, la colonne role\_id contiendra l'ID du rôle associé à chaque utilisateur.

### ***L'entité Role***

Représente les rôles disponibles dans le système. Chaque enregistrement dans cette table représente un rôle spécifique, et chaque rôle a un identifiant unique (roleId) et un nom (nameRole).

La relation entre Users et Role est établie par le biais de la colonne role\_id dans la table users, qui stocke l'identifiant du rôle associé à chaque utilisateur. Avec cette configuration, vous pouvez attribuer un rôle spécifique à chaque utilisateur et, grâce à la relation

@ManyToOne, obtenir toutes les informations complètes sur le rôle auquel appartient un utilisateur.

## Création d'un DAO (Data Access Object)

### Classe UserDao

```
package com.cansell.web.dao;

import com.cansell.web.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.Optional;
@Repository
public interface UserDao extends JpaRepository<User, Long> {
    @Query("From User u JOIN FETCH u.role WHERE u.email = :email")
    Optional<User> findByEmail(@Param("email") String email);
}
```

La classe UserDao est une interface qui étend JpaRepository. Cette interface fournit des méthodes pour effectuer des opérations CRUD (Create, Read, Update, Delete) sur l'entité User, qui représente les utilisateurs de l'application. En étendant JpaRepository, la classe hérite automatiquement des méthodes telles que save(), findAll(), findById(), deleteById(), etc., ce qui simplifie l'interaction avec la base de données pour les utilisateurs.

La méthode personnalisée findByEmail() dans cette classe utilise l'annotation @Query pour effectuer une requête personnalisée dans la base de données et rechercher un utilisateur par son adresse e-mail (email). Le résultat de la requête est facultatif (Optional), ce qui signifie qu'elle peut renvoyer un utilisateur s'il est trouvé ou une valeur vide s'il ne l'est pas.

### La Classe RoleDao

Est également une interface qui étend JpaRepository. Cette interface est utilisée pour interagir avec l'entité Role, qui représente les rôles disponibles dans le système. En étendant JpaRepository, cette classe hérite automatiquement des méthodes CRUD pour l'entité Role,

ce qui permet d'effectuer des opérations de base sur la table des rôles dans la base de données.

La méthode personnalisée `findByNameRole()` dans cette classe est utilisée pour rechercher un rôle par son nom (`nameRole`). Il prend simplement le nom du rôle en tant que paramètre et renvoie l'objet `Role` correspondant s'il est trouvé.

## Création d'un contrôleur dans Spring Boot

### *Classe ConnexionController*

```
package com.cansell.web.controller;

import com.cansell.web.dao.UserDao;
import com.cansell.web.model.User;
import com.cansell.web.security.JwtUtils;
import com.cansell.web.security.UserDetailsImpl;
import com.cansell.web.security.UserDetailsServiceImpl;
import io.jsonwebtoken.Claims;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@CrossOrigin( origins = "http://localhost:4200/" )
@RestController
public class ConnexionController {
    @Autowired
    AuthenticationManager authenticationManager;
    @Autowired
    UserDetailsServiceImpl userDetailsService;
    @Autowired
    UserDao userDao;
}
```

```

@Autowired
PasswordEncoder passwordEncoder;
@Autowired
JwtUtils jwtUtils;

@PostMapping("/login")
public ResponseEntity<String> connexion(@RequestBody User
user){
    System.out.println("user.toString() = ingrese a la
conexion" + user.toString());
    UserDetailsImpl userDetails;
    try {
        userDetails =
(UserDetailsImpl)authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(
                user.getEmail(),
                user.getPassword()
            )
        ).getPrincipal();
        System.out.println("userDetails = " + userDetails);
    } catch (AuthenticationException e){
        System.out.println("e = " + e);
        return new ResponseEntity<>(HttpStatus.FORBIDDEN);
    }
    System.out.println("userDetails = " + userDetails);
    return new
ResponseEntity<>(jwtUtils.generateJwt(userDetails),
HttpStatus.OK);
}

@GetMapping("/profil")
public ResponseEntity<User>
getProfil(@RequestHeader("Authorization") String bearer){
    System.out.println(" ingrese al profil ");
    String jwt = bearer.substring(7);

    Claims donnees = jwtUtils.getData(jwt);

    Optional<User> utilisateur =
userDao.findByEmail(donnees.getSubject());

    if (utilisateur.isPresent()){
        System.out.println("hola hola");
        return new ResponseEntity<>(utilisateur.get(),
HttpStatus.OK);
    }
}

```



```
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);  
    }  
}
```

Cette classe est un contrôleur Spring qui gère les requêtes liées à l'authentification et à la génération de tokens JWT. Elle contient deux méthodes principales :

1. Méthode connexion () : Cette méthode est appelée lorsque l'utilisateur tente de se connecter à l'application. Elle utilise l'AuthenticationManager pour authentifier les informations d'identification de l'utilisateur (adresse e-mail et mot de passe). Si l'authentification est réussie, un token JWT est généré à l'aide de JwtUtils et UserDetailsImpl. Le token est renvoyé dans la réponse HTTP si l'authentification est valide, et une erreur HTTP 403 est renvoyée si les informations d'identification sont incorrectes.
2. Méthode getProfil() : Cette méthode est utilisée pour obtenir le profil de l'utilisateur authentifié. Elle utilise le token JWT envoyé dans l'en-tête d'autorisation pour obtenir les données de l'utilisateur (adresse e-mail) à l'aide de JwtUtils. Ensuite, elle recherche l'utilisateur correspondant dans la base de données via UserDao et renvoie le profil de l'utilisateur s'il est trouvé, ou une erreur HTTP 404 s'il ne l'est pas.

### **Description algorithme de connexion pour les utilisateurs et les administrateurs**

#### ***Pour se connecter en tant qu'utilisateur***

1. Le client envoie une requête POST à "/login" avec les informations d'identification (adresse e-mail et mot de passe) dans le corps de la requête.
2. Le contrôleur ConnexionController reçoit la requête et authentifie l'utilisateur en utilisant l'AuthenticationManager. Si les informations d'identification sont valides, un jeton JWT est généré pour l'utilisateur à l'aide de la classe JwtUtils.
3. Le contrôleur répond avec un ResponseEntity contenant le jeton JWT généré, et le statut de la réponse est HttpStatus.OK.
4. Le client reçoit le jeton JWT et le stocke localement (par exemple, dans le stockage de session ou les cookies) pour les futures requêtes.

#### ***Pour se connecter en tant qu'administrateur***

1. Le client effectue une requête vers une URL protégée qui nécessite le rôle "ADMINISTRATEUR" pour y accéder, par exemple, "/admin/\*\*".

2. Le filtre `JwtFilter` intercepte la requête et vérifie la validité du jeton JWT envoyé dans l'en-tête d'autorisation.
3. Si le jeton est valide, le filtre extrait les revendications (claims) du jeton, y compris le rôle de l'utilisateur authentifié.
4. Le filtre vérifie si l'utilisateur a le rôle "ADMINISTRATEUR" dans les revendications du jeton. Si c'est le cas, la requête continue et est traitée normalement ; sinon, l'accès est refusé et une réponse avec le statut `HttpStatus.FORBIDDEN` est renvoyée.

## Sécurité: Spring Boot Security et des tokens JWT

### **Classe `JwtUtils`**

Cette classe fournit des méthodes pour générer et vérifier les tokens JWT. Elle comporte trois méthodes principales :

**1. Méthode `generateJwt()`** : Cette méthode est utilisée pour générer un token JWT basé sur les détails de l'utilisateur (`UserDetailsImpl`) fournis en paramètre. Les informations de l'utilisateur sont ajoutées sous forme de données personnalisées (claims) au token JWT, puis il est signé à l'aide d'un secret ("azerty") pour garantir son authenticité.

**2. Méthode `getData()`** : Cette méthode est utilisée pour obtenir les données (claims) stockées dans un token JWT donné. Elle utilise le même secret ("azerty") pour vérifier la signature du token et s'assurer qu'il n'a pas été modifié.

**3. Méthode `isTokenValid()`** : Cette méthode est utilisée pour vérifier si un token JWT donné est valide. Elle utilise la méthode `getData()` pour obtenir les données du token et vérifie si la signature est valide. Si le token est valide, elle renvoie `true` ; sinon, elle renvoie `false`.

### **Classe `UserDetailsImpl`**

Cette classe implémente l'interface `UserDetails` de Spring Security, qui représente les détails de l'utilisateur nécessaires pour l'authentification et l'autorisation. Elle contient des méthodes qui renvoient les informations de l'utilisateur, telles que les rôles, les mots de passe, etc. Elle est également utilisée pour charger les détails de l'utilisateur depuis la base de données dans la classe `UserDetailsServiceImpl`.

### **Classe `UserDetailsServiceImpl`**

Cette classe implémente l'interface `UserDetailsService` de Spring Security et est utilisée pour charger les détails de l'utilisateur depuis la base de données en fonction de l'adresse e-mail

fournie lors de l'authentification. Elle utilise UserDao pour rechercher l'utilisateur dans la base de données et renvoie un objet UserDetailsImpl contenant les détails de l'utilisateur.

Il est important de noter que le jeton JWT contient des informations sur l'utilisateur, y compris son adresse e-mail et ses rôles. Cela permet au serveur de vérifier et de contrôler les autorisations d'accès pour différentes fonctionnalités et URL en fonction du rôle de l'utilisateur.

Cet algorithme fournit une méthode sécurisée et efficace pour authentifier les utilisateurs et les administrateurs dans l'application, évitant ainsi la nécessité de maintenir l'état de session côté serveur. De plus, l'utilisation de JWT fournit une couche supplémentaire de sécurité, car le jeton est signé et peut être vérifié par le serveur pour garantir son authenticité.

## b. Front-end

### Création d'une modal sous Angular

#### *Classe HomeComponent.html*

```
import {
  Component,
  HostListener,
  AfterViewInit,
  ChangeDetectorRef,
  OnInit,
} from '@angular/core';
import { Router, RouterModule } from '@angular/router';
import { LoginService } from 'src/app/services/login.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css'],
})
export class HomeComponent {
  userRole: string | null = null;

  constructor(
    private cdRef: ChangeDetectorRef,
    public login: LoginService,
    private router: Router
  ) {}
}
```

```

isConnected: boolean = false;

ngOnInit(): void {
  this.cdRef.detectChanges();
  this.userRole = this.login.getUserRole();
}

@HostListener('window:popstate', ['$event'])
onPopState(event: any) {
  if (this.login.isLoggedIn()) {
    this.logout();
  }
}

public logout() {
  this.login.logout();
  window.location.reload();
  this.router.navigate(['/login']);
}

selectedOption: string = 'achat';

showContent(option: string) {
  this.selectedOption = option;
}

toggleConnection(): void {
  this.isConnected = !this.isConnected;
}
}

```

### Clase HomeComponent.ts

```

<app-navbar></app-navbar>
<div class="content">
  <!-- Contenedor de la imagen -->
  <div class="image-container">
    
  </div>

```

```

</div>

<!-- Recuadro de texto -->
<div class="text-box">
  <h1 style="text-align: center">
    Découvrez la valeur de vos biens immobiliers avec nos estimations
  </h1>
  <br />
  <p>
    * Options flexibles pour évaluer votre propriété, que ce soit en
ligne ou
    lors d'une visite personnalisée
  </p>
  <p>
    * Vous décidez si vous utilisez vos propres photos ou si vous
préférez
    notre service, garantissant une qualité encore plus
exceptionnelle.
  </p>
  <p>
    * Nous avons des experts généralistes dédiés à l'évaluation de
votre bien
    immobilier, vous offrant une vision experte et précise.
  </p>
  <p>
    * Nos services couvrent la captivante région de Lorraine et le
prospère
    Luxembourg
  </p>
  <br />
  <h1 style="text-align: center">Qui sommes-nous</h1>
  <br />
  <p>
    Nous sommes une entreprise familiale spécialisée dans le secteur
immobilier avec plus de 50 ans d'expérience. Notre engagement est
de
    fournir un service fiable et de qualité à nos clients, en nous
adaptant à
    leurs besoins et en offrant des solutions efficaces sur le marché
immobilier.
  </p>
</div>
</div>

```

```

<div class="content2">
  <div class="options-container">
    <ul class="options-list">
      <li (click)="showContent('achat')">Achat </li>
      <li (click)="showContent('location')">Location </li>
      <li (click)="showContent('revente')">Revente </li>
      <li (click)="showContent('renovation')">Rénovation </li>
    </ul>
    <div class="selected-option-content">
      <div *ngIf="selectedOption === 'achat'" class="option-content">
        <h1>Achat</h1>
        <br />
        <p>
          Favorisons la rencontre entre les propriétaires et les
acheteurs à
          travers des canaux digitaux, puis les conseillons sur les
aspects
          notariés, bancaires, fiscaux et contractuels impliqués dans
les
          transactions immobilières. Quant aux acheteurs, nous les
écoutons pour
          ensuite leur présenter des biens correspondant à leur profil
de
          recherche, avec des évaluations et des inspections légales
préalables
          garantissant un achat intelligent et sécurisé. Tarif pour les
propriétaires : X% de commission si la transaction est
réalisée. Tarif
          pour les acheteurs : X% de commission si la transaction est
réalisée.
        </p>
      </div>
      <div *ngIf="selectedOption === 'location'"
class="option-content">
        <p>Contenido para Location.</p>
      </div>
      <div *ngIf="selectedOption === 'revente'" class="option-content">
        <p>Contenido para Revente.</p>
      </div>
      <div *ngIf="selectedOption === 'renovation'"
class="option-content">
        <h1>Rénovation</h1>

```

```

    <p>
      En ce qui concerne notre spécialisation dans la rénovation
      immobilière, nous sommes fiers de proposer des services
complets de
      réaménagement et de rénovation afin d'optimiser la valeur et
      l'apparence des propriétés. Notre équipe d'experts en
construction et
      en design travaille en étroite collaboration avec nos clients
pour
      transformer les espaces, des petites améliorations aux
rénovations
      complètes, en portant une attention particulière aux détails
et en
      mettant l'accent sur la qualité et l'excellence. Faites-nous
confiance
      pour transformer votre propriété en un espace rénové et
attrayant,
      répondant à vos attentes et mettant en valeur son potentiel
sur le
      marché immobilier
    </p>
  </div>
</div>
</div>
</div>
<app-footer></app-footer>

```

Le composant "HomeComponent" de l'application Angular. Le code gère la logique de la page d'accueil ("home") et fournit des fonctionnalités pour afficher différentes options de contenu liées à l'achat, la location, la revente et la rénovation de biens immobiliers.

1. Dans le composant "HomeComponent", certaines variables telles que "userRole" et "isConnected" sont initialisées pour gérer les informations de l'utilisateur connecté et son rôle.
2. La fonction "ngOnInit()" est un hook du cycle de vie du composant qui s'exécute une fois que le composant a été initialisé. Dans ce cas, il est utilisé pour obtenir le rôle de l'utilisateur connecté via le service "LoginService" et mettre à jour la variable "userRole".
3. Le décorateur "@HostListener" est utilisé pour écouter les événements du navigateur. Dans ce cas, il écoute l'événement "popstate" qui se produit lorsqu'un changement est effectué dans l'historique du navigateur (comme appuyer sur le bouton "retour"). Si

l'utilisateur est connecté (utilisant le service "LoginService"), la fonction "logout()" est appelée pour déconnecter l'utilisateur.

4. La fonction "logout()" gère la déconnexion de l'utilisateur en utilisant le service "LoginService". Ensuite, la page est rechargée pour mettre à jour l'état de la connexion et l'utilisateur est redirigé vers la page de connexion ("/login").

5. La variable "selectedOption" est utilisée pour conserver l'état de l'option sélectionnée dans la liste des options. Cette variable est mise à jour lorsque l'utilisateur clique sur l'une des options de la liste.

6. La fonction "showContent (option: string)" est utilisée pour changer le contenu affiché dans la section centrale de la page, en fonction de l'option sélectionnée dans la liste. Cette fonction met à jour la variable "selectedOption" avec la valeur de l'option sélectionnée.

7. Le code HTML à l'intérieur du composant "HomeComponent" affiche le contenu de la page d'accueil. Il comprend une image de fond, une boîte de texte avec des informations sur les services d'estimation immobilière, des informations sur l'entreprise et une liste d'options liées à l'achat, la location, la revente et la rénovation de biens immobiliers.

8. Selon l'option sélectionnée dans la liste, le contenu correspondant est affiché dans la section centrale de la page. Le contenu pour chaque option est défini dans des sections distinctes avec la directive '\*ngIf\*', qui affiche ou masque le contenu en fonction de l'option sélectionnée.

## Création d'un menu de navigation sous Angular

### Classe NavbarComponent.ts

```
import {
  Component,
  HostListener,
  AfterViewInit,
  ChangeDetectorRef,
  OnInit,
} from '@angular/core';
import { Router, RouterModule } from '@angular/router';
import { LoginService } from 'src/app/services/login.service';

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css'],
})
```



```

}))
export class NavbarComponent implements OnInit {
  userRole: string | null = null;
  showOptions: boolean = false;
  selectedOption: string = 'achat';

  constructor(
    private cdRef: ChangeDetectorRef,
    public login: LoginService,
    private router: Router
  ) {}

  isConnected: boolean = false;

  ngOnInit(): void {
    this.cdRef.detectChanges();
    this.userRole = this.login.getUserRole();
    console.log('userRole:', this.userRole);
    this.isConnected = this.login.isLoggedIn();
  }

  // @HostListener('window:popstate', ['$event'])
  // onPopState(event: any) {
  //   if (this.login.isLoggedIn()) {
  //     this.logout();
  //   }
  // }

  public logout() {
    this.login.logout();
    window.location.reload();
    this.router.navigate(['/login']);
  }

  // showContent(option: string) {
  //   this.selectedOption = option;
  // }

  toggleConnection(): void {
    this.isConnected = !this.isConnected;
  }

  toggleOptions(): void {

```

```

    this.showOptions = !this.showOptions;
  }

  redirectToRegistro() {
    this.router.navigate(['/login']); // Reemplaza 'registro' con la
ruta a la página de registro
  }

  // Método para redirigir a la página de salida
  redirectToSalir() {
    this.router.navigate(['']); // Reemplaza 'salir' con la ruta a la
página de salida
  }
}

```

Cette barre de navigation est responsable de montrer les options de menu et les options de connexion aux utilisateurs.

Le composant Navbar implémente OnInit, qui est utilisé pour initialiser les propriétés et détecter les changements dans le cycle de vie du composant.

Dans la fonction ngOnInit(), certaines propriétés sont initialisées, telles que userRole et isConnected. userRole stocke le rôle de l'utilisateur actuel (s'il est connecté) obtenu via le service LoginService. isConnected vérifie si l'utilisateur est connecté ou non en utilisant la méthode isLoggedIn() du même service.

La fonction logout() est responsable de déconnecter l'utilisateur et de le rediriger vers la page de connexion. Elle utilise également window.location.reload() pour recharger la page et mettre à jour l'état de connexion.

Les fonctions toggleConnection() et toggleOptions() sont utilisées pour basculer l'affichage de l'état de connexion et des options utilisateur, respectivement.

Les fonctions redirectToRegistro() et redirectToSalir() sont utilisées pour rediriger l'utilisateur vers la page d'inscription et la page d'accueil, respectivement.

Dans le code HTML à l'intérieur de navbar.component.html, la barre de navigation est affichée en utilisant la balise mat-toolbar d'Angular Material. Cette barre contient un logo d'entreprise, des boutons pour différentes options de menu, un champ de recherche, l'image de l'utilisateur et les options de connexion.

Le menu d'options est affiché en utilisant une liste (ul) qui est masquée ou affichée en fonction de l'état de showOptions. Les options incluent "Connexion", "S'inscrire" (inscription) et "Accueil" (page d'accueil).

Lorsque l'utilisateur clique sur l'image de l'utilisateur, les options de connexion sont affichées ou masquées en utilisant la fonction toggleOptions().

### Sécurisé: Authentification par JWT

```
import { Component } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';
import { LoginService } from 'src/app/services/login.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent {
  loginData = {
    email: '',
    password: '',
  };

  constructor(
    private snack: MatSnackBar,
    private loginService: LoginService,
    private router: Router
  ) {}

  formSubmit() {

    if (
      this.loginData.email.trim() == '' ||
      this.loginData.email.trim() == null
    ) {
      this.snack.open('El email es obligatorio!!', 'Aceptatar', {
        duration: 5000,
      });
    }
    return;
  }
}
```

```

    }
    if (
      this.loginData.password.trim() == '' ||
      this.loginData.password.trim() == null
    ) {
      this.snack.open('El password es obligatorio!!', 'Aceptatar', {
        duration: 5000,
      });
      return;
    }

    console.log(this.loginData);

    this.loginService.generateToken(this.loginData).subscribe(
      (data: any) => {
        console.log('estoy aca' + data);
        this.loginService.loginUser(data);
        setTimeout(() => {
          this.loginService.getCurrentUser().subscribe((user: any) => {
            console.log(user);
            this.loginService.setUser(user);
            console.log('estoy en el usuario ' + user);

            if (this.loginService.getUserRole() ==
'ROLE_ADMINISTRATEUR') {

              console.log('admin ');
              this.router.navigate(['/admin']);
              console.log('entre al administrador' + 'user ');
              this.loginService.loginStatusSubjec.next(true);
            } else if (this.loginService.getUserRole() ==
'ROLE_UTILISATEUR') {
              //DASHBOARD USER
              console.log('entre al administrador' + 'user ');
              // window.location.href = '/user-dashboard';
              this.router.navigate(['/estimation-immobiliere']);
              this.loginService.loginStatusSubjec.next(true);
            } else {
              console.log('salida esta ');
              this.loginService.logout();
            }
          });
        }, 1000);
      }
    );
  }
}

```

```

    },
    (error: any) => {
      console.log(error);
      this.snack.open('Registro invalido! intente de nuevo',
'Aceptar', {
        duration: 5000,
      });
      alert('error al generar el token');
    }
  );
}
}

```

Il gère le processus de connexion de l'utilisateur via le formulaire de connexion sur la page de connexion.

Le formulaire de connexion (<form (ngSubmit)="formSubmit()">) est associé à la fonction formSubmit() dans le composant LoginComponent.

Lorsque l'utilisateur clique sur le bouton "Connexion", la fonction formSubmit() est exécutée.

La fonction formSubmit() effectue les validations suivantes :

Vérifie si l'adresse e-mail (email) et le mot de passe (password) fournis par l'utilisateur sont valides et non vides. S'il manque des informations, un message d'avertissement est affiché à l'aide du service MatSnackBar d'Angular Material. Si les informations de l'utilisateur sont valides, le service LoginService est appelé pour générer un jeton JWT à l'aide de la méthode generateToken(). Le jeton JWT est obtenu en réponse dans la variable data.

Ensuite, le service LoginService est utilisé pour stocker le jeton JWT et les détails de l'utilisateur authentifié à l'aide des méthodes loginUser() et setUser().

Après avoir obtenu le jeton JWT et les détails de l'utilisateur, la méthode getCurrentUser() du service LoginService est utilisée pour obtenir des informations supplémentaires sur l'utilisateur depuis le serveur.

Selon le rôle de l'utilisateur (obtenu via le service LoginService), l'utilisateur est redirigé vers la page correspondante en utilisant l'objet Router d'Angular. Si l'utilisateur est administrateur, il est redirigé vers la page "/admin" ; sinon, il est redirigé vers la page "/estimation-immobiliere". De plus, la variable loginStatusSubject du service LoginService est définie sur true pour indiquer que l'utilisateur s'est connecté avec succès.

S'il y a une erreur pendant le processus de connexion, un message d'erreur est affiché à l'aide du service MatSnackBar.

Le code HTML dans login.component.html contient le formulaire de connexion et quelques styles et éléments de conception pour afficher la page de connexion.

Ce code gère la logique de connexion de l'utilisateur et utilise des services tels que MatSnackBar pour afficher des messages de notification et Router pour rediriger l'utilisateur vers des pages spécifiques en fonction de son rôle. De plus, en haut de la page, il indique si l'utilisateur est connecté en tant qu'utilisateur normal ou en tant qu'administrateur.

### Inscription d'un utilisateur

```
import { Component } from '@angular/core';
import { MatSnackBar, MatSnackBarModule } from
'@angular/material/snack-bar';
import { UserService } from 'src/app/services/user.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css'],
})
export class SignupComponent {
  public user = {
    firstName: '',
    lastName: '',
    phone: '',
    address: '',
    email: '',
    password: '',
  };

  constructor(private userService: UserService, private snack:
MatSnackBar) {} //se inyecta en el constructor

  formSubmit() {
    console.log(this.user);
    if (this.user.email == '' || this.user.email == null) {
      this.snack.open('E-mail est requis!!', 'Accepter', {
        duration: 5000,
        verticalPosition: 'top',
        horizontalPosition: 'center',
```

```

    });
    return;
  }

  this.userService.enregistrerUtilisateur(this.user).subscribe(
    (response: any) => {
      console.log(response);
      alert('Utilisateur créé ');
      Swal.fire(
        'utilisateur enregistré avec succès dans le système',
        'success'
      );
    },
    (error) => {
      console.log(error);
      if (error.status === 400) {
        this.snack.open('Utilisateur déjà enregistré!!', 'Accepter',
{
          duration: 5000,
          verticalPosition: 'top',
          horizontalPosition: 'center',
        });
      } else {
        this.snack.open('Erreur dans le système!!', 'Accepter', {
          duration: 5000,
          verticalPosition: 'top',
          horizontalPosition: 'center',
        });
      }
    }
  );
}
}

```

Le composant et le fichier HTML travaillent ensemble pour fournir un formulaire d'inscription sur une page d'inscription. L'utilisateur peut saisir ses données dans les champs du formulaire et les soumettre pour s'inscrire. En cas de succès de l'inscription, un message de confirmation est affiché.

Le composant 'SignupComponent' gère la logique du formulaire d'inscription. Dans la classe du composant, un objet 'user' est déclaré avec des propriétés telles que 'firstName', 'lastName', 'phone', 'address', 'email' et 'password'. Ces propriétés sont liées au formulaire

d'inscription via les directives '[(ngModel)]'. Dans le constructeur du composant, les services 'UserService' et 'MatSnackBar' sont injectés. Le service 'UserService' est utilisé pour envoyer les informations d'inscription de l'utilisateur au serveur, tandis que 'MatSnackBar' est un composant d'Angular Material utilisé pour afficher des messages de notification. La fonction 'formSubmit()' est appelée lorsque l'utilisateur soumet le formulaire d'inscription. Cette fonction effectue plusieurs validations sur les champs du formulaire et affiche des messages de notification en cas d'erreurs ou de succès de l'inscription. Si l'inscription est réussie, un message de confirmation est affiché en utilisant 'Swal.fire()' de SweetAlert2.

Le fichier HTML contient le formulaire d'inscription avec des balises Material Design. La directive '(ngSubmit)="formSubmit()"' est utilisée dans le formulaire pour appeler la fonction 'formSubmit()' du composant lorsque l'utilisateur soumet le formulaire. Chaque champ du formulaire est lié aux propriétés de l'objet 'user' en utilisant la directive '[(ngModel)]'. Par exemple, '[(ngModel)]="user.lastName"' lie le champ du nom de famille à la propriété 'lastName' de l'objet 'user'. Le bouton d'inscription ('button mat-raised-button color="primary">Inscription /button>') a la balise 'mat-raised-button' d'Angular Material pour le styliser comme un bouton en relief avec la couleur d'accentuation principale. Le bouton de réinitialisation ('button type="reset" style="margin-left: 20px;" mat-raised-button color="accent">Nettoyer /button>') a la balise 'mat-raised-button' et l'attribut 'type="reset"' pour réinitialiser les champs du formulaire à leur état d'origine.



## VIII. Présentation des jeux d'essai

## Tests sur l'API Spring Boot

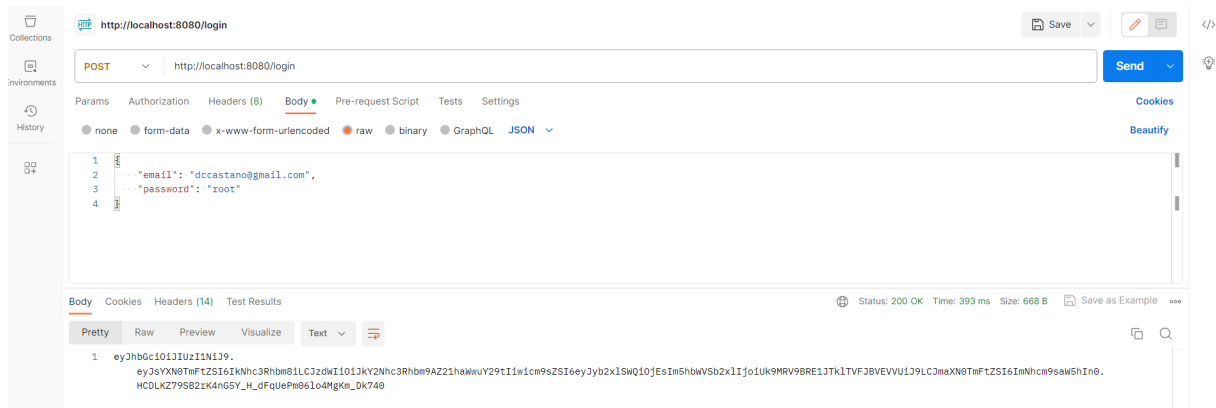


Figure #9 -> Authentification de l'utilisateur

Dans l'image fournie, on peut voir un test réalisé à l'aide de l'outil Postman pour valider la connexion d'un utilisateur. Pendant ce test, l'adresse e-mail et le mot de passe de l'utilisateur ont été saisis dans le but d'obtenir le jeton d'authentification (Token).

Postman est un outil largement utilisé pour tester les API et les services web, et dans ce cas, il a été utilisé pour vérifier la fonctionnalité d'authentification du système. En saisissant les identifiants de l'utilisateur, le système a généré et fourni un jeton d'authentification valide, qui est essentiel pour autoriser et valider les futures demandes de l'utilisateur dans l'application.

La réussite de l'obtention du jeton d'authentification démontre l'efficacité du processus de connexion de l'utilisateur et que la connexion avec le serveur a été établie correctement.

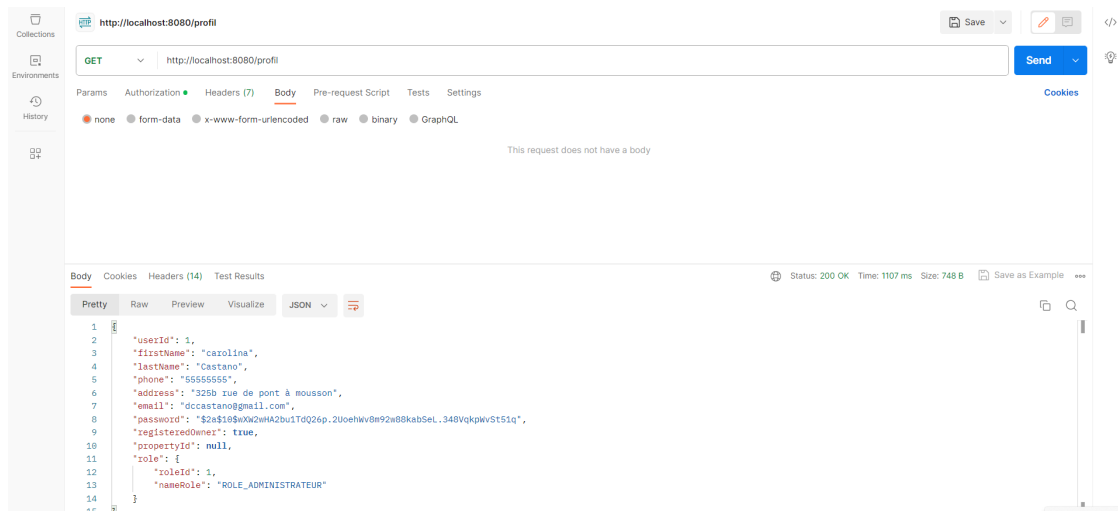


Figure #10 -> Spécifiques de l'utilisateur

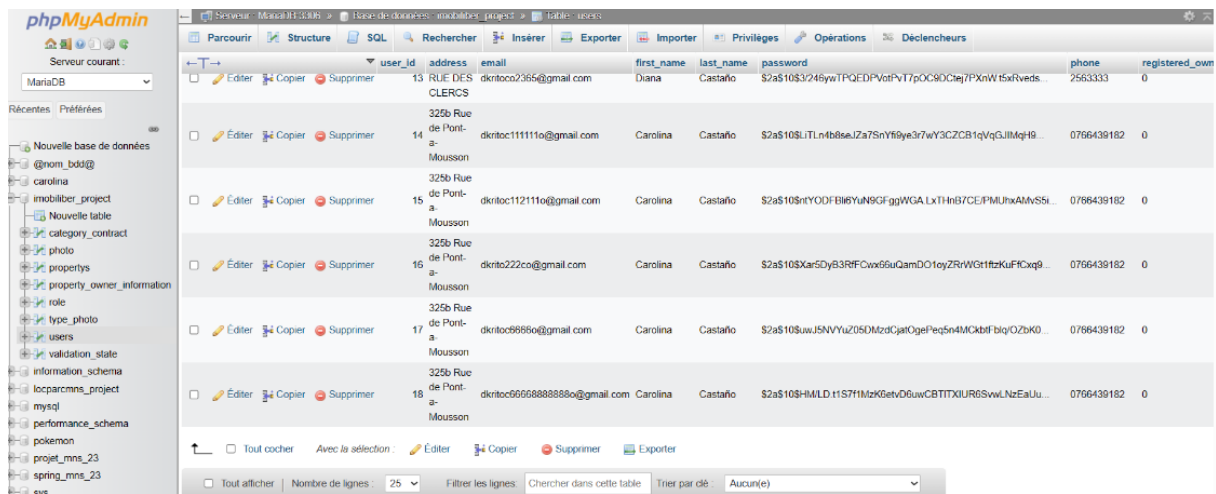
Une fois que nous avons obtenu l'authentification via le jeton, nous pouvons également récupérer les informations de l'utilisateur qui a été authentifié, comme le montre l'image suivante.

L'obtention de l'authentification via le jeton nous permet d'accéder aux données spécifiques de l'utilisateur qui s'est connecté à l'application. Ces informations peuvent inclure des détails tels que le nom, l'adresse e-mail, le rôle et toute autre information pertinente sur l'utilisateur.

En utilisant le jeton d'authentification dans les demandes ultérieures, nous pouvons autoriser et valider les actions de l'utilisateur dans l'application, en nous assurant qu'il n'a accès qu'aux fonctionnalités et aux ressources autorisées en fonction de son rôle et de ses permissions.

L'image fournie montre comment les informations de l'utilisateur sont renvoyées par le serveur en réponse à une demande d'authentification valide avec le jeton. Cela démontre l'efficacité du système d'authentification et sa capacité à fournir des données personnalisées et sécurisées à chaque utilisateur authentifié.

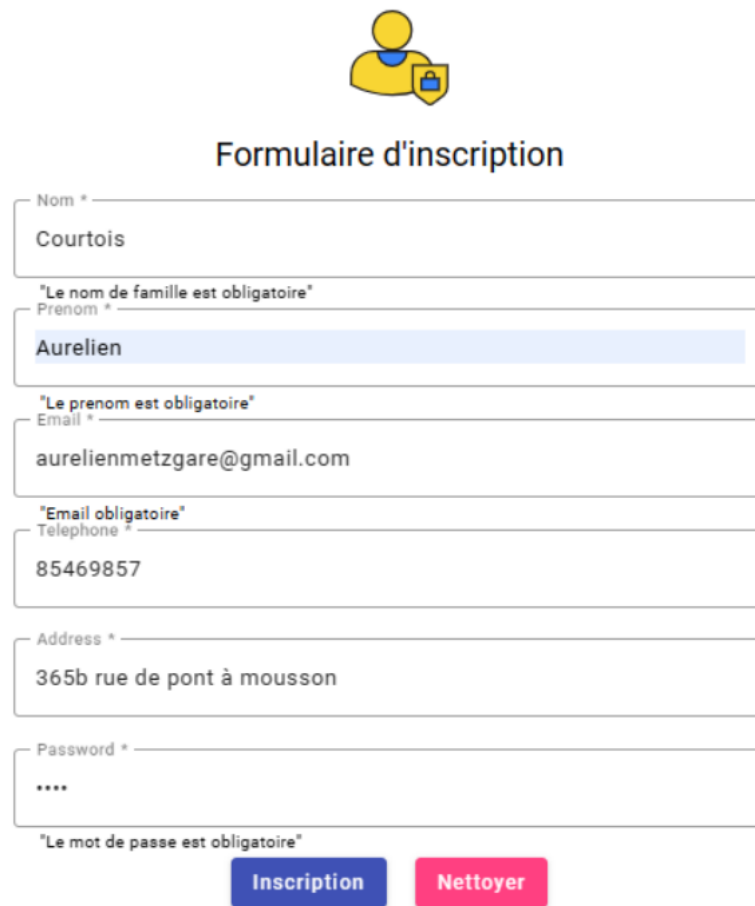
## Tests sur la partie Front-end



The screenshot shows the phpMyAdmin interface with the 'users' table selected. The table contains the following data:

user_id	address	email	first_name	last_name	password	phone	registered_owm
13	RUE DES CLERCS	dkritoc2365@gmail.com	Diana	Castello	\$2a\$10\$3z469wTPQEDPvotVt17pOC9DCq7/PXnWt5xRveds...	2563333	0
14	325b Rue de Pont-a-Mousson	dkritoc111111@gmail.com	Carolina	Castello	\$2a\$10\$LTLn4b8seJz7SnY69e37wY3C2CB1qVqGJIMqH9...	0766439182	0
15	325b Rue de Pont-a-Mousson	dkritoc112111@gmail.com	Carolina	Castello	\$2a\$10\$ntYODfB96Yn9GFggWGA LxTHi67CE/PMUhxAMvS5...	0766439182	0
16	325b Rue de Pont-a-Mousson	dkritoc222co@gmail.com	Carolina	Castello	\$2a\$10\$Xar5DYB3RfCwx6GuQamDO1oyZrRWGt1tzbKuFCxq9...	0766439182	0
17	325b Rue de Pont-a-Mousson	dkritoc9666o@gmail.com	Carolina	Castello	\$2a\$10\$uuvJSNVYuz05DMzdCjaOgePeg5n4MCKbfbq/OzbK0...	0766439182	0
18	325b Rue de Pont-a-Mousson	dkritoc666688888o@gmail.com	Carolina	Castello	\$2a\$10\$HMLD t1S7H1MzK6etvO6uwcBTITXOUR6SvwLnZuEaU...	0766439182	0

Figure #11 -> Bases de données avant l'inscription



**Formulaire d'inscription**

Nom \*  
Courtois

"Le nom de famille est obligatoire"

Prenom \*  
Aurelien

"Le prenom est obligatoire"

Email \*  
aurelienmetzgare@gmail.com

"Email obligatoire"

Telephone \*  
85469857

Address \*  
365b rue de pont à mousson

Password \*  
\*\*\*\*

"Le mot de passe est obligatoire"

**Inscription** **Nettoyer**

Figure #12 -> Enregistrement de l'utilisateur

Figure #13 -> Message d'inscription réussie

user_id	address	email	first_name	last_name	password	phone	registered_ownership
13	METZ RUE DES CLERCS	dkritoco2365@gmail.com	Diana	Castarfo	\$2a\$10\$3/246ywTPQEDPVotPV7pOC9DCIej7FXnW15xRveds...	2563333	0
14	325b Rue de Pont-a-Mousson	dkritoc111111o@gmail.com	Carolina	Castarfo	\$2a\$10\$LTLn4b8seJ2a7SnY89e3r7wY3CZCB1qVqGJIMqH9...	0766439182	0
15	325b Rue de Pont-a-Mousson	dkritoc112111o@gmail.com	Carolina	Castarfo	\$2a\$10\$ntYODFBi6YUN9GFggWGA LxTHnB7CE/PMUhxAMvSSi...	0766439182	0
16	325b Rue de Pont-a-Mousson	dkritoc222co@gmail.com	Carolina	Castarfo	\$2a\$10\$Xar5Dy63RIFCwx66uQamDO1oyZRWG11tZKuFICxq9...	0766439182	0
17	325b Rue de Pont-a-Mousson	dkritoc6996o@gmail.com	Carolina	Castarfo	\$2a\$10\$uwJ5NVYuzZ05DMzdCjatOgePeg5n4MCKbFblyOZbK0...	0766439182	0
18	325b Rue de Pont-a-Mousson	dkritoc6996888888o@gmail.com	Carolina	Castarfo	\$2a\$10\$HwLD11S711MzK6etVd6uwCBTITXIU9BSwLzEalUu...	0766439182	0
19	365b rue de pont à mousson	aurelienmetzgare@gmail.com	Aurelien	Courtois	\$2a\$10\$qqRgvjER7MTYIBIDkFbOZvAmms9Li56SCj7h3cOt...	85469857	0

Figure #14 -> Bases de données après l'inscription

Dans l'image fournie, on peut voir comment un utilisateur peut s'inscrire au système. L'utilisateur remplit un formulaire avec ses informations, y compris son nom, son adresse e-mail et son mot de passe. En cliquant sur le bouton d'inscription, ces informations sont envoyées au serveur pour traitement.

Sur le serveur, plusieurs actions importantes sont effectuées pour garantir la sécurité et l'authenticité de l'utilisateur. Tout d'abord, le mot de passe fourni par l'utilisateur est soumis à un processus de chiffrement en utilisant des algorithmes de sécurité avancés. Ce

chiffrement transforme le mot de passe en un format illisible, ce qui augmente considérablement la protection des informations sensibles de l'utilisateur.

Ensuite, les données de l'utilisateur, y compris le mot de passe chiffré, sont stockées en toute sécurité dans la base de données du système. Cela garantit que les données de l'utilisateur sont protégées et ne sont accessibles qu'aux personnes disposant des autorisations appropriées.

Une fois que l'utilisateur s'est inscrit avec succès, un jeton d'authentification unique est généré et renvoyé au front-end. Ce jeton agit comme une identification numérique pour l'utilisateur et lui permet d'accéder aux zones et aux fonctions autorisées de l'application.

## IX. Liste des Figures

- \* Figure #1 -> Gestion de projet
- \* Figure #2 -> Diagramme de classe
- \* Figure #3 -> Modèle MCD
- \* Figure #4 -> Modèle MLD
- \* Figure #5 -> Modèle UML
- \* Figure #6 -> Maquette 1
- \* Figure #7 -> Maquette 2
- \* Figure #8 -> Comportement intégral de Spring Boot et Angular
- \* Figure #9 -> Authentification de l'utilisateur
- \* Figure #10 -> Spécifiques de l'utilisateur
- \* Figure #11 -> Bases de données avant l'inscription
- \* Figure #12 -> Enregistrement de l'utilisateur
- \* Figure #13 -> Message d'inscription réussie
- \* Figure #14 -> Bases de données après l'inscription