RÉPUBLIQUE DU BÉNIN

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY CALAVI (UAC)

ÉCOLE NATIONALE D'ÉCONOMIE APPLIQUÉE ET DE MANAGEMENT (ENEAM)

MÉMOIRE DE FIN DE FORMATION POUR L'OBTENTION DU DIPLÔME DE TECHNICIEN SUPÉRIEUR (DTS)

Filière : Informatique de Gestion (IG)

Spécialité : Analyse Informatique et Programmation (AIP)

THÈME

Mise en place d'une application mobile de recherche de logements.

Réalisé par : PADONOU Dieu-Donné et TOSSOU Cyriaque

Sous la direction de :

Maître de stage :
M. Fabrice KIKI
Développeur mobile à acumen

Maître de mémoire :
Dr Maurice Comlan
Enseignant à l'ENEAM

ANNÉE ACADÉMIQUE : 2018 - 2019

DÉDICACES

À vous nos Parents, voici peut-être un pas de plus vers votre espoir !

REMERCIEMENTS

Ce travail est le fruit des efforts conjugués de bienveillantes personnes à qui nous devons toute notre reconnaissance et toute notre gratitude. Nos remerciements vont particulièrement à l'endroit de :

- Prof Rosaline D. WOROU HOUNDEKON, Directrice de l'ENEAM ;
- Prof Théophile K. DAGBA, directeur adjoint, chargé des affaires académiques de l'ENEAM ;
- Dr Maurice COMLAN, notre encadreur qui, malgré ses multiples occupations, a accepté de diriger ce travail et y a accordé une attention particulière ;
- M. Luckmann GNAGOLI et M. Fabrice KIKI, nos maitres de stage ;
- Nos enseignants, qui ont su nous inculquer le goût du travail bien fait ;
- Enfin, tous ceux qui de près ou de loin ont contribué à la réalisation ce travail.

RÉSUME

Dans ce mémoire nous réalisons une application mobile, permettant de rechercher tous types de logements pour une location ou une réservation. Cette application fait intervenir le propriétaire et le client, autour d'une même perspective, celle d'augmenter la visibilité des propriétaires sur le marché de l'immobilier et de réduire le temps consacrer à la recherche d'un logement au strict minimum. L'application permet également aux clients de laisser des annonces pour être informer de toutes publications de logements répondant à des caractéristiques précises. Le client dispose donc d'un large éventail de publications semblables et il lui est aussi permis de noter et de commenter une publication, ce qui pourrait susciter chez le propriétaire la nécessité d'améliorer la qualité de ses logements afin de répondre aux attentes du client.

D'une part, ce travail comprend tout d'abord une analyse détaillée des différentes plateformes les plus utilisées dans le monde ayant pour objectif la recherche de logement. On y parle des particularités de chacune, ce qu'elles représentent et comment elles fonctionnent. D'autre part, le document transcrit une étude du projet qui explique l'objectif de la plateforme et qui définit les buts à atteindre. Par la suite, on trouve un petit descriptif des outils qui nous ont aidés dans l'élaboration du projet, suivis par les multiples phases de développement qui ont permis au projet d'exister.

Mots clés : logements, Hohaya, réservation, recherche.

ABSTRACT

In this thesis we realize a mobile application, allowing to search or reserve all types of accommodation. This application involves the owner and the client, around the same perspective, that of increasing the visibility of owners on the real estate market and reducing the time spent looking for accommodation to the bare minimum. The application also allows customers to leave announcements to be informed of all housing publications meeting specific characteristics. The client therefore has a wide range of similar publications and is also allowed to rate and comment on a publication, which could prompt the owner to improve the quality of his accommodation in order to meet client expectations.

On the one hand, this work firstly includes a detailed analysis of the various platforms most used in the world with the objective of finding accommodation. We talk about the particularities of each, what they represent and how they work. On the other hand, the document transcribes a study of the project which explains the objective of the platform and which defines the goals to be achieved. Thereafter, we find a short description of the tools that helped us in the development of the project, followed by the multiple development phases that allowed the project to exist.

Keywords: accommodation, Hohaya, reservation, search.

Liste des tableaux

Liste des Figures

Liste des sigles et abréviations

- **AOT**: Ahead Of Time.
- **API** : Application Programming Interface.
- **BD** : Base de données.
- **ENEAM** : École Nationale d'Économie Appliquée et de Management.
- **IDE** : Integrated Development Interface.
- **HP** : Hewlett Packard.
- **JIT** : Just In Time.
- **JSON** : JavaScript Object Notation.
- **JWT** : Json Web Token.
- **REST** : REpresentational State Transfer.
- **SGBD** : Système de Gestion de Base de Données.
- **SQL** : Structured Query Language.
- **UML** : Unified Modeling Language.

TABLE DES MATIÈRES

INTRODUCTION

Se loger est une nécessité pour tout individu. Mais à défaut de se construire une demeure, opter pour une location reste la meilleure solution. Au Bénin une grande partie de la population vit dans des maisons louées surtout dans les grandes villes comme Cotonou, Abomey Calavi et autres, en raison de la concentration des grandes institutions et des universités du pays dans ces villes. Il existe donc une manière courante et acceptée de tous pour se trouver un logement à des fins de location ou de réservation.

Le constat est que de nos jours les recherches effectuées dans l'optique de trouver un logement convenable sont très chronophages et nécessitent également beaucoup de ressources financières. Cela est dû est due à plusieurs facteurs dont le principal est la vétusté des procédés hasardeux mis en œuvre pour effectuer ces recherches à un moment où le numérique se répand dans nos vies et dans tous les secteurs d'activité. Par ailleurs, il est pratiquement impossible pour un locataire de partager ses impressions sur un logement et son expérience avec un potentiel locataire en dépit de l'avantage certain qu'offre une telle possibilité au futur locataire qui disposera donc de données empiriques pour évaluer le logement. Cette lacune profite à certains propriétaires qui remettent sur le marché des logements présentant des problèmes déjà souligner par un locataire précédent sans les avoir corrigés.

Ainsi vient la nécessité de développer une solution permettant de régler l'intégralité de ces problèmes, pour satisfaire au mieux les clients et améliorer par la même le rendement du propriétaire tout en tenant compte des réalités du Bénin. C'est dans cette optique que s'inscrit notre stage pratique de trois (3) mois en programmation au sein de l'entreprise Acumen où nous avons développé une solution applicative sur la base du thème « Mise en place d'une application mobile de recherche de logements ».

Le présent mémoire est structuré en trois (3) chapitres et expose le travail d'analyse et d'implémentation pour la mise en place du système. Le premier chapitre aborde la présentation du contexte d'étude à travers la présentation de la structure d'accueil et la présentation du thème. Le deuxième chapitre expose l'analyse et la spécification des besoins effectués pour ledit système. Enfin le troisième chapitre présente l'aspect sécuritaire du système, les outils utilisés pour sa réalisation ainsi que quelques interfaces issues de l'implémentation du système.

Chapitre I : Présentation du contexte de l'étude

1.1. Présentation de Acumen Network

1.1.1 Historique et mission

Acumen Network est une entreprise franco-béninoise créée en 2016. Elle fournit à ses clients un soutien holistique en vue de les accompagner dans leurs ambitions de transformations technologiques et digitales.

De façon plus spécifique, l'entreprise met à la disposition de ses clients son expertise dans les domaines du business et de la mise en œuvre de projets IT. Elle propose également des formations et des séances de coachings.

1.1.2 Organigramme

Le fonctionnement d'Acumen network est organisé selon la hiérarchie décrite par l'organigramme ci-dessous.

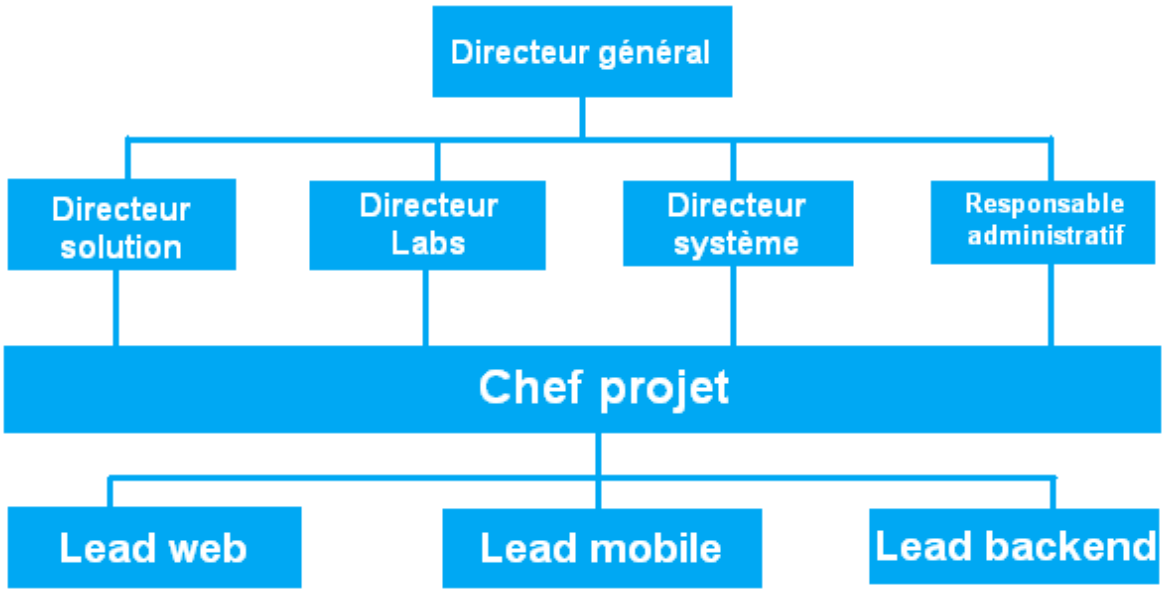

Figure 1: Organigramme de Acumen network

1.1.3 Déroulement du stage

Notre stage à Acumen a débuté le 27 octobre 2019 et a pris fin le 27 janvier 2020. Durant cette période, hormis le développement de notre application, nous avons eu l'opportunité de participer à certains travaux au sein de l'entreprise en réalisant les tâches qui nous étaient confiées. Nous avons aidé à peupler la base de données utilisée pour tester la plateforme digitale Agrosfer. Nous avons également été formé à l'utilisation de quelques technologies et outils tels que le Framework Java Spring pour la création des microservices et des applications web, les technologies Java et Flutter pour le développement d'applications mobiles et iOS, le Framework frontend Angular pour le développement web et le logiciel de design (UI/UX design) Figma pour la réalisation des interfaces utilisateurs.

1.1.4 Ressources matérielles et logiciels

Pour mener à bien ses activités et maintenir une productivité optimale, Acumen Network dispose ses ressources matérielles suivantes :

- quatre (04) serveurs répartis comme suit :
- neuf (09) ordinateurs portables de marque HP ;
- une imprimante laser HP ;
- un routeur isocèle ;
- une photocopieuse ;
- un projecteur et
- une télévision Samsung (smart TV).

1.1.5 Problèmes rencontrés et proposition de solution

Pendant notre stage nous avons été confrontés à un problème majeur qui est celui de la connexion internet qui n'était pas très stable provoquant le crash de certains tests et des difficultés à télécharger des dépendances ou tout autre outil nécessaire au bon déroulement de notre travail. De plus cela rendait impossible l'utilisation de certains outils comme Figma qui nécessite constamment la connexion.

L'instabilité de la connexion étant due aux retards de paiement des factures d'électricité et au fournisseur d'accès internet de la structure, nous proposons comme solution de :

Payer une recharge supplémentaire pour ne pas être surpris par l'épuisement de la recharge en cours ;

Opter pour un fournisseur d'accès internet plus fiable et offrant un meilleur service après-vente.

1.2. Étude préliminaire

1.2.1 Présentation de l'existant

Les opérations de recherche, de location et de réservation de logement sont de plus en plus récurrentes. Sollicités par divers clients (Apprenants, Artisans, Fonctionnaires, toutes personnes en besoin de logement étranger ou non), les services de logements se font de maintes manières.

D'abord, de bouche à oreille et passant de maison en maison, certains locataires parviennent à identifier les logements disponibles pour location ou réservation.

En l'état actuel des choses, les clients en recherche de logement doivent nécessairement se rapprocher des agences immobilières de la place ou s'offrir les services ''démarcheurs'', qui officient comme des agents immobiliers indépendants. Ces démarcheurs disposent au préalable des contacts de propriétaires de logement mis en location selon leurs types et les zones. Pour rendre ce service, les démarcheurs perçoivent des frais de déplacement pour chaque nouveau logement visité indépendamment de la satisfaction du client et un mois de loyer du logement lorsque celui-ci a été accepté par le client.

Outre cette manière de procéder, les clients internationaux peuvent se tourner vers les plateformes d'e-commerce au monde dans le secteur du voyage. Ces plateformes, à l'instar de Booking, Abritel, Expedia et bien d'autres encore, proposent, entre autres services, des logements dans plus d'une cinquantaine de pays.

1.2.2 Critique de l'existant

La procédure de bouche à oreille et de passage de maison en maison est très peu efficace et parfois hasardeuse. Elle nécessite énormément de travail avec l'incertitude de parvenir à un résultat concluant obligeant le client à avoir une parfaite maitrise des zones.

Recourir aux services d'un démarcheur peut se révéler très fastidieux et onéreux pour le client en besoin de logement, car il se trouve contraint d'assurer le déplacement du démarcheur pour tous les logements qu'ils auront à visiter faisant le tour de maison en maison jusqu'à trouver un logement qui lui convient. Ensuite, une fois le logement approprié trouvé, s'ils en trouvent, le client doit verser une somme équivalant à un mois de loyer du logement choisi, cette somme étant considérée comme frais du service rendu, il est indépendant des frais de location à verser au propriétaire.

Les services offerts par les plateformes de booking internationaux sont axés sur les voyages à travers le monde. En raison de leur clientèle composée majoritairement de particuliers devant visiter ou séjourner dans un autre pays, les logements proposés sont principalement les hôtels et les résidences de haut standing. Elles ne gèrent donc pas les locations de petite taille à l'intérieur d'un pays. De plus les moyens de paiement dont disposent les clients ne se limitent qu'à ceux employés de façon classique dans les transactions monétaires en ligne et ne prennent donc pas en compte les moyens moins connus, mais bien plus utilisés au Bénin.

1.2.3 Approche de solution : une application mobile de recherche et de réservation de logement.

Dans le but de simplifier les opérations de recherche et de réservation de logements, nous proposons une solution qui rassemble les acteurs impliqués dans ce processus. Il s'agit d'une application mobile nommée Hohaya, qui met à la disposition du propriétaire un espace lui permettant d'aller à la rencontre de ses locataires en publiant ses logements après avoir souscrit à un abonnement dérisoire. Hohaya permet au client de retrouver un logement correspondant exactement à ces attentes au bout d'une recherche de quelques minutes tout au plus. Il permet également au client de laisser des annonces afin de recevoir des notifications dès qu'un logement présentant les caractéristiques énumérées dans l'annonce est disponible. De plus, contrairement aux services existants dans ce domaine sur internet, les publications couvrent un large panel de logements allant des maisons dans les zones les plus reculées aux hôtels les plus connus et les moyens de paiement mis à la disposition de l'utilisateur sont les plus répandus sur dans le pays en matière de transaction financière en ligne, en l'occurrence Mobile money et Flooz.

1.2.4 Objectif principal et objectifs spécifiques

2.2.2.1 Objectif principal

Ce projet vise principalement à offrir aux utilisateurs, une plateforme facile d'utilisation et adaptée aux réalités du Bénin pour la mise en location de logements et optimisée pour la recherche de logements à louer ou à réserver.

2.2.2.2 Objectifs spécifiques

Plus spécifiquement les objectifs à atteindre à travers la mise en place de cette solution sont :

- permettre à un client de trouver un logement qui lui correspond sans se déplacer ;
- permettre à un client de dépenser le moins possible pour trouver un logement ;
- permettre à un locataire de donner son appréciation sur son séjour dans un logement ;
- permettre aux propriétaires d'étendre leur clientèle cible à toute l'étendue du territoire national ;
- permettre aux propriétaires des occupants pour ses logements plus facilement et de façon rapide ;
- connaître les besoins des riverains en termes de types de logements ;

Afin d'atteindre les objectifs précédemment énumérés, les utilisateurs de Hohaya pourront principalement selon leurs attributions :

- ajouter des logements ;
- publier des logements ;
- commenter ou noter un logement ;
- contacter le propriétaire d'un logement ;
- rechercher un logement ;
- recevoir des notifications pertinentes de nouvelles publications ;
- publier des annonces.

Chapitre II : Modélisation et spécification des besoins

2.1. Spécification des besoins

2.1.1 Présentation des acteurs

Un acteur est une entité qui définit le rôle joué par un utilisateur ou par un système qui interagit avec le système modélisé **[1]**. Les acteurs présents dans notre système sont les suivants :

- le client

2.1.2 **le propriétaire**.

2.1.3 Spécifications fonctionnelles

Le système à mettre en place devra fournir certaines fonctionnalités. Il doit permettre :

- Au client :
  - de passer en revue les publications disponibles.
  - de rechercher un type de logement spécifique.
  - d'émettre une annonce.
  - de contacter un propriétaire pour un logement donné.
  - de réserver un logement pour une période déterminée.
  - d'apprécier un logement (noter ou ajouter des commentaires).
- Au propriétaire :
  - d'ajouter de nouveaux logements.
  - de publier ses logements.
  - d'apporter des modifications ou de supprimer un logement.
  - d'accepter une réservation.

2.1.4 Description fonctionnelle

Hohaya a pour but de faciliter les opérations de recherche ou de réservation de logements. Ainsi il retrace toutes les étapes nécessaires à la prise de contact avec un propriétaire ou à la

réservation effective d'un logement tout en occultant les tâches encombrantes et inutiles ou en simplifiant les tâches redondantes. Hohaya permettra donc :

- de créer des comptes utilisateur :

  Cette fonctionnalité permet à tout utilisateur de créer un compte permettant de l'identifier dans le système et de lui attribuer des privilèges selon qu'il est propriétaire ou client.

- d'ajouter un logement :

  Cette fonctionnalité permet aux propriétaires d'ajouter de nouveaux logements dans le système en fournissant des images et des caractéristiques propres à chaque nouveau logement.

- de publier un logement :

  Cette fonctionnalité permet à un propriétaire de rendre un de ses logements visible pour tous les utilisateurs du système quand les conditions de son abonnement le lui permettent.

- de faire une réservation :

  Cette fonctionnalité permet à un utilisateur de réserver un logement de son choix pour une période donnée.

- de rechercher un logement :

  Cette option permet à tout utilisateur d'effectuer une recherche pour trouver un logement présentant des caractéristiques très précises.

- de contacter un propriétaire :

  Cette option permet à un client d'obtenir les contacts d'un propriétaire pour un logement particulier.

- de répondre à une réservation :

  Cette fonctionnalité permet à un propriétaire d'accepter ou non la réservation d'un client.

- de publier des annonces :

  Cette fonctionnalité permet à un client de créer des annonces présentant les caractéristiques d'un logement susceptible de lui convenir et donc de recevoir des notifications dès qu'un logement correspondant à la plupart des critères énumérés est publié.

2.1.5 Spécifications non fonctionnelles

En plus des fonctionnalités citées ci-dessus, l'application doit répondre aux critères suivants, qui garantiront sa fiabilité et son utilisabilité :

LA SÉCURITÉ ET L'INTÉGRITÉ DES DONNES : En raison du caractère hautement sensible des données à traiter, l'accès à certaines fonctionnalités de l'application n'est permis qu'après un processus d'authentification visant à s'assurer de l'identité et des droits de l'utilisateur en cause. Les informations d'inscription et d'authentification doivent être

confidentielles. L'application doit garantir l'intégrité, la cohérence et la persistance des données.

LA RAPIDITÉ DE TRAITEMENT : Compte tenu de nombre élevé de traitements à effectuer à chaque instant il est impératif de veiller à réduire le délai d'exécution des différentes requêtes au minimum.

LA PERFORMANCE : Notre application doit être avant tout performante, c'est-à-dire à travers ses fonctionnalités, répondre à toutes les exigences des usagers d'une manière optimale.

LA CONVIVIALITÉ : Hohaya doit proposer des options faciles d'accès et d'utilisation. En effet, les interfaces utilisateurs doivent respecter les différentes normes ergonomiques en la matière et en matière d'expérience utilisateur.

2.2. Analyse orientée objet

Dans le cadre de l'analyse orientée objet du système, nous avons opté pour le langage de modélisation unifié, plus connu sous la désignation United Modeling Language (UML) en anglais.

UML 2.3 propose 14 diagrammes subdivisés en 2 grandes catégories : les diagrammes dynamiques ou comportementaux et les diagrammes statiques ou structurels. Dans ce travail d'analyse, nous ne présenterons que 5 d'entre ces diagrammes. Il s'agit des diagrammes de cas d'utilisation, de classe, de séquence, d'état transition et d'activité.

2.2.1 Le diagramme de cas d'utilisation

2.2.2.1 Identification des cas d'utilisation de Hohaya.

Il est nécessaire de définir, pour chacun des acteurs, les actions à mener (cas d'utilisation). Les divers acteurs intervenant dans le système ainsi que les actions qu'ils peuvent mener sont représentés dans le tableau ci-dessous.

Tableau 1 : Identification des acteurs et des cas d'utilisation de Hohaya

Acteurs	Cas d'utilisation
Client	- S'inscrire - Se connecter

	- Gérer son compte - Apprécier une publication - Rechercher un logement - Gérer ses annonces - Ajouter une réservation - Contacter un propriétaire
Propriétaire	- S'inscrire - Se connecter - Gérer son compte - Apprécier une publication - Rechercher un logement - Gérer ses annonces - Ajouter une réservation - Contacter un propriétaire - Gérer ses logements - S'abonner - Répondre à une réservation

2.2.2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation qui permet de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. **[2]**

La figure suivante illustre le diagramme de cas d'utilisation de Hohaya.

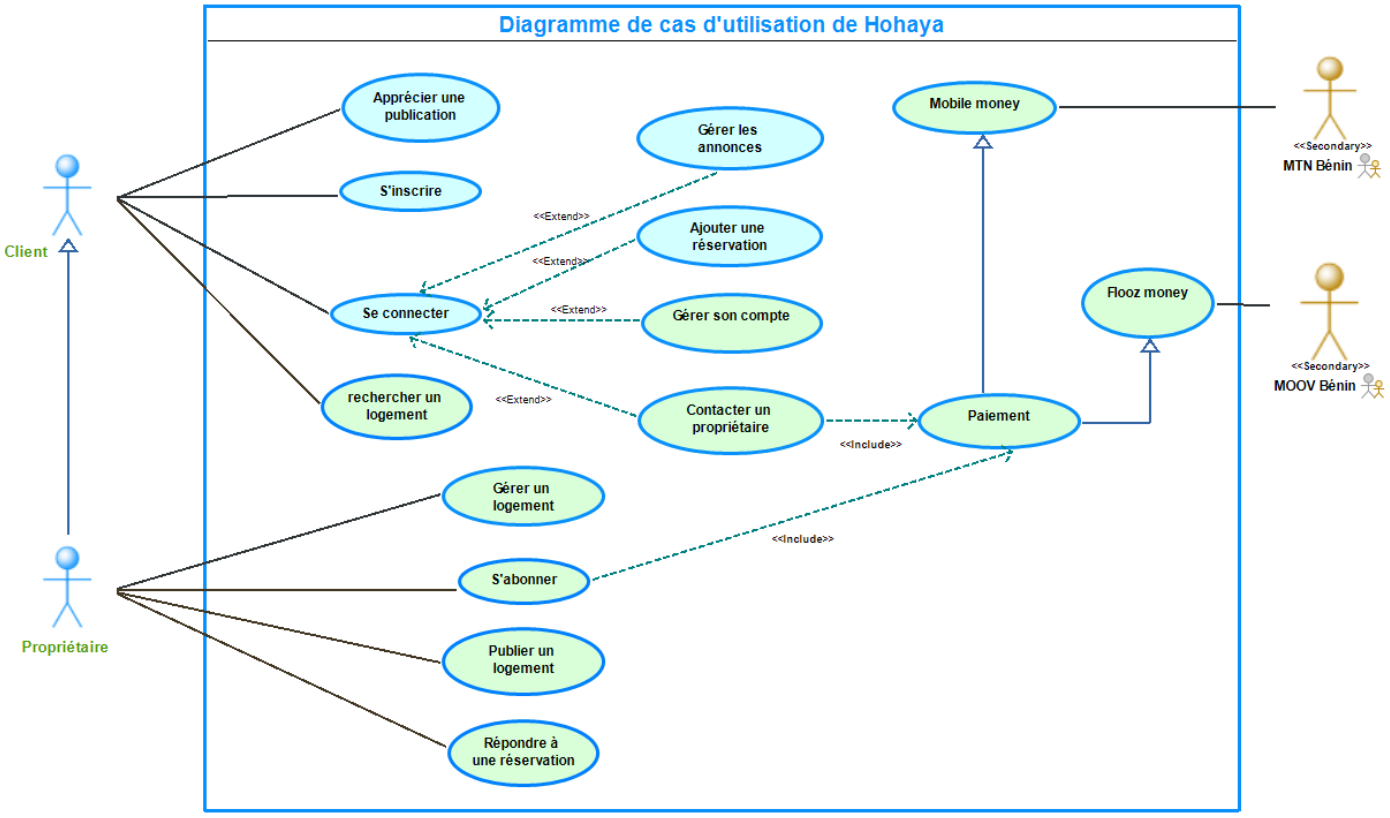

Figure 2 : Diagramme de cas d'utilisation de Hohaya

Suite à une analyse approfondie du système, nous avons élaboré le diagramme de cas d'utilisation ci-dessus qui résume aussi bien les interactions des différents acteurs avec le système que les interactions entre différents cas d'utilisation.

2.2.2.3 Description textuelle des cas d'utilisation

La description textuelle des cas d'utilisation nous permet d'établir de manière précise toutes les interactions qui auront lieu entre les acteurs et le système au cours de la réalisation d'un cas d'utilisation particulier.

Description textuelle du cas d'utilisation « **Contacter un propriétaire** »

Sommaire d'identification

Titre : Contacter un propriétaire

Résumé : Ce cas permet aux utilisateurs d'obtenir les contacts du propriétaire d'un logement donné

Acteurs : Client ou Propriétaire

Date de Création : 25/01/2020

Version : 1.0

Responsable : PADONOU Dieu-Donné et TOSSOU Cyriaque

Préconditions : Aucune

Description des Scénarii

- **Scénario nominal**
  1. L'acteur clique sur le bouton « 'Contacter le propriétaire »'.
  2. Le système vérifie que l'utilisateur est connecté.
  3. Le système affiche le formulaire de paiement.
  4. L'acteur renseigne les informations de paiement.
  5. Le système vérifie la validité des informations renseignées.
  6. Le système retire le montant prévu du compte de l'utilisateur.
  7. Le système affiche les contacts du propriétaire.
- **Scénario alternatif**

  A1 : Un (ou plusieurs) champ non renseigné(s) ou invalide(s)

  Le scénario A1 démarre au point 5 du scénario nominal.

  Le point 6 devient « Le système indique à l'acteur que certains champs sont requis ou invalides ».

  Le scénario reprend au point 3 du scénario nominal.

  A2 : L'utilisateur n'est pas connecté.

  Le scénario A2 démarre au point 2 du scénario nominal.

  Le point 3 devient « le système affiche la page de connexion ».

  Le scénario reprend au point 3 du scénario nominal.

Description textuelle du cas d'utilisation « **Publier un logement** »

Sommaire d'identification

Titre : Publier un logement

Résumé : Ce cas permet aux propriétaires de publier un logement déjà existant dans le système.

Acteurs : Propriétaire

Date de Création : 25/01/2020

Version : 1.0

Responsable : PADONOU Dieu-Donné et TOSSOU Cyriaque.

Préconditions : Acteur authentifié comme Propriétaire.

Description des Scénarii

Scénario nominal

1. Le propriétaire clique sur le bouton « Publier »'.
2. Le système vérifie que son abonnement est actif.
3. Le système vérifie que les conditions de son abonnement lui permettent de publier ce logement.
4. Le système crée une nouvelle instance de publication pour ce logement dans la base de données.
5. Le système affiche un message de succès.

Scénario alternatif

A1 : Le propriétaire n'a aucun abonnement actif.

Le scénario A1 démarre au point 2 du scénario nominal.

Le point 3 devient « Le système indique au propriétaire qu'il n'a aucun abonnement actif ».

Le point 4 devient « Le système déclenche l'exécution du cas d'utilisation : S'abonner ».

Le scénario reprend au point 4 du scénario nominal.

A2 : Les conditions de son abonnement ne lui permettent pas de publier ce logement.

Le scénario A2 démarre au point 3 du scénario nominal.

Le point 4 devient « Le système lui notifie que son abonnement ne prend pas en charge cette publication ».

Le cas d'utilisation prend fin.

Postconditions :

Le statut de la publication du logement passe de « 'inactif "' à ''actif ''.

Le logement est désormais visible pour tous les utilisateurs du système.

2.2.2 Classes et modèle relationnel

2.2.2.1 Règles de gestion

Les règles de gestion ci-dessous régissent le fonctionnement du système.

R1- Seul un propriétaire peut publier, modifier ou supprimer un logement.

R2- Un propriétaire peut faire tout ce que peut faire un client.

R3- Une appréciation est soit une note ou une note suivie d'un commentaire.

R4- Un utilisateur ne peut émettre qu'une seule appréciation pour un logement donné.

R5- Plusieurs logements peuvent avoir la même localisation.

R6- Un même locataire peut prendre contact avec un propriétaire pour le même logement plusieurs fois.

R7- Un logement n'a qu'une publication qui lui est relative.

R8- Un propriétaire peut ne pas souscrire à un abonnement.

R9- Seuls les propriétaires peuvent souscrire à un abonnement.

R10- Un propriétaire ne peut souscrire qu'à un seul abonnement à la fois.

R11- Un logement n'est plus disponible pour la réservation dès qu'un propriétaire accepte une réservation faite pour ce logement.

2.2.2.2 Diagramme de classes

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. **[3]**

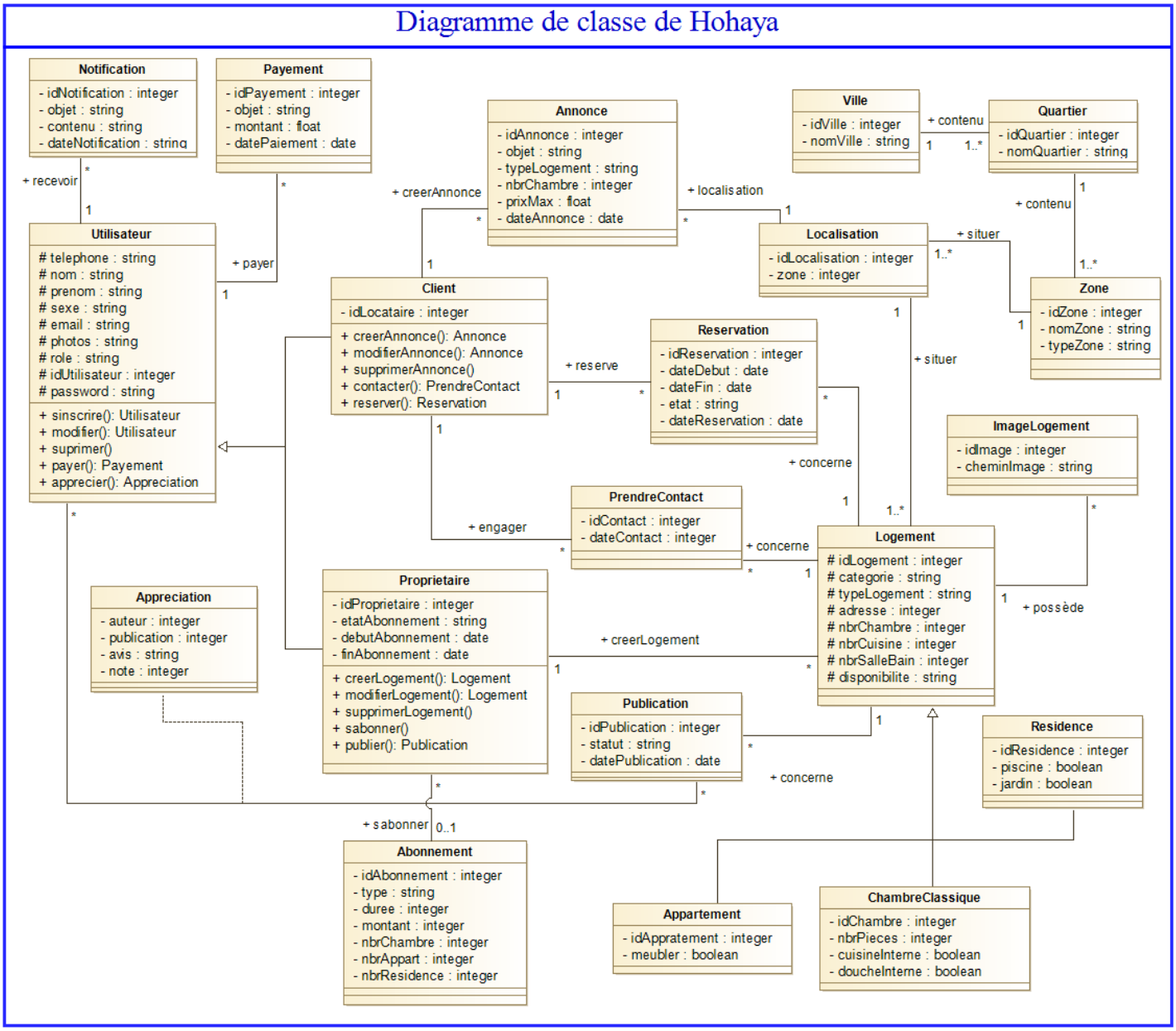

Figure 3 : Diagramme de classe de Hohaya

2.2.2.3 Modèle relationnel du système

Le modèle relationnel est une manière de modéliser les informations, et de les ordonner entre elles. Il permet d'avoir un aperçu des différentes classes qui constitueront la base de données du système et des attributs qui composeront chaque table. Le modèle relationnel en 3FN de notre système est le suivant :

- Locataire (idLoacataire, nom, prenom, telephone, password, sexe, email, photo, role).
- Proprietaire (idProprietaire, nom, prenom, telephone, password, sexe, email, photo, role, etatAbonnement, debutAbonnement, finAbonnement, #idAbonnement).
- Abonnement (idAbonnement, type, duree, montant, nbrChambre, nbrResidence, nbrAppart).
- Notification (idNotification, objet, contenu, dateNotification, # idUtilisateur).
- Paiement (idPaiement, objet, montant, datePaiement, # idUtilisateur).
- Annonce (idAnnonce, objet, typeLogement, nbrChambre, prixMax, dateAnnonce, # idUtilisateur, nbrCuisine, nbrSalleBain, #idLocalisation, #idClient).
- Reservation (idReservation, dateDebut, dateFin, etat, dateReservation, #idUtilisateur, #idLogement).
- Logement (idLogement, categorie, typeLogement, nbrChambre, nbrCuisine, nbrSalleBain, #idProprietaire, #idLocalisation).
- ChambreClassique (idChambre, nbrPiece, cuisineInterne, doucheInterne).
- Residence (idResidence, jardin, piscine).
- Appartement (idAppart, meubler).
- Appreciation (#idUtilisateur, #idPublication, avis, note, dateAppr).
- Publication (idPublication, statut, datePublication, idLogement).
- ImageLogement (idImage, cheminImage, #idLogement).
- PrendreContact (idContact, dateContact, #idUtilisateur, #idLogement).
- Localisation (idlocalisation, #idZone).
- Zone (idZone, nomZone, typeZone, #idQuartier).
- Quartier (idQuartier, nomQuartier, #idVille).
- Ville (idVille, nomVille).

2.2.3 Diagrammes de séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML. **[4]**

❖ Diagramme de séquence du cas « **publier un logement** ».

Le diagramme ci-dessous illustre le processus aboutissant à la publication d'un logement. Ainsi après avoir cliqué sur le bouton « **Publier** », le système vérifie que le propriétaire dispose

d'un abonnement en cours de validité auquel cas le système vérifie si l'abonnement du propriétaire lui permet de faire cette publication. Si oui, alors pour une première publication, une nouvelle instance de publication est créée en base pour ce logement et l'état de cette publication est mis à « **actif** ». Si l'abonnement du propriétaire ne couvre pas cette publication alors une notification est envoyée par le système pour en avertir le propriétaire et la publication n'est pas faite. Dans le cas où le propriétaire n'a pas du tout d'abonnement, il est redirigé vers l'interface de souscription à un nouvel abonnement.

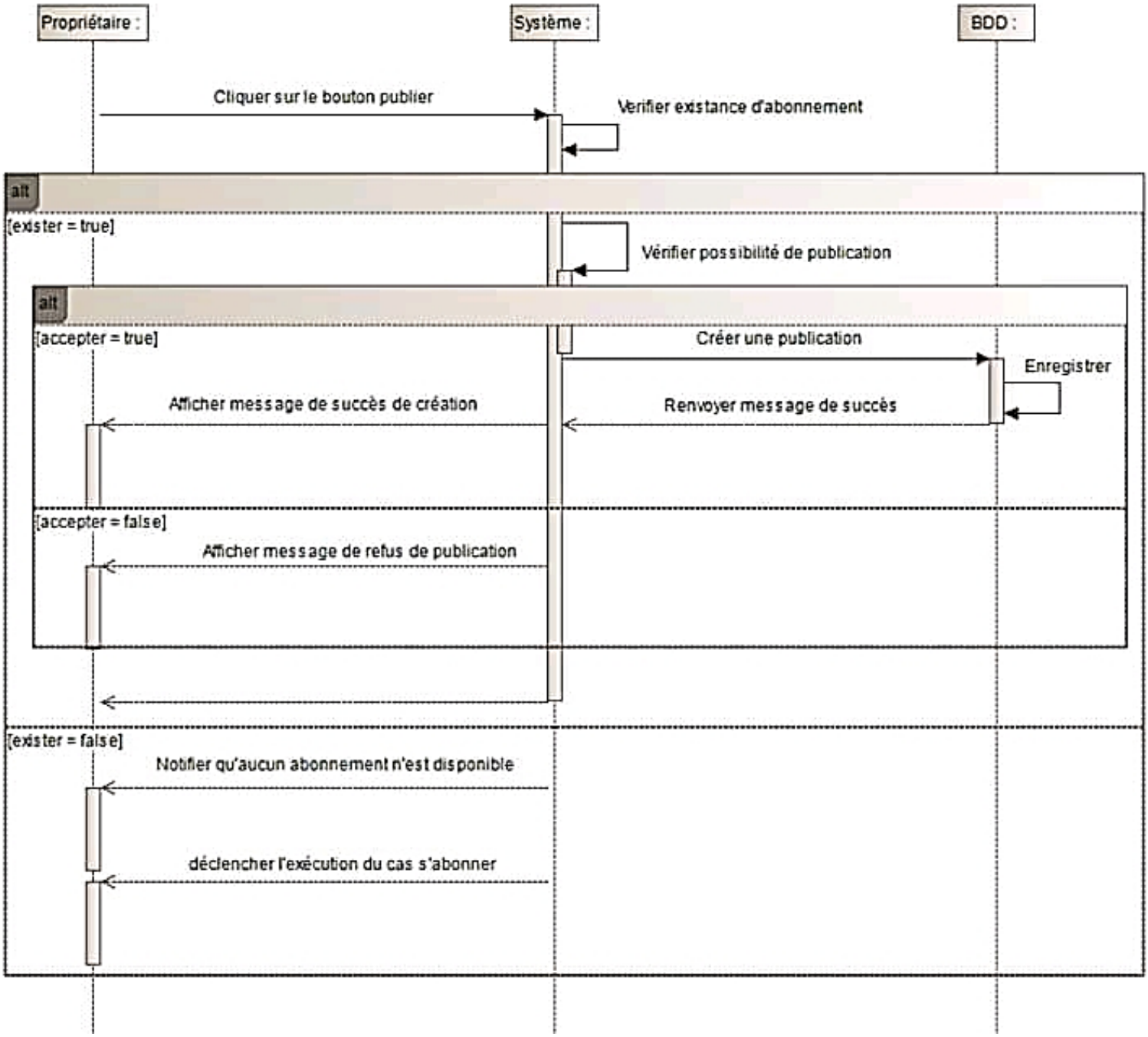

Figure 4: Diagramme de séquence du cas "Publier un logement"

❖ Diagramme de séquence du cas « **contacter un propriétaire** »

Le diagramme suivant illustre le fonctionnement du cas d'utilisation permettant à un client de contacter un propriétaire. Après avoir choisi l'option « contacter le propriétaire », pour un logement donné, le système vérifie si le l'utilisateur est connecté. Si ce n'est pas le cas, le système amène l'utilisateur à s'authentifier avant de continuer. Une fois l'utilisateur connecté, le système invite l'utilisateur à insérer les informations de son compte mobile money ou flooz pour le retrait d'une somme convenue. Après la saisie des informations, le système vérifie la validité des informations et lorsque celles-ci sont les bonnes la transaction est faite. Une nouvelle instance de prise contact est alors enregistrée en base de données et le système affiche

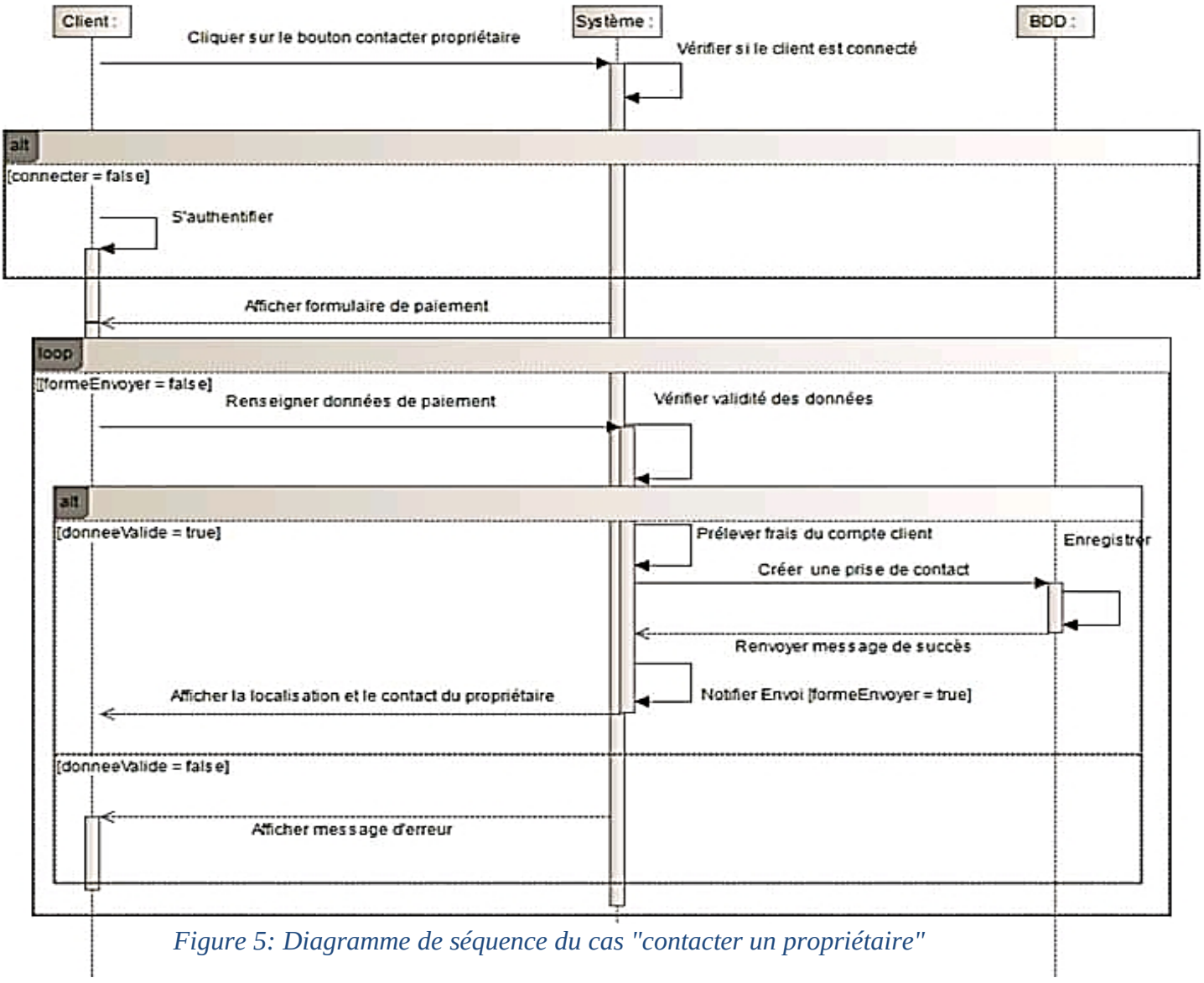

Figure 5: Diagramme de séquence du cas "contacter un propriétaire"

au client toutes les informations nécessaires au client pour une prise de contact physique avec le propriétaire.

2.2.4 Diagrammes d'état transition

Les diagrammes d'états-transitions d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode). Ils spécifient habituellement le comportement d'une instance de classeur (classe ou composant), mais parfois aussi le comportement interne d'autres éléments tels que les cas d'utilisation, les sous-systèmes, les méthodes. Le diagramme d'états-transitions est le seul diagramme, de la norme UML, à offrir une vision complète et non ambiguë de l'ensemble des comportements de l'élément auquel il est attaché **[5]**.

Voici illustrer ci-dessous quelques diagrammes d'état transition de Hohaya.

❖ Diagramme d'état transition du cas « Publier un logement »

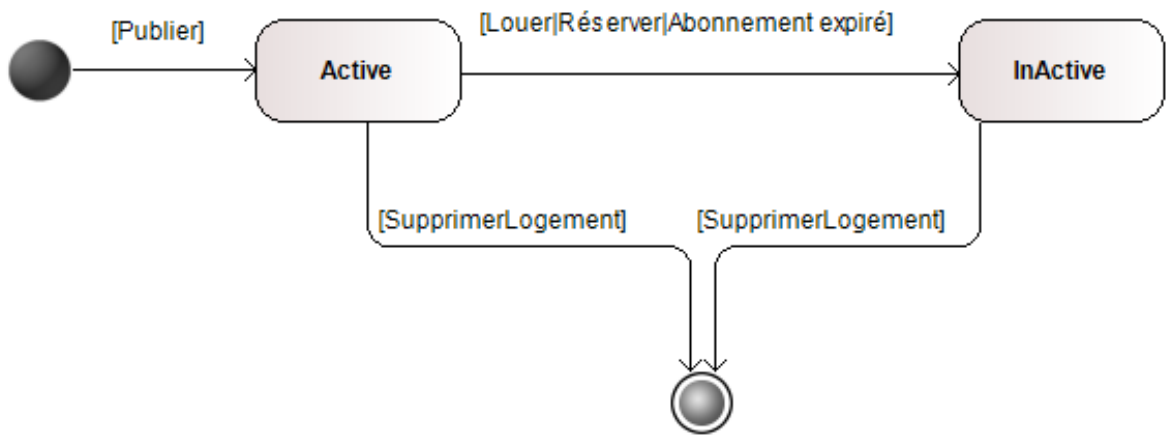

Figure 6 : Diagramme d'état transition de l'objet publication

❖ Diagramme d'état transition du cas « **Réserver un logement** »

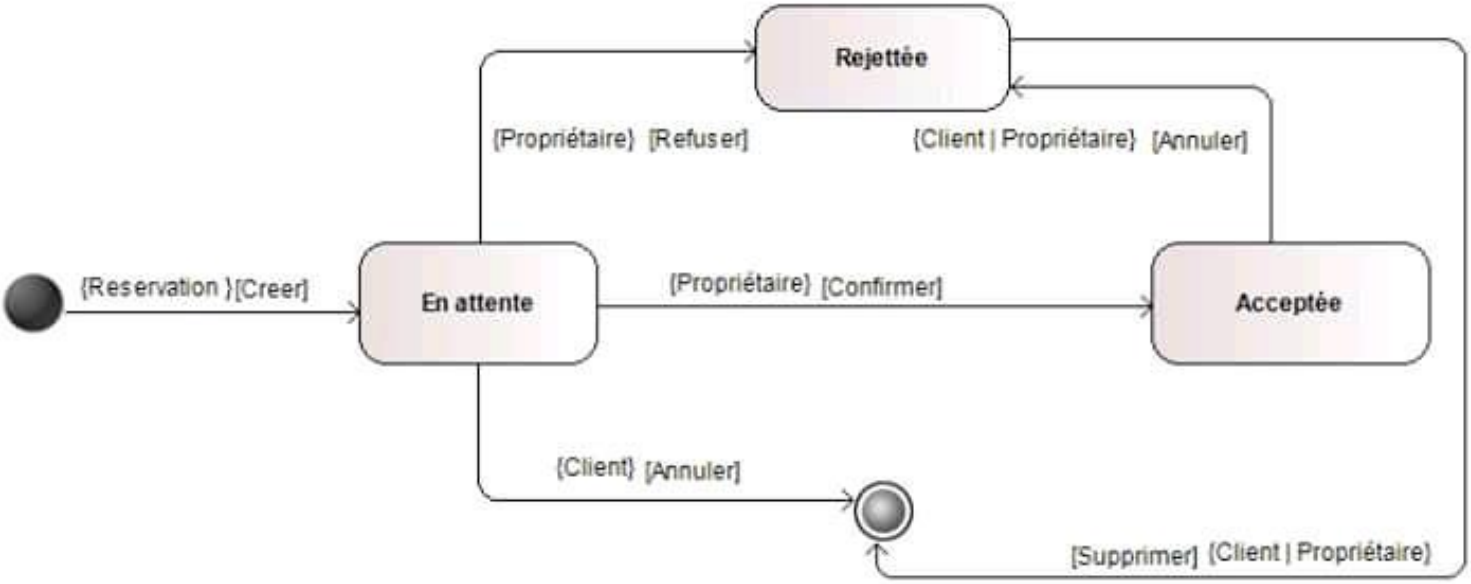

Figure 7: Diagramme d'état transition de l'objet réservation

2.2.5 Diagramme d'activité

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables (multithreads ou multiprocessus). Le diagramme d'activité est également utilisé pour décrire un flux de travail (workflow). Un diagramme d'activité permet de modéliser un processus interactif, global ou partiel pour un système donné **[6]**.

❖ Diagramme d'activité des processus de location ou de réservation de logement

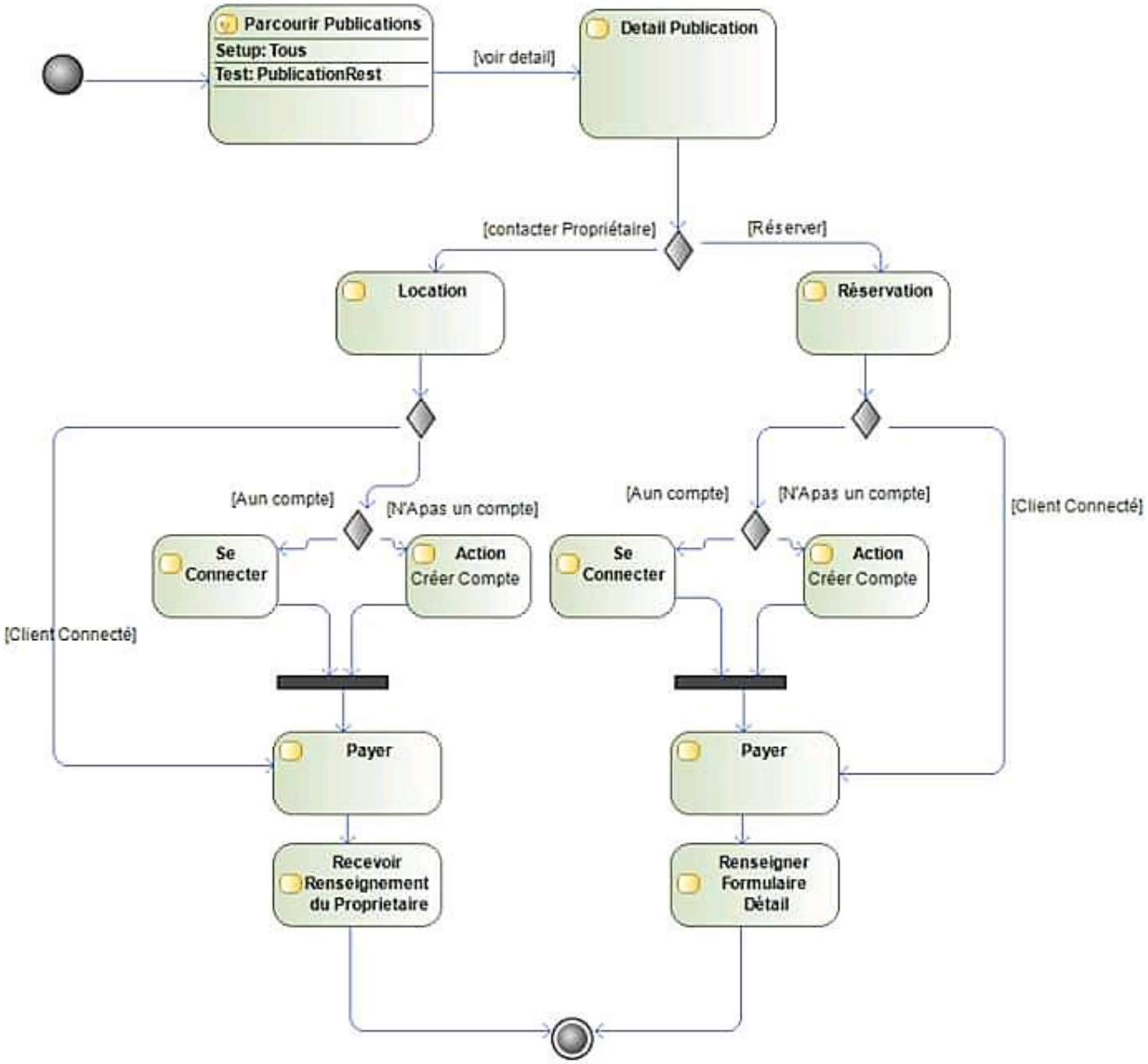

Figure 8: Diagramme d'activité des processus de location ou de réservation de logement

Chapitre III : Implémentation du système

3.1. Outils de développement utilisés

❖ Spring boot

Spring Boot est un Framework open source Java utilisé pour créer des microservices. Il est développé par l'équipe Pivotal et est utilisé pour construire des applications prêtes à ressort autonomes et de production. Très connu pour la facilité de déploiement et le minimum de configuration qu'il requiert, Spring boot nous a permis de développer des API Rest pour notre application.**[7]**

❖ Java

Java est un langage de programmation inspiré du langage C++, avec un modèle de programmation orienté objet. Il permet d'obtenir un code robuste et entièrement orienté objet. De plus les programmes java sont portables. En effet, un code Java peut être exécuté sur un serveur ou un client doté d'une machine virtuelle Java. Cette dernière traduit le code compilé en code exécutable sur le matériel informatique. Cela signifie que les différences entre les plateformes, comme la longueur des instructions, peuvent être reconnues et gérées en local au fil de l'exécution du programme. Il n'est donc plus nécessaire de créer des versions différentes du programme pour chaque plateforme.[8]

❖ Flutter

Flutter est un Framework développé par Google, le plus récent de tous. De ce fait, les ingénieurs ont pu observer les points forts et les faiblesses de chaque outil existant pour n'en extraire que la quintessence. En effet ce Framework est utilisé pour tout ce qui est interface utilisateur. Mais aujourd'hui Flutter se fait surtout connaître pour sa capacité à concevoir des applications natives multiplateforme pour Android et iOS (Windows/Mac/Linux sont également supportés). En raison du langage de programmation utilisé, à savoir **Dart,** le développement des applications avec Flutter est bien plus rapide.**[9]**

❖ Dart

Dart (initialement appelé Dash4) est un langage de programmation web développé par Google. Son but initial est de remplacer JavaScript pour devenir la nouvelle lingua franca du développement web, néanmoins la priorité actuelle des développeurs est que le code Dart puisse être converti en code JavaScript compatible avec tous les navigateurs modernes, ainsi que sur le développement d'application multi-plateforme. Dart peut aussi être utilisé pour la programmation côté serveur, ainsi que le développement d'applications mobiles (via l'API Flutter). Dart offre deux modes de fonctionnement. Le premier, nommé AOT (pour Ahead Of Time), permet de générer une application native pour chaque plateforme. Le deuxième mode de fonctionnement est dit JIT (Just-In-Time) et offre la fonctionnalité de Hot Reload lors des développements. L'idée du Hot Reload en Flutter est de corriger le problème du temps nécessaire entre chaque build en ne mettant plus que quelques millisecondes (voire secondes dans le pire des cas) entre chaque modification. Le développement de son application est alors bien plus rapide.[9] [10]

❖ SGBD (Système de Gestion de Base de Données)

Pour la sauvegarde des données des utilisateurs, il est nécessaire de mettre en place une base de données. Les outils nécessaires à la mise en place et à la gestion de cette base sont fournis par un SGBD. Dans le cadre de ce travail, nous avons opté pour le SGBD **PostgreSQL**.

- **PostgreSQL** est un système de gestion de base de données de modèle relationnel & objet open source et fait partie des plus utilisés dans le monde en raison des nombreux avantages qu'il offre en comparaison des autres SGBD. Au nombre de ces avantages, on peut énumérer ceux qui suivent : [11].
- PostgreSQL est plus fiable et l'intégrité des données y est plus performante ;
- PostgreSQL dispose d'un planificateur de requêtes sophistiqué et d'un optimiseur de requêtes ;
- La documentation de PostgreSQL est plus dense et complète et de façon générale le support est meilleur ;
- PostgreSQL est meilleur pour les requêtes avec sous-requêtes ;
- PostgreSQL propose une licence MIT qui est plus permissive que la licence GPL de MySQL.

❖ Postman

Postman fait partie des solutions pour appeler/tester une API Web. Il présente beaucoup de fonctionnalités avancées, faisant de lui un des outils les plus utilisés pour le test des API Web. L'interface graphique proposée est claire et permet une prise en main rapide de l'outil. Les requêtes construites et exécutées sont stockées dans un historique, leur permettant d'être rejouées facilement ultérieurement. Chaque requête peut être documentée au sein même de Postman, tout comme la collection elle-même. Le tout peut être partagé pour faciliter le travail collaboratif. De plus il propose des facilités pour la prise en charge de l'authentification, supportant ainsi les modes les plus communs (BASIC, DIGEST, OAuth 1 & 2).[12]

- InteliJ IDEA

InteliJ IDEA est l'IDE (Integrated Development Interface) que nous avons utilisée pour réaliser notre application. Plusieurs raisons justifient ce choix. En effet l'IDE est optimisée pour un démarrage rapide et propose des fonctionnalités avancées en termes de saisie automatique intelligente et d'autocomplétions de code. De plus il intègre un outil de gestion de bases de données SQL, un décompilateur Java par défaut et dispose d'une prise en charge de premier ordre pour des principaux Framework comme Spring ou encore AngularJS, tout ceci dans un environnement convivial et très ergonomique.[13]

3.2. Sécurité de l'application

En raison de l'importance et de la sensibilité des données traitées, certaines mesures ont été mises en place afin de garantir la sécurité de l'application. Ainsi :

- L'accès à l'application requiert le renseignement d'un identifiant et d'un mot de passe permettant d'identifier l'acteur ;
- Les mots de passe sont cryptés grâce à l'algorithme de chiffrement bcrypt avant d'être stockés dans la base de données ;
- Les fonctionnalités réservées aux utilisateurs ayant un certain privilège sont cachées aux autres utilisateurs de sorte qu'ils en ignorent même l'existence ;
- L'utilisation du JWT (Json Web Token) pour encrypter les données sensibles échangées entre le client et le serveur ;

3.3. Travaux réalisés

Les captures suivantes présentent les principales interfaces de la version bêta de Hohaya.

❖ **Interface d'accueil de l'application**

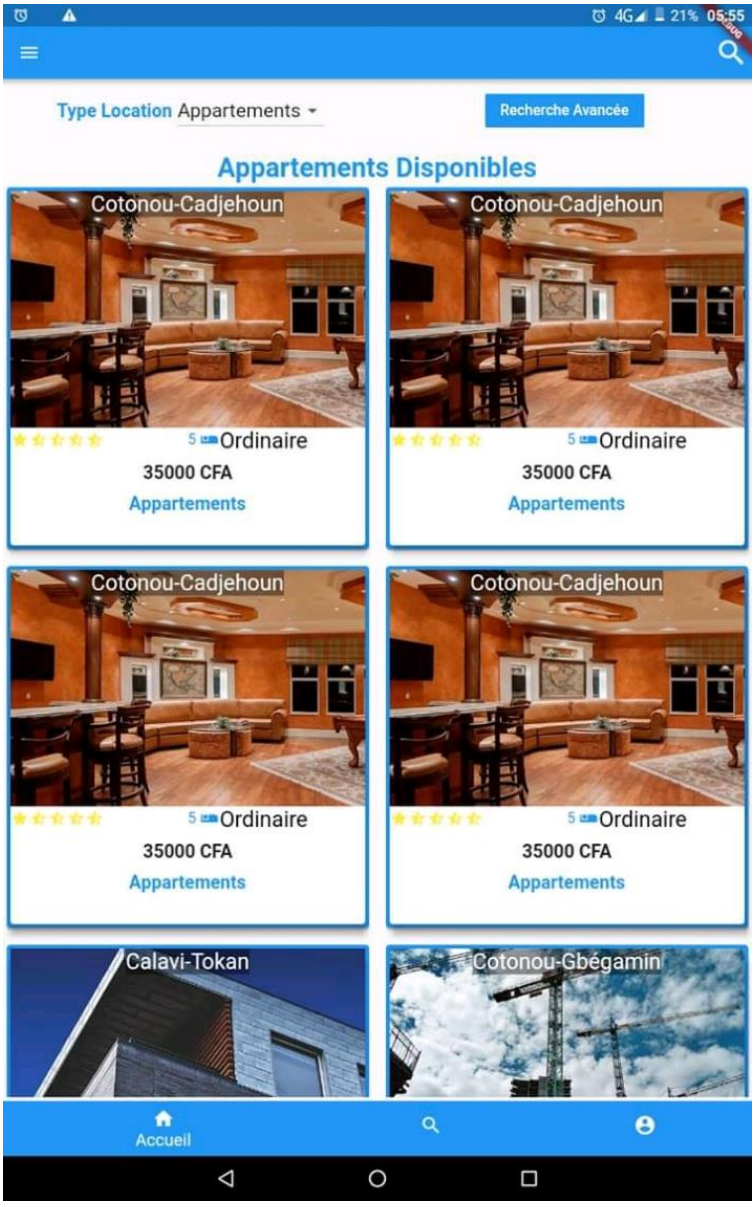

Figure 9: Interface d'accueil de l'application

Description : Il est proposé par défaut à l'utilisateur, les logements disponibles dans sa localité, indépendamment du type de logement.

❖ Menu de l'application

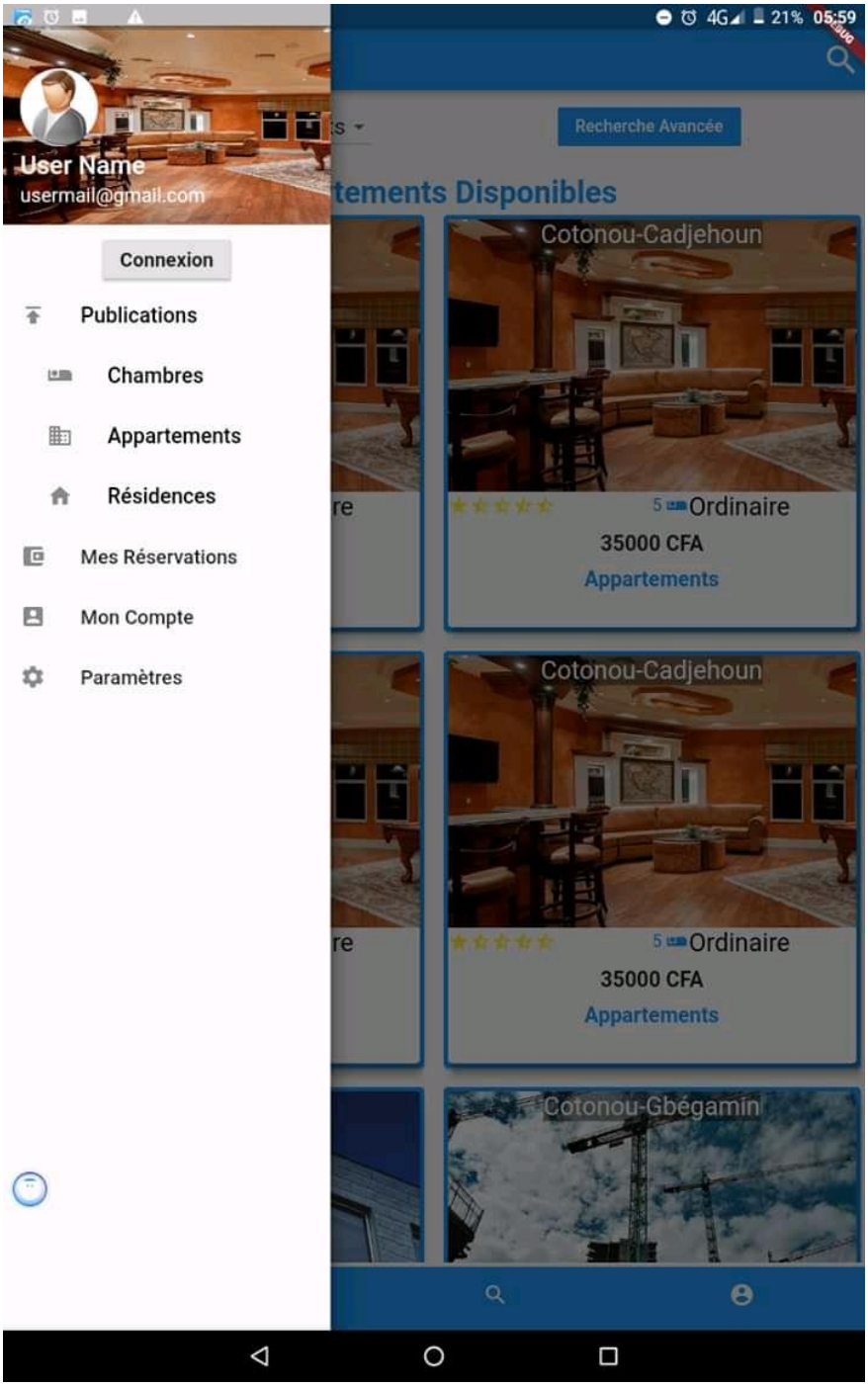

Figure 10: Menu de l'application

Description : Le menu présente une liste exhaustive des options disponibles dans l'application et il est accessible sur toutes les interfaces de l'application.

❖ Recherche rapide

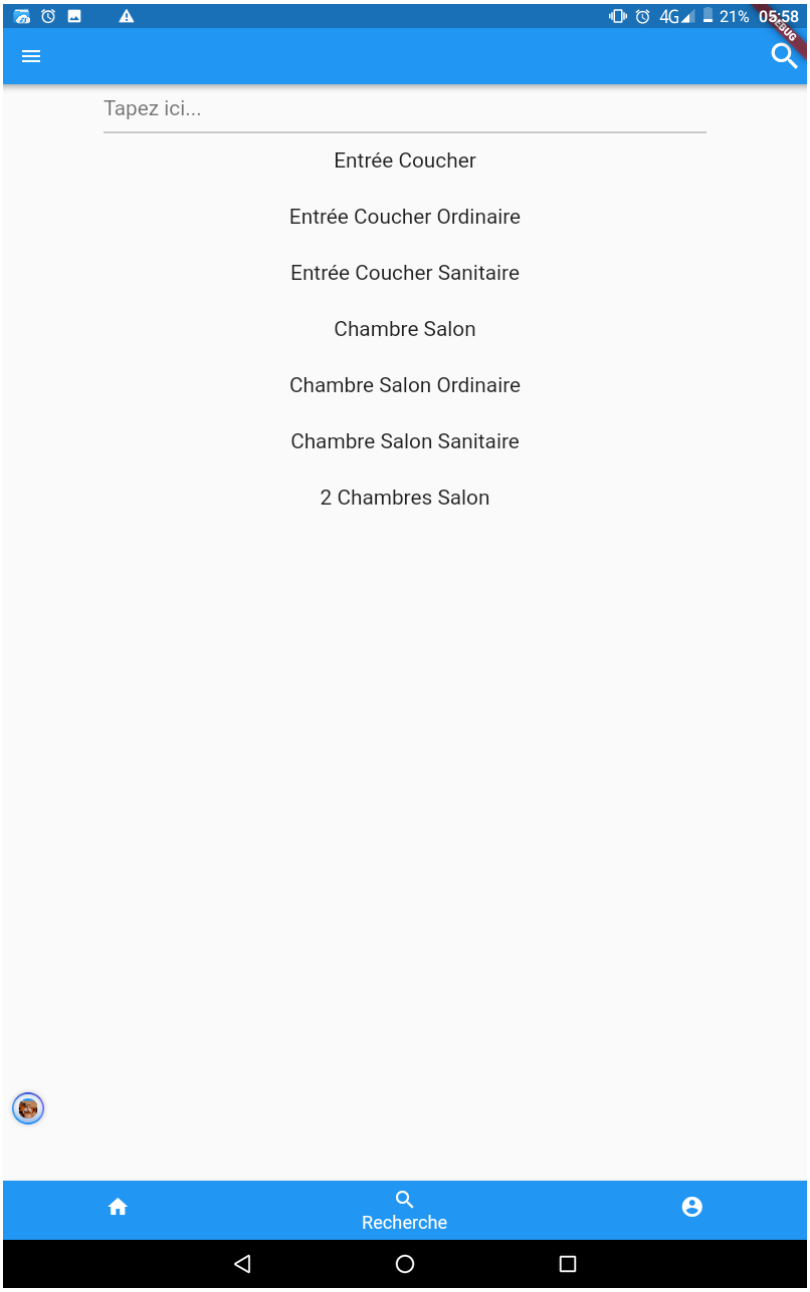

Figure 11: Interface de recherche rapide

Description : Ici l'utilisateur peut facilement filtrer les logements qui lui sont proposés suivant quelques propositions qui lui sont faites en partant de ses recherches antérieures et de celles des autres utilisateurs du système.

CONCLUSION

Ce stage académique effectué à Acumen network nous a permis d'une part de mettre en application les connaissances théoriques reçues au cours de notre formation de trois (03) ans en Analyse Informatique et Programmation (AIP) et d'autre part d'avoir un aperçu des réalités dans le monde professionnel. Face aux difficultés liées à la location d'un logement au Bénin, nous avons développé Hohaya. Son but est d'aider les citoyens rencontrant des difficultés dans la mise en location de leurs logements ou dans l'obtention d'un logement à louer. C'est pour cela qu'il associe simplicité, sécurité et surtout rapidité, dans une approche simpliste qui vise à stimuler son efficacité et à favoriser une prise en main très facile du système par l'ensemble des utilisateurs concernés. Ce nouveau système de location et de réservation de logement permet, entre autres, de trouver un logement à louer, de prendre contact avec un propriétaire pour réserver un logement ou encore de mettre des annonces pour un trouver un logement. Le travail d'analyse et de conception ayant abouti à Hohaya est basé sur le langage de modélisation UML et l'utilisation de diverses méthodes de cryptages comme le bcrypt ou le JWT à divers niveaux ont pour but de garantir la confidentialité des données et d'assurer la sécurité du système.

PERSPECTIVES

Tout le système étant appelé à évoluer dans le temps, des améliorations peuvent être apportées à ce travail afin de le rendre plus utile. On pourrait penser au développement d'une plateforme web spécialement pour permettre aux clients d'interagir beaucoup plus facilement avec l'entreprise et de pouvoir s'offrir nos services sur des supports autres que les appareils mobiles. Une possibilité de réserver le logement directement via la plateforme est également en cours d'études. De plus le système actuel n'intègre que très petit panel de type de locations, il sera donc pris en charge, dans les versions futures, une plus large gamme de biens allant des boutiques aux studios et autres locations dans le secteur de l'immobilier pour rendre l'application plus exhaustive. La nouvelle version de notre système prendra aussi en compte les insuffisances qui auraient été relevées dans l'utilisation de la première version.

BIBLIOGRAPHIE & WEBOGRAPHIE

[1] :https://fr.wikipedia.org/wiki/Acteur_(UML) – consulté le 17/02/20.

[2] :https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation – consulté le 17/02/20.

[3] :https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes consulté le 19/02/20.

[4] :https://fr.wikipedia.org/wiki/Diagramme_de_séquence - consulté le 19/02/20.

[5] :https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-etats-transitions – consulté le 20/02/20.

[6] :https://fr.wikipedia.org/wiki/Diagramme_d%27activité – consulté le 20/02/20.

[7] :https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm-consulté le 20/02/20.

[8] :https://www.lemagit.fr/definition/Java - consulté le 22/02/20.

[9]:https://www.frandroid.com/android/535194_quest-ce-que-flutter-loutil-permettant-de-creer-des-applications-android-et-ios - consulté le 22/02/20.

[10]:https://fr.wikipedia.org/wiki/Dart_(langage) - consulté le 22/02/20.

[11] :https://www.base-de-donnees.com/preferer-postgresql-a-mysql/ - consulté le 22/02/20

[12] :https://blog.ippon.fr/2015/01/30/postman-le-client-pour-api-web-qui-vous-fera-oublier-les-autres/ - consulté le 22/02/20

[13]:https://www.jetbrains.com/fr-fr/idea/features/?_ga=2.56081507.911422188.1583511617-1877008177.1583511617#polyglot-experience – consulté le 22/02/20

ANNEXES

Code source de quelques interfaces :

- Interface d'accueil

```
class _LastHomePageState extends State<LastHomePage> {
  //contenu publication
  Widget contenu;

  int _currentIndex = 0;
  Widget homeContainer;

  //DropDown Manager
  List<Categorie> _categorieList = Categorie.getCategorie();
  List<DropdownMenuItem<Categorie>> _dropDownCategorieItems;
  Categorie _selectedCategorie;

  List<DropdownMenuItem<Categorie>> buildDropDownItems(
      List<Categorie> categorieList) {
    List<DropdownMenuItem<Categorie>> items = List();
    for (Categorie cat in categorieList) {
      items.add(DropdownMenuItem(
        value: cat,
        child: Text(cat.nom),
      )); // DropdownMenuItem
    }
    return items;
  }

  @override
  void initState() {
    _dropDownCategorieItems = buildDropDownItems(_categorieList);
    print("numero ${widget.numero}");
    _selectedCategorie = _dropDownCategorieItems[widget.numero + 1].value;
    // TODO: implement initState
    print("init ${_selectedCategorie.num}");
    super.initState();
  }
```

Figure 12: Code de l'interface d'accueil

```
void _incrementCounter() {
  setState(() {
    //
  });
}

@override
Widget build(BuildContext context) {
  //print("Widget ${_selectedCategorie.num}");
  contenu = new PublicationCategorie(type: _selectedCategorie.num);
  homeContainer = Column(
    children: <Widget>[
      SizedBox(
        height: 10,
      ), // SizedBox
      Row(
        children: <Widget>[
          Expanded(
            flex: 1,
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.center,
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                Text(
                  "Type Location ",
                  style: TextStyle(
                      fontSize: 15,
                      fontWeight: FontWeight.bold,
                      color: Colors.blue), // TextStyle
                ), // Text
                DropdownButton(
                  items: _dropDownCategorieItems,
                  value: _selectedCategorie,
                  onChanged: onCategorieChanged,
                ), // DropdownButton
```

Figure 13: Code de l'interface d'accueil

- ❖ Code permettant de « persister » un logement en base de données

```
package com.memoire.wohaya.domaine;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Embedded;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Inheritance(strategy=InheritanceType.JOINED)
@DiscriminatorColumn(name="type_logement")
@DiscriminatorValue("logement")
public class Logement implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    protected Long idLogement;

    @Column(nullable = false)
    protected String categorie;

    @Column(nullable = false)
    protected int nbrChambre;

    @Column(nullable = false)
```

```
    protected int nbrCuisine;

    @Column(nullable = false)
    protected int nbrSalleBain;

    @OneToOne(optional = false)
    @JoinColumn(name = "proprietaire")
    protected Proprietaire proprietaire;

    @Embedded
    protected Localisation adresse;

    @OneToMany(mappedBy = "logement", orphanRemoval = true)
    protected List<ImageLogement> imageLogements = new ArrayList<>();

    @OneToMany(mappedBy = "logement", orphanRemoval = true)
    protected List<Reservation> reservations;

    @OneToMany(mappedBy = "logement", orphanRemoval = true)
    protected List<Publication> publications = new ArrayList<>();

}
```

❖ Code permettant de persister un utilisateur en base de données

```
package com.memoire.wohaya.domaine;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.OneToMany;
import java.io.Serializable;
```

```
import java.util.ArrayList;
import java.util.List;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@DiscriminatorValue(value = "client")
@DiscriminatorColumn(name = "role")
@Inheritance(strategy = InheritanceType.JOINED)
public class Utilisateur implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long idUtilisateur;

    @Column(nullable = false)
    private String nom;

    @Column(nullable = false)
    private String prenom;

    @Column(unique = true)
    private String telephone;

    @Column(name = "password", nullable = false)
    private String password;

    @Column(unique = true)
    private String email;

    @Column(nullable = false, length = 8)
    private String sexe;

    private String photo;

    @OneToMany(mappedBy = "auteur")
    private List<Annonce> annonces = new ArrayList<>();

    @OneToMany(mappedBy = "auteur")
    private List<Reservation> reservations= new ArrayList<>();

    @OneToMany(mappedBy = "destinataire")
```

```
    private List<Notification> notifications = new ArrayList<>();

    @OneToMany(mappedBy = "auteur")
    private List<Appreciation> appreciations = new ArrayList<>();

}
```