|  |  |  |  |
| --- | --- | --- | --- |
| **Document Technique :**  **Stephi Place Software** | | | |
| **Document Technique** | | | |
| Référence | YNOV-2020- SOFTWARE-JAVA | Auteurs | J.Deblaecker & I.Riviere |

# Informations

|  |  |
| --- | --- |
| Nom du projet | Stephi Place Software |
| Type de document | Documentation technique logiciel |
| Date | 02/06/2020 |
| Version | 1.6 |
| Mots-clés | Fonctionnel – Technologie – Architecture – Versioning - Convention de nommage – Planification |
| Auteurs | DEBLAECKER Jérémy  RIVIERE Isatys |

# Rédaction et modifications

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Nom | Description |
| 1.1 | 14/05/20 | DEBLAECKER | Création du document technique. |
| 1.2 | 18/05/20 | DEBLAECKER | Ajout du MCD. |
| 1.3 | 23/05/20 | DEBLAECKER | Ajout de l’étude de besoins. |
| 1.4 | 28/05/20 | DEBLAECKER | Ajout des diagrammes de séquences. |
| 1.5 | 30/05/20 | DEBLAECKER | Ajout des wireframes. |
| 1.6 | 02/06/20 | DEBLAECKER | Annexes UML(M.Palermo) et vérification final. |

Table des matières

[Informations 2](#_Toc41982993)

[Rédaction et modifications 2](#_Toc41982994)

[1. Etudes des besoins : 3](#_Toc41982995)

[I. Présentation du projet 3](#_Toc41982996)

[II. Besoins fonctionnels 3](#_Toc41982997)

[2. Conceptualisation : 4](#_Toc41982998)

[I. Utilisateurs de l’application 4](#_Toc41982999)

[II. Diagrammes de séquences 4](#_Toc41983000)

[a. Connexion d’un administrateur 5](#_Toc41983001)

[b. Ajouter/modifier/supprimer un agent immobilier 6](#_Toc41983002)

[c. Ajouter/modifier/supprimer une agence 7](#_Toc41983003)

[d. Connexion d’un agent 8](#_Toc41983004)

[e. Ajouter/modifier/supprimer un bien 9](#_Toc41983005)

[f. Ajouter/modifier/supprimer des documents 10](#_Toc41983006)

[g. Modifier informations de l’agent 11](#_Toc41983007)

[III. Conception de la base de données 11](#_Toc41983008)

[IV. Modèle logique de données 12](#_Toc41983009)

[3. Environnement de travail et technologies utilisées : 13](#_Toc41983010)

[4. Langage utilisé : 13](#_Toc41983011)

[5. Wireframe : 13](#_Toc41983012)

[a. Connexion agent immobilier 14](#_Toc41983013)

[b. Espace agents 15](#_Toc41983014)

[c. Gérer les clients 15](#_Toc41983015)

[d. Gérer les biens 16](#_Toc41983016)

[e. Gestion des documents 16](#_Toc41983017)

[f. Tableau de bord 17](#_Toc41983018)

[g. Connexion administrateur 17](#_Toc41983019)

[h. Espace administrateur 18](#_Toc41983020)

[i. Gérer les agences 18](#_Toc41983021)

[j. Gérer les agents 19](#_Toc41983022)

[6. Annexes 19](#_Toc41983023)

[a) Notation UML 19](#_Toc41983024)

# Etudes des besoins :

## Présentation du projet

La société Stephi Place Real Estate désire développer un logiciel qui permettra de faciliter la gestion des données du site par les administrateurs et les agents immobiliers.

## Besoins fonctionnels

Le logiciel comportera différentes fonctionnalités nécessaires pour les agents immobiliers et les administrateurs afin de faciliter la gestion des données.

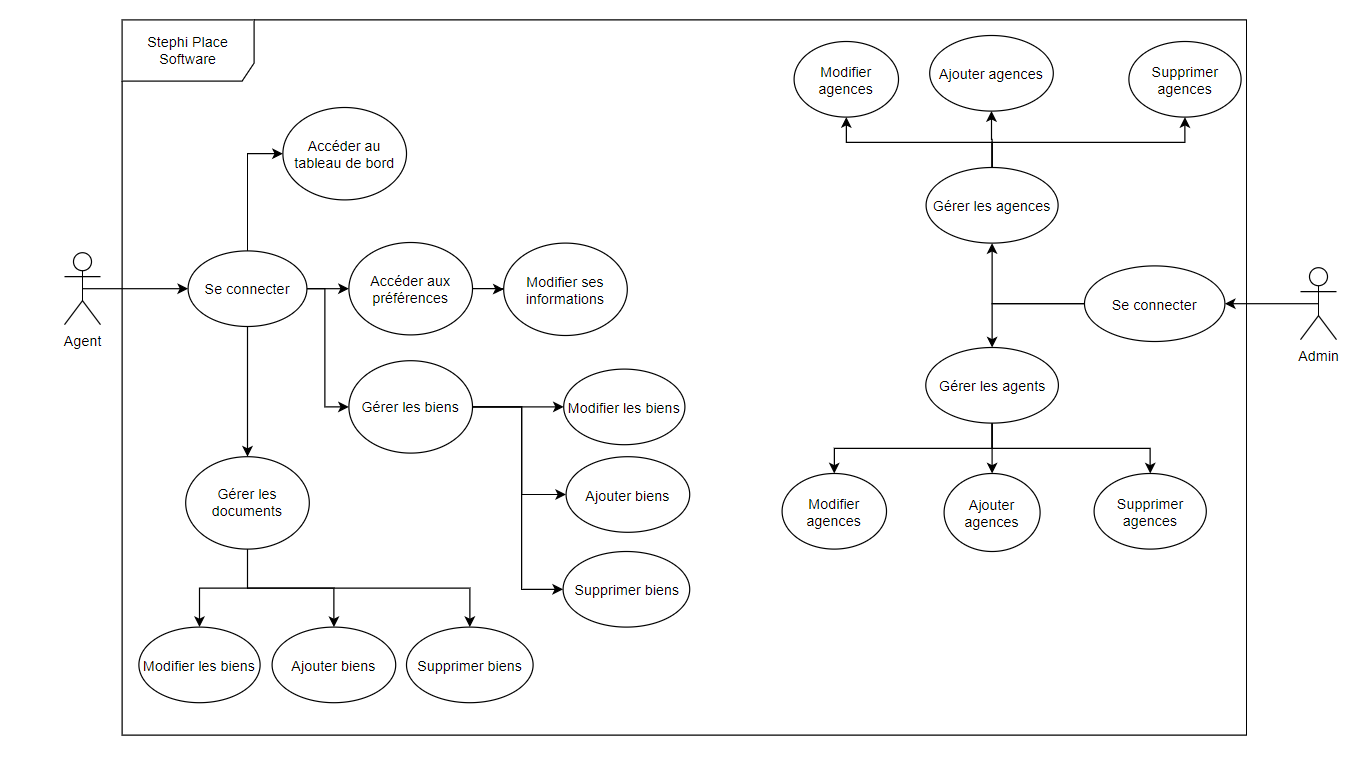
* Espace Agent Immobilier :
  + Préférences compte agent immobilier
  + Gestion des clients
  + Gestion des biens
  + Tableau de bord agent immobilier
  + Gestion des documents
* Espace Administration
  + Gestion des agences
  + Gestion des agents immobiliers

# Conceptualisation :

## Utilisateurs de l’application

Le logiciel sera utilisable par deux types d’utilisateurs :

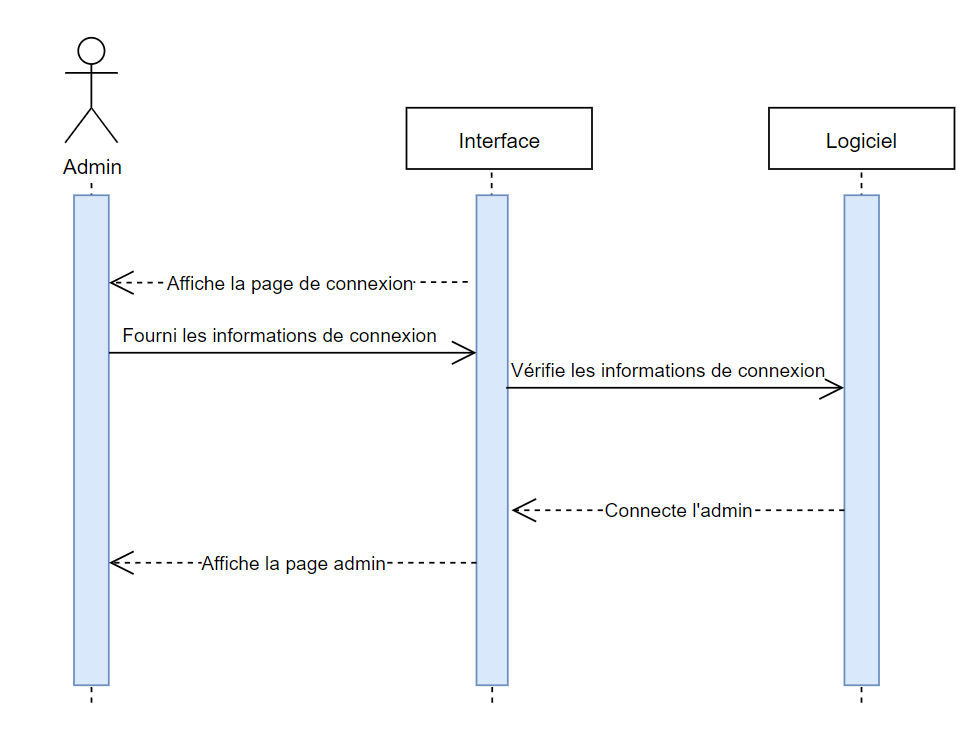
* Les administrateurs qui pourront gérer les agences et leurs agents immobiliers.
* Les agents immobiliers qui pourront gérer les biens, les documents, accéder à leur préférence, accéder au tableau de bord.



## Diagrammes de séquences

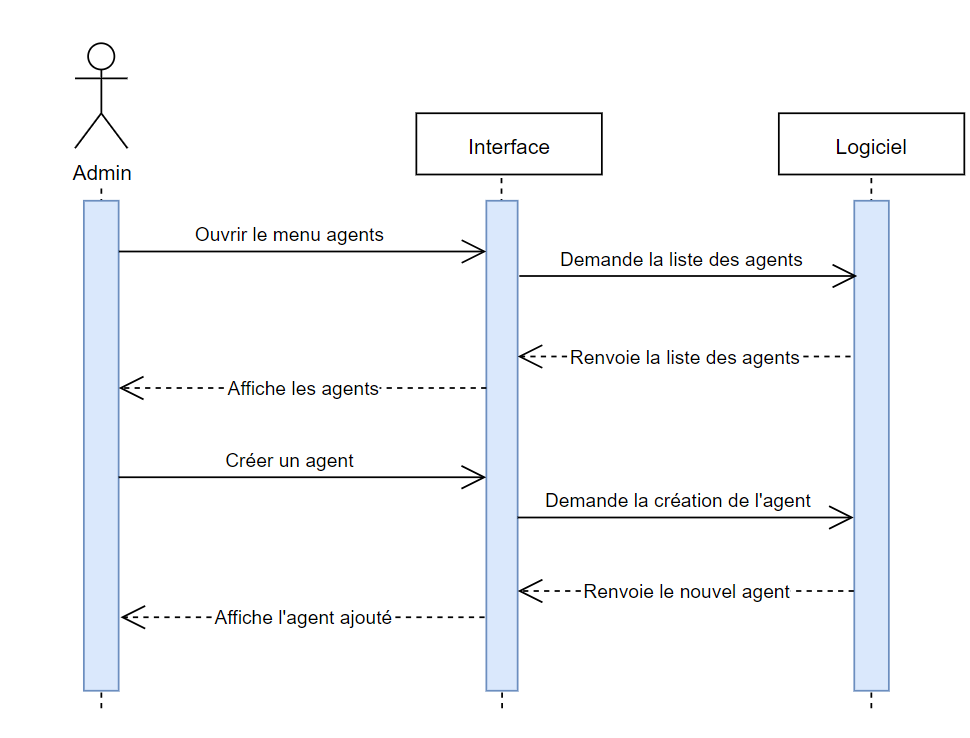
Dans cette partie nous allons voir les diagrammes de séquences associés à notre logiciel.

### Connexion d’un administrateur



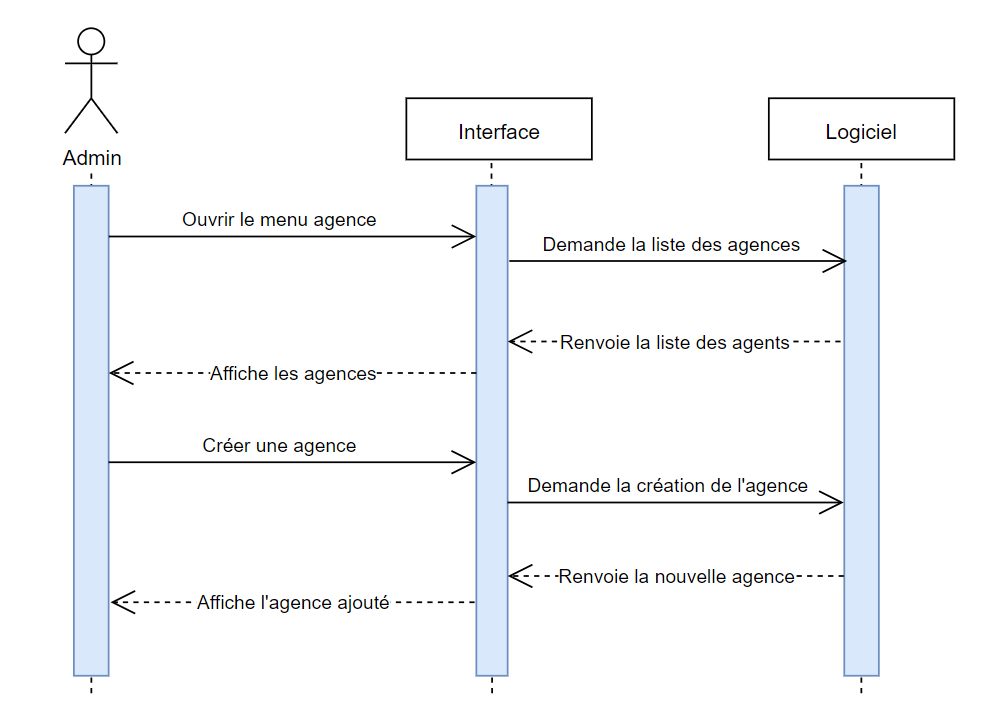
Ici nous pouvons voir comment se passe la connexion pour l’administrateur. Une fois ses informations de connexions fournis et validés par le logiciel, celui-ci le renvoie vers la page administrateur.

### Ajouter/modifier/supprimer un agent immobilier



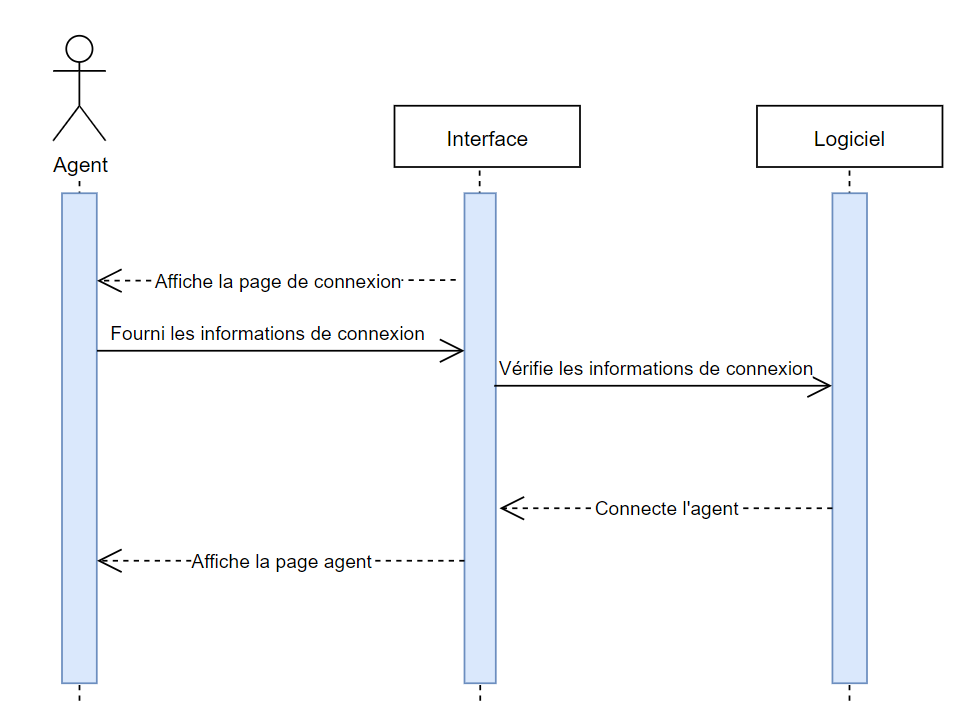
Ici nous pouvons voir comment se passe l’ajout, la suppression et la modification d’agents par l’administrateur. Dans le cas de ce diagramme nous partons du principe que l’administrateur désir créer un client.

### Ajouter/modifier/supprimer une agence



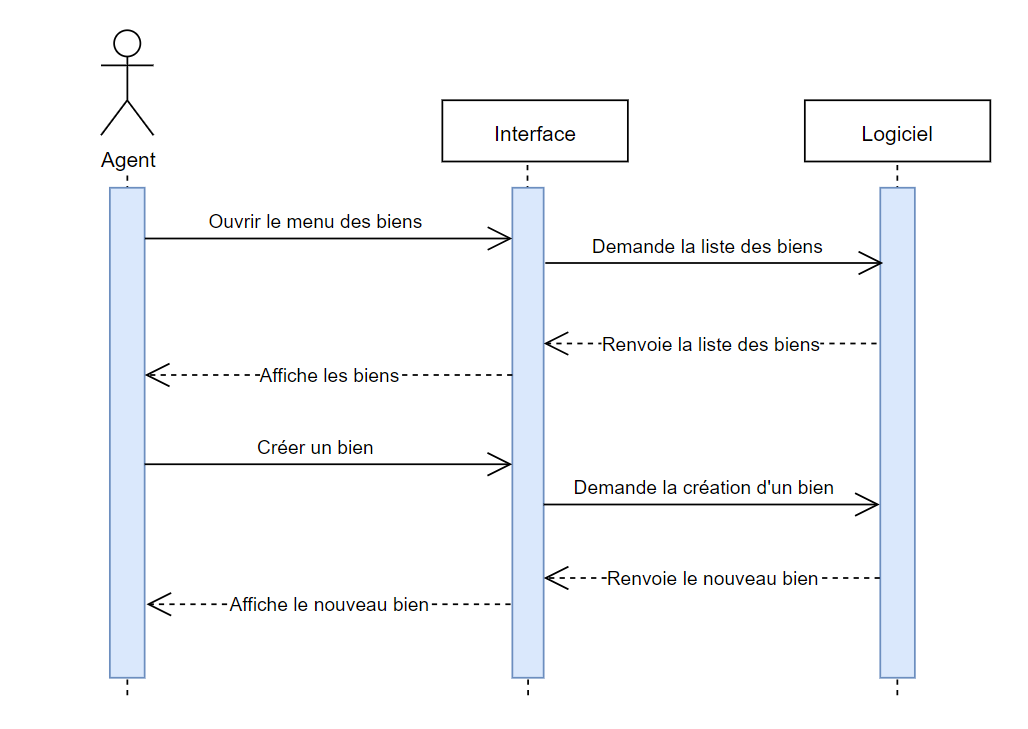
Ici nous pouvons voir comment se passe l’ajout, la suppression et la modification d’agences par l’administrateur. Dans le cas de ce diagramme nous partons du principe que l’administrateur désir créer une agence.

### Connexion d’un agent



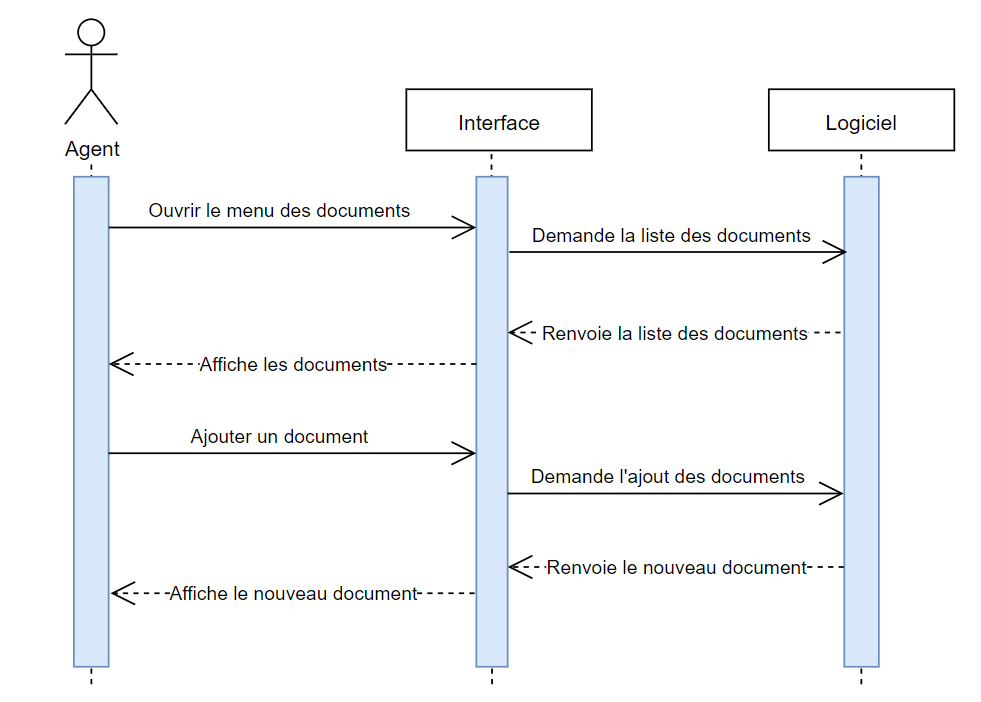
Ici nous pouvons voir comment se passe la connexion pour l’agent immobilier. Une fois ses informations de connexions fournis et validés par le logiciel, celui-ci le renvoie vers la page agent immobilier.

### Ajouter/modifier/supprimer un bien



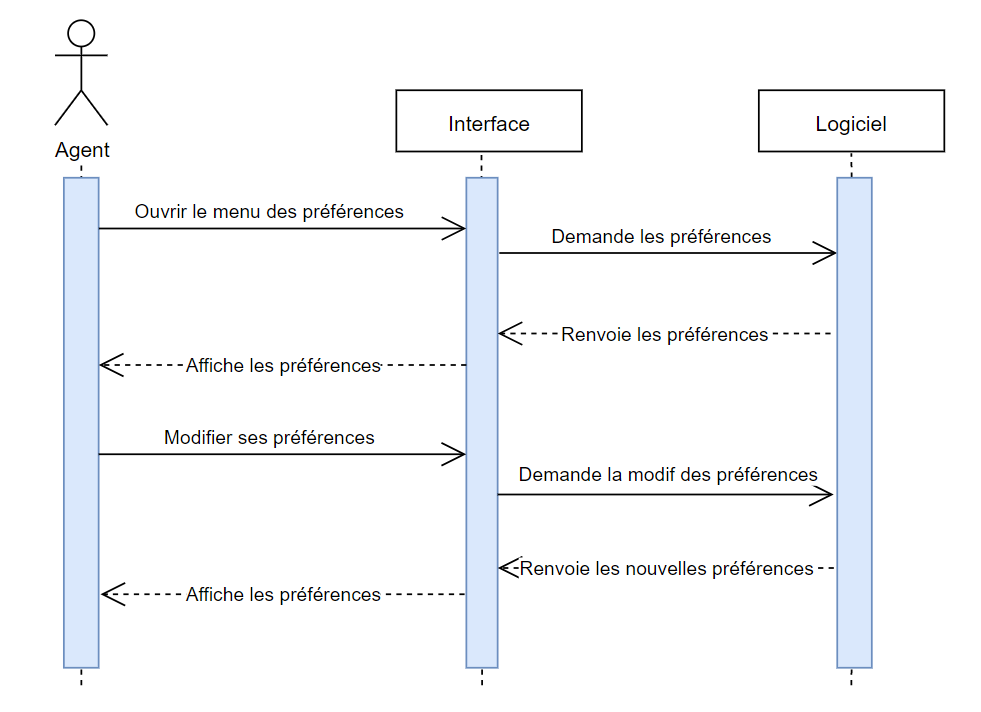
Ici nous pouvons voir comment se passe l’ajout, la suppression et la modification de biens immobiliers par un agent immobilier. Dans le cas de ce diagramme nous partons du principe que l’agent désir créer un bien immobilier.

### Ajouter/modifier/supprimer des documents



Ici nous pouvons voir comment se passe l’ajout, la suppression et la modification de biens documents associer à bien immobilier. Dans le cas de ce diagramme nous partons du principe que l’agent désir créer ajouter un nouveau document.

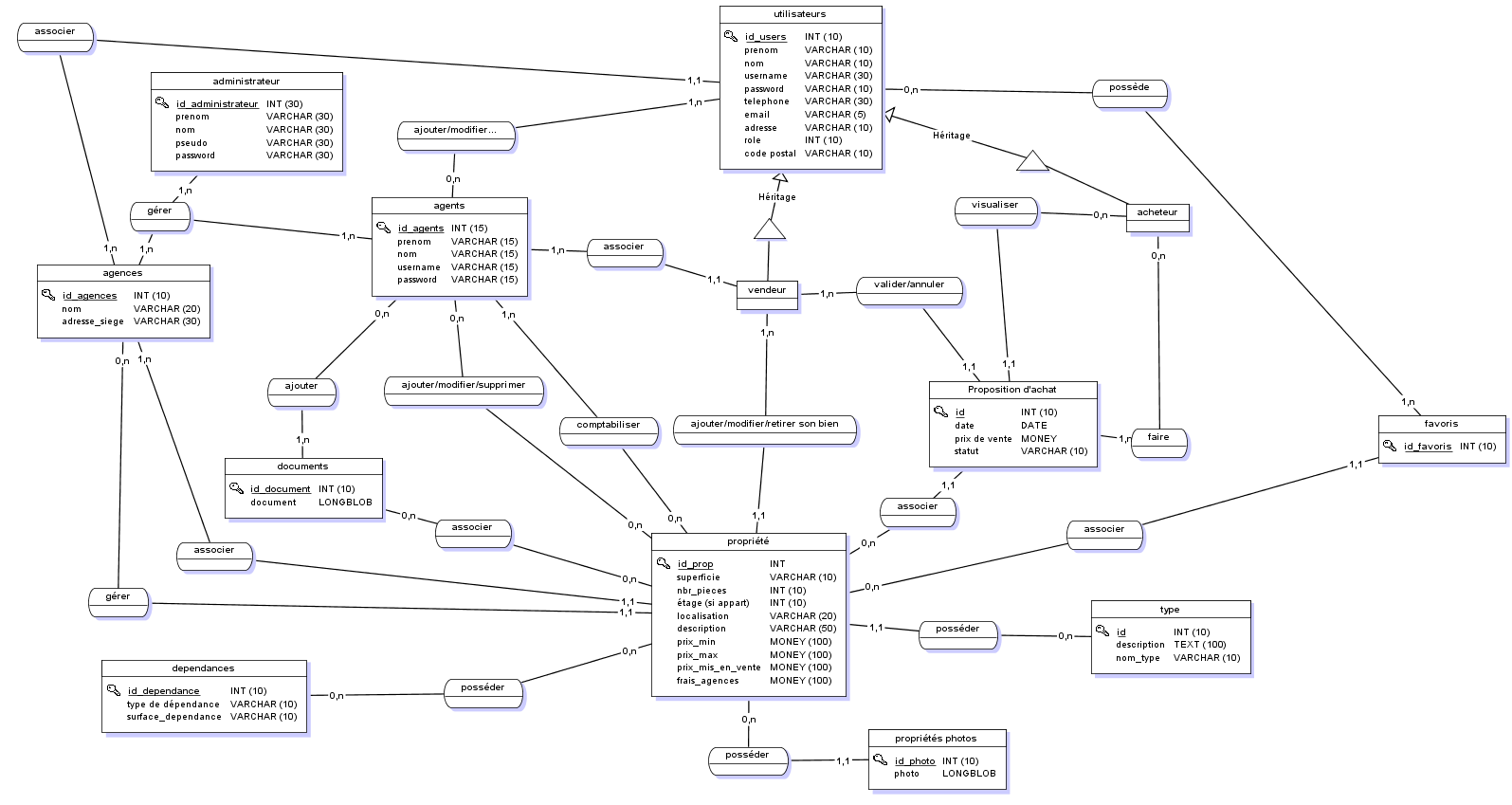
### Modifier informations de l’agent



Ici nous pouvons voir comment un agent peut modifier ses préférences (ses informations de connexion personnel).

# Conception de la base de données

La figure suivante représente le Modèle Conceptuel de données (MCD) :



# Modèle logique de données

Le MLD représente le MCD sous forme de texte, les clés primaires sont soulignés et les clés étrangères sont en gras :

Propriété (id\_prop, Superficie, nbr\_pieces, étages, adresse, prix\_min, prix\_max, frais\_agences, Status, **#id\_users, #id\_agences, #id\_type, #id\_dependances**) ;

Type (id\_type, description, nom\_type) ;

Photos (id\_photo, photo, **#id\_prop**) ;

Dépendances (id\_dependances, type\_dependance, surface\_dependance) ;

Agence (id\_agences, nom, adresse\_siege) ;

Agents (id\_agents, prenom, nom, username, password) ;

Administrateur (id\_admin , prenom, nom, pseudo, password) ;

Users (id\_users, nom, prenom, username, password, telephone, email, adresse, role, code\_postal **#id\_agences, #id\_agents**) ;

Favoris (id\_favoris, **#id\_prop, #id\_users**) ;

Proposition d’achats (id\_proposition, date, prix\_vente, statut, **#id\_users, #id\_prop**) ;

Documents (id\_document, document) ;

# Environnement de travail et technologies utilisées :

Dans cette partie, nous vous présentons les outils qui nous ont permis de réaliser le logiciel :

* JMerise
  + Avec cet outil nous avons conceptualisé le MCD (Modèle Conceptuel de Données) de notre base de données.
* Adobe XD
  + Avec cet outil nous avons conceptualisé les wireframes du logiciel.
* Eclipse
  + Avec cet outil nous avons réalisé le code du logiciel.
  + Nous avons également utilisé la bibliothèque étrangère MySQL Connector/J 8.0.20 pour lier notre code avec notre base de données.
* WampServer
  + Avec cet outil nous avons gérer notre base de données.
* GitHub
  + Avec cet outil de versionning nous avons pu sauvegarder et partager notre code.

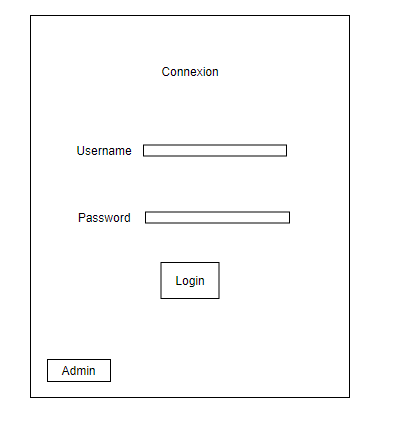
# Langage utilisé :

Pour la création du logiciel Stephi Place Software nous avons utilisé le langage **Java**.

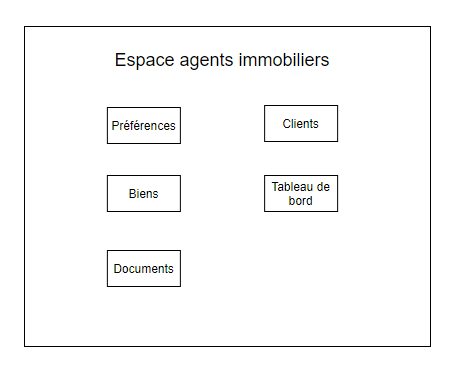
# Wireframe :

Dans cette partie nous allons vous présenter les wireframes de Stephi Place Software.

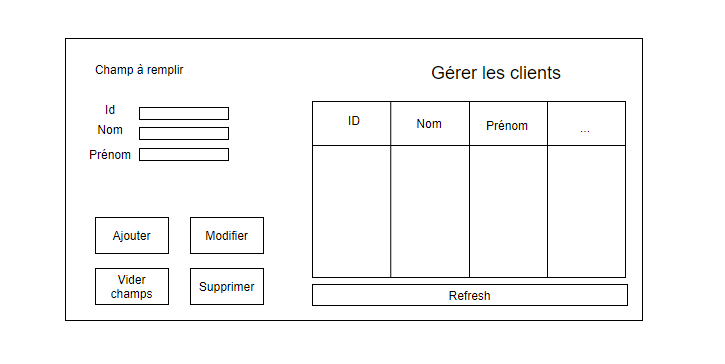
## Connexion agent immobilier



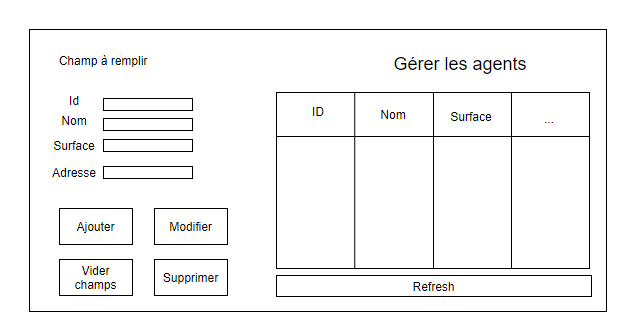
## Espace agents



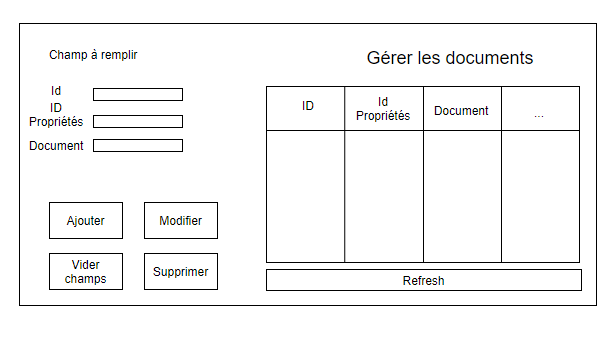
## Gérer les clients



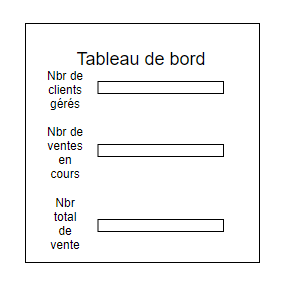
## Gérer les biens



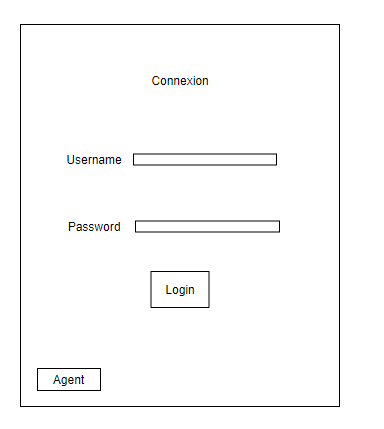
## Gestion des documents



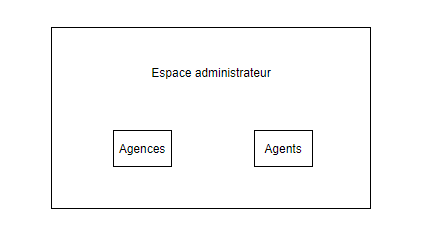
## Tableau de bord



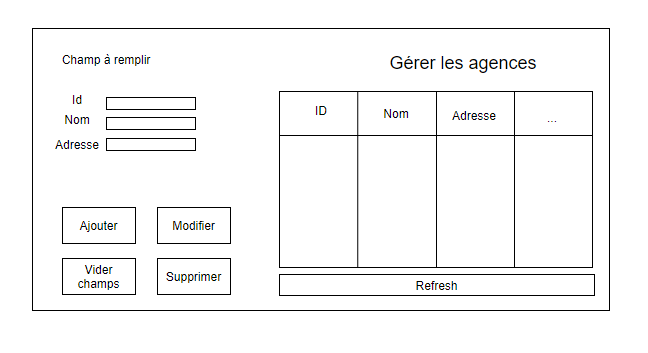
## Connexion administrateur



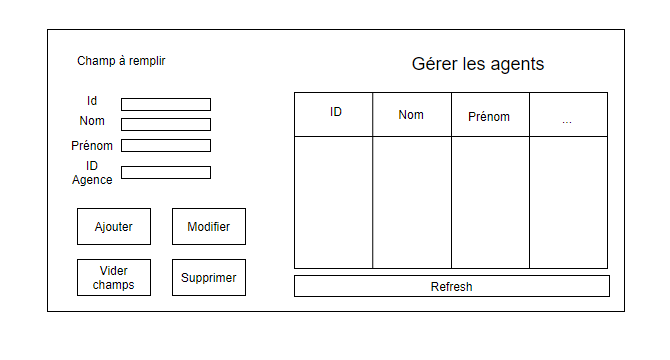
## Espace administrateur



## Gérer les agences



## Gérer les agents



# Annexes

## Notation UML

|  |  |
| --- | --- |
| **Notion UML** | **Description** |
|  | **Acteur :**  Rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système étudié. Un acteur participe à au moins un cas d’utilisation. |
|  | **Cas d’utilisation (use case)**  Ensemble de séquences d’actions réalisées par le système produisant un résultat observable intéressant pour un acteur particulier. Collection de scénarios reliés par un objectif utilisateur commun. |
|  | **Périmètre du Système**  Il est habituel pour afficher des cas d'utilisation comme étant à l'intérieur du système et les acteurs comme étant en dehors du système. |
|  | **Association**  utilisée dans ce type de diagramme pour relier les acteurs et les cas d’utilisation par une relation qui signifie simplement « participe à ». |
|  | **Extension**  le cas d’utilisation de base en incorpore implicitement un autre, de façon optionnelle, à un endroit spécifié indirectement dans celui qui procède à l’extension |
|  | **Inclusion**  le cas d’utilisation de base en incorpore explicitement un autre, de façon obligatoire, à un endroit spécifié dans ses enchaînements. |
|  | **Généralisation**  les cas d’utilisation descendants héritent de la description de leur parent commun. Chacun d’entre eux peut néanmoins comprendre des relations spécifiques supplémentaires avec d’autres acteurs ou cas d’utilisation.. |

**Tableau 6‑1 : Notation UML : Diagramme de cas d'utilisation**

|  |  |
| --- | --- |
| **Notion UML** | **Description** |
|  | **Classe**  Description abstraite d’un ensemble d’objets qui partagent les mêmes propriétés et comportements  **Attribut**  Donnée déclarée au niveau d’une classe, éventuellement typée, à laquelle chacun des objets de cette classe donne une valeur. Un attribut peut posséder une multiplicité et une valeur initiale. Un attribut dérivé (« / ») est un attribut dont la valeur peut être déduite d’autres informations disponibles dans le modèle.  **Opération ou Méthode :**  Élément de comportement des objets, défini de manière globale dans leur classe. Une opération peut déclarer des paramètres ainsi qu’un type de retour. |
|  | **Association**  Relation sémantique durable entre deux classes, qui décrit un ensemble de liens entre instances. Une association est bidirectionnelle par défaut, sauf si l’on restreint sa navigabilité en ajoutant une flèche.  **Rôle**  nom donné à une extrémité d’une association ; par extension, manière dont les instances d’une classe voient les instances d’une autre classe au travers d’une association.  **Multiplicité**  le nombre d’objets (min, max) qui peuvent participer à une relation avec un autre objet dans le cadre d’une association. Multiplicités fréquentes :   * + 0..1 = optionnel (mais pas multiple)   + 1 = exactement 1   + 0..\* = \* = quelconque   + 1..\* = au moins 1 |
|  | **Agrégation**  cas particulier d’association non symétrique exprimant une relation de contenance. |
|  | **Composition**  forme forte d’agrégation, dans laquelle les parties ne peuvent appartenir à plusieurs agrégats et où le cycle de vie des parties est subordonné à celui de l’agrégat. |
|  | **Super-classe**  Classe générale reliée à d’autres classes plus spécialisées (sous-classes) par une relation de généralisation.  **Généralisation**  Relation entre « classifieurs » où les descendants héritent des propriétés de leur parent commun. Ils peuvent néanmoins comprendre chacun des propriétés spécifiques supplémentaires, mais aussi modifier les comportements hérités. |
|  | **Classe d’association**  Association promue au rang de classe. Elle possède tout à la fois les caractéristiques d’une association et celles d’une classe et peut donc porter des attributs qui prennent des valeurs pour chaque lien entre objets. |
|  | **Qualifieur (ou qualificatif)**  Attribut qui permet de « partitionner » l’ensemble des objets en relation avec un objet donné dans le cadre d’une association multiple. |
|  | **Dépendance**  Relation sémantique entre deux éléments, dans laquelle la modification d’un des éléments peut affecter la sémantique de l’autre élément |

**Tableau 6‑2 : Notation UML : Diagramme de classes**

|  |  |
| --- | --- |
| **Notion UML** | **Description** |
|  | **Ligne de vie**  représentation de l’existence d’un élément participant dans un diagramme de séquence. Cela peut être un acteur ou le système en modélisation d’exigences, des objets logiciels en conception préliminaire ou conception détaillée. |
|  | **Message**  Élément de communication unidirectionnel entre objets qui déclenche une activité dans l’objet destinataire. La réception d’un message provoque un événement dans l’objet récepteur. La flèche pointillée  Représente un retour au sens UML. Cela signifie que le message en question est le résultat direct du message précédent. |
|  | **Occurrence d’interaction**  une interaction peut faire référence explicitement à une autre interaction grâce à un cadre avec le mot-clé ref et indiquant le nom de l’autre interaction.  UML 2 a ajouté une nouvelle notation très utile : les cadres d’interaction. Chaque cadre possède un opérateur et peut être divisé en fragments. Les principaux opérateurs sont :   * loop : boucle. Le fragment peut s’exécuter plusieurs fois, et la condition de garde explicite l’itération. * opt : optionnel. Le fragment ne s’exécute que si la condition fournie est vraie. * alt : fragments alternatifs. Seul le fragment possédant la condition vraie s’exécutera. |

**Tableau 6‑3 : Notation UML : Diagramme de séquence**

|  |  |
| --- | --- |
| **Notion UML** | **Description** |
|  | **Package (ou paquetage)**  mécanisme général de regroupement d’éléments tels que classes, interfaces, mais aussi acteurs, cas d’utilisation, etc.  Les packages peuvent être imbriqués dans d’autres packages. |
|  | **Importation**  relation de dépendance entre packages qui rend visibles  les éléments publics de l’un des packages au sein d’un autre |

**Tableau 6‑4 : Notation UML : Diagramme de package**