RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
UNIVERCITÉ DE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTÉ DE GÉNIE ELECTRIQUE ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

MÉMOIRE DE FIN D'ÉTUDE

En vue de l'obtention du diplôme de master en informatique

Option Ingénierie des Systèmes d'Information

Thème

Réalisation d'une application de mise en relation artisan-particulier « **VieFacile** »

Réalisé par :
Mr. Ziani Yanis
Mr. Ouidir Mohamed

Dirigé par :
Mr. Talbi Saïd

Promotion : 2019/2020

Remerciements

Nous tenons à exprimer toute notre reconnaissance à notre encadreur, Monsieur Saïd TALBI. Nous le remercions de nous avoir orienté, aidé et conseillé.

Nous adressons nos sincères remerciements aux membres du jury d'avoir accepté d'évaluer ce travail, à tous les professeurs, intervenants et toutes les personnes par leurs écrits, leurs conseils et leurs critiques, et qui ont accepté de nous rencontrer et de répondre à nos questions.

Nous remercions nos parents, qui ont toujours été là pour nous. Nous remercions nos sœurs et frères pour leurs encouragements.

Enfin, nous remercions notre cher ami Ahmed TOUATI qui nous a offert son local pour travailler pendant les durs moments du confinement.

À tous nous disons merci. Tout notre respect et gratitude.

Table des matières

Table des Figures

Introduction Générale

La mise en relation en ligne est l'une des activités les plus tendance du moment. Selon certains observateurs, elle rapporte beaucoup aux sites eux-mêmes sans parler des avantages qu'elle procure aux particuliers comme aux professionnels. D'ailleurs, les sites de mise en relation se multiplient à grande vitesse sur le net. La création d'une plateforme de mise en relation requiert pourtant un travail considérable, du temps et de l'argent.

La mise en relation est un secteur qui rapporte à condition de trouver la niche idéale pour développer son business. La commission sur chaque transaction peut varier d'un secteur à l'autre. Dans le secteur du bâtiment et de la construction, les transactions sont irrégulières, mais élevées. À l'inverse, dans le secteur du petit artisanat, les transactions peuvent être plus fréquentes.

Dans le cadre de notre projet de fin d'étude, nous avons choisi de développer une plateforme mobile que nous avons appelé ***VieFacile***. Cette plateforme aura pour objectif de mettre en relation les particuliers demandeurs de services et les artisans fournisseur de ces services. Notre choix est motivé par l'évolution très prometteuse de ce secteur pour la création de la start-up notamment ici en Algérie où le domaine est à peine à ses débuts.

Afin de mener à bien notre projet de fin d'étude, nous avons considéré le plan de travail suivant :

Le premier chapitre intitulé "Plateformes en ligne de mise en relation artisan-particulier" consiste à décrire en détail le domaine tout en présentant certaines plateformes les plus utilisées dans le monde. Dans ce chapitre, il sera aussi question de la réglementation qui encadre cette activité dans les pays pionniers.

Le deuxième chapitre intitulé "Analyse et spécification des besoins de plateforme ***VieFacile*** a pour objectif de fournir une vue experte de notre plateforme de mise en relation artisan-particulier sur le plan fonctionnel du point de vu utilisateur.

Le troisième chapitre intitulé "Conception de la plateforme ***VieFacile*** sera consacré à la description opérationnelle de notre application mobile sur le plan technique. Par-ailleurs, un modèle conceptuel de données manipulées dans notre application, ainsi que le modèle logique correspondant seront présentés dans ce chapitre.

Le quatrième chapitre intitulé "Implémentation de l'application ***VieFacile*** permettra, dans un premier temps, de présenter l'environnement de développement que nous avons utilisé pour l'implémentation de notre plateforme. Dans un second temps, ce chapitre donnera un aperçu global sur les différents espaces implémentés pour notre application mobile ***VieFacile***, ainsi que le modèle physique de données associé au modèle logique proposé dans le chapitre précédent.

Enfin, nous concluons notre mémoire par un bilan qui portera sur l'apport de ce projet pour notre formation d'ingénieur en informatique, ainsi qu'une évaluation des objectifs réalisés. Par-ailleurs, ce chapitre ouvrira les perspectives permettant de compléter et surtout la mise en service de notre application ***VieFacile***.

I. Plateformes de Mise en Relation Artisan-Particulier

1.1 Introduction

Ce chapitre est consacré au domaine des plateformes en ligne de mise en relation. Plus précisément, nous abordons les motivations et les aspects techniques et règlementaires (i.e., juridiques) qui ont permis le développement de cette activité sur le web. Puis, une analyse de l'existant concernant cette activité en Algérie fera l'objet d'une étude.

1.2 Plateformes de mise en relation

Dans un monde toujours plus connecté, les nouvelles technologies de l'information ont permis de contracter (voire d'effacer) les distances physiques. Des plateformes web permettent ainsi de mettre en relation des personnes qui, autrement, ne se seraient jamais rencontrées. Dans ce contexte, les sites internet de mise en relation se développent à grande vitesse. Ils font rencontrer des offreurs et des demandeurs de services de tout horizon : des professionnels entre eux, des particuliers entre eux, ou des professionnels avec leurs clientèles [1].

Derrière l'expression générique de "plateforme de mise en relation" se cache une vaste réalité, aujourd'hui incontournable dans l'économie du web, fondée sur différents modèles d'affaires, et qui touche de plus en plus de secteurs autrefois peu digitalisés.

1.3 Facteurs de développement

Plusieurs raisons peuvent expliquer le succès des plateformes de mise en relation. D'abord, elles innovent en facilitant ou en offrant l'accès à certains services. Ensuite, ces plateformes web permettent de mettre en relation des "prestataires" et des "clients" de façon dématérialisée. Rapides et simples à utiliser, elles encouragent une économie collaborative et une mutualisation des biens . Dans la suite de cette section, nous présentons brièvement quelques facteurs clés qui ont permis l'émergence des plateformes de mise en relation.

1.3.1 Internet

Internet et plus généralement toute forme d'utilisation des réseaux électroniques "ouverts" ont et auront un impact déterminant sur la société et son devenir. L'échange ouvert de données numérisées par le biais des messageries électroniques, des procédures de transfert de fichiers informatisés et des sites Web constitue certainement le facteur clé de développement des sociétés modernes.

Internet est le réseau informatique mondial accessible au public. C'est un réseau de réseaux, à commutation de paquets, sans centre névralgique, composé de millions de réseaux aussi bien publics que privés, universitaires, commerciaux et gouvernementaux, eux-mêmes regroupés en réseaux autonomes (il y en avait plus de 91 000 en 2019). L'information est transmise via Internet grâce à un ensemble standardisé de protocoles de transfert de données, qui permet des applications variées comme le courrier électronique, le World Wide Web, la messagerie instantanée, le pair-à-pair, le streaming et la téléconférence. C'est l'apparition du Web qui a popularisé Internet dans les années 1990. Un nombre croissant d'objets sont connectés à Internet dans les années 2010, formant l'Internet des objets [2].

1.3.2 Web

Incontestablement, le web est à l'origine de l'émergence de la digitalisation de plusieurs services (e.g. commerce, hôtellerie, santé, transport, restauration, bénévolat, entre-aide) sur le réseau Internet. Tim Berners-Lee, chercheur britannique, a inventé le Web au CERN en 1989. À l'origine, le projet, baptisé "World Wide Web", a été conçu et développé pour que des scientifiques travaillant dans des universités et instituts du monde entier puissent s'échanger des informations instantanément. Le 30 avril 1993, le CERN a mis le logiciel du "World Wide Web" dans le domaine public. Puis, il a émis une version suivante de l'application sous licence libre, une façon plus sûre de maximiser sa diffusion. Ce faisant, il a permis à la Toile de se tisser [3].

1.3.3 Mobilité des applications

Le développement de smartphones et l'introduction d'applications mobiles ont permis une évolution très considérable pour l'économie numérique. En effet, environ 200 milliards d'applications mobiles ont été téléchargées jusqu'en 2015, alors qu'en 2009, deux milliards seulement l'avaient été. De 2011 à 2015, les applications mobiles ont généré un revenu de 45,37 milliards de dollars. En 2017, 178,1 milliards d'applications mobiles ont été téléchargées. En 2018, le revenu généré se chiffre à 205,4 milliards [4].

1.3.4 Faible coût de développement et déploiement

Pour un budget limité, il est possible dans un premier temps de faire un compromis avec toutes les fonctionnalités auxquelles nous pensons. Au lieu de se concentrer sur tout ce que notre application peut réaliser, nous devrions se focaliser sur les fonctionnalités indispensables. Il est très important d'être flexible et accepter de s'adapter à l'interface conçue dans un premier temps. Le produit final sera quelque

chose de viable et au moins concrétisera l'idée de base qui pourra bien être complétée par d'autres fonctionnalités dans le futur pour étendre les possibilités de l'application. Quant au déploiement, il existe un grand nombre d'hébergeur de sites web ou d'applications mobiles qui applique un tarif très abordable voire même gratuit.

1.3.5 Réactivité des autorités

Lancer une application sur un marché virtuel suppose en aval la régularisation juridique des mentions légales à afficher pour l'utilisateur, des conditions générales d'utilisation. Les autorités gouvernementales doivent impérativement suivre en permanence l'évolution de la virtualisation de la société et du marché afin d'apporter des réponses juridiques à l'utilisation de ces applications en respectant des délais très raisonnables.

1.4 Architecture de l'infrastructure des plateformes de mise en ligne

Comme toute application web dynamique nécessitant la manipulation de données, les plateformes de mise en relation reposent sur une architecture à 3 niveaux (i.e., architecture trois-tiers) entre serveur de données, serveur d'applications et client web (voir Figure 1.1). Ce type d'architecture, plus complexe que le client-serveur, permet l'accès aux bases de données stockées elles-mêmes sur un serveur. Plus généralement, elle offre la possibilité d'exécuter des programmes du côté serveur. Les résultats sont prétraités avant leur envoi final, en format HTML, vers le navigateur [5].

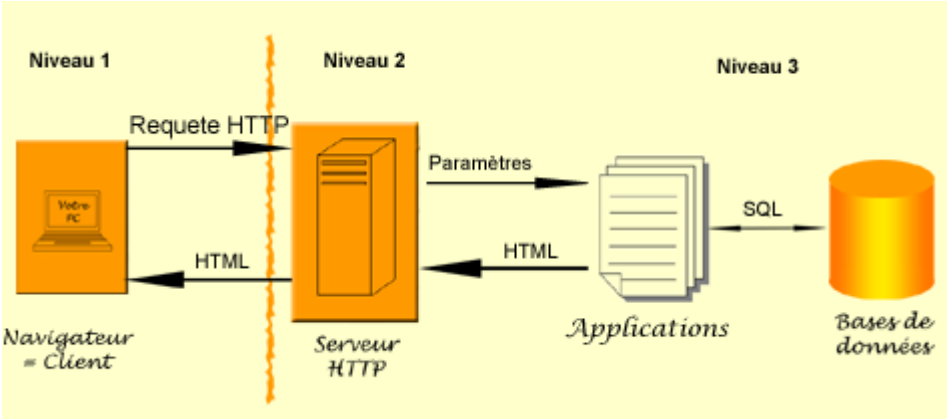

Figure I. 1 Architecture trois-tiers

Les 3 niveaux s'articulent dès lors de la manière suivante :

1. le premier niveau s'occupe de l'interface avec l'utilisateur depuis le navigateur.
2. le second héberge le serveur web qui est complété par le serveur d'application qui exécutent les traitements demandés lors de l'appel HTTP d'une page. Le serveur HTTP, aussi appelé middleware, est donc à la fois serveur et client. Serveur vis-à-vis du navigateur et client par rapport au serveur d'applications à qui il envoie une requête et dont il attend en retour le résultat. Une fois reçus, le serveur HTTP les compose dans un format assimilable par le navigateur client.
3. le troisième niveau assure la gestion des données au sein d'un SGBD (Système de Gestion de Bases de Données) et répond aux requêtes du serveur HTTP.

1.5 Exemples de plateformes à caractère économique

Dans cette section, nous allons présenter quelques exemples de plateformes de mise en relation par secteur d'activité.

1.5.1 E-commerce

Amazon est une entreprise de commerce électronique américaine basée à Seattle. Elle est un des géants du Web, regroupés sous l'acronyme GAFAM, aux côtés de Google, Apple, Facebook et Microsoft. L'activité initiale de la société Amazon concernait la vente à distance de livres, avant que la société ne se diversifie dans la vente de produits culturels, puis marchands. Aujourd'hui, certains produits alimentaires peuvent aussi être commandés via Amazon. Créée par Jeff Bezos en juillet 1994, l'entreprise a été introduite en bourse au NASDAQ en mai 1997. En 2018, Amazon comptait un effectif de 647600 employés à travers le monde entier et a réalisé un chiffre d'affaire de plus de 238 milliards de dollars [6].

1.5.2 Location d'appartements de particuliers

Airbnb est une plateforme communautaire payante de location et de réservation de logements de particuliers fondée en 2008 par les Américains Brian Chesky, Joe Gebbia et Nathan Blecharczyk. Airbnb permet à des particuliers de louer tout ou une partie de leur propre habitation comme logement d'appoint. Le site offre une plateforme de recherche et de réservations entre la personne qui offre son logement et le vacancier qui souhaite le louer. Il couvre plus de 1,5 million d'annonces dans plus de 34 000 villes et 191 pays. De la création, en août 2008, jusqu'en juin 2012, plus de 10 millions de nuits ont été réservées sur Airbnb. En 2019, Airbnb comptait un effectif de 12 376 employés dans le monde entier [7].

1.5.3 Voiture de transport avec chauffeur

Uber est une entreprise technologique américaine qui développe et exploite des applications mobiles de mise en contact d'utilisateurs avec des conducteurs réalisant des services de transport. L'entreprise est basée dans la ville californienne de San Francisco, aux États-Unis. En 2015, elle est valorisée à 50 milliards de dollars et ses applications sont commercialisées dans plus de 310 villes dans le monde. Elle a été fondée par Garrett Camp, Idir Hedjem et Travis Kalanick. En 2019, Uber comptait un effectif de 6700 employés dans le monde et a réalisé un chiffre d'affaire dépassant les 14 milliards de dollars [8].

1.5.4 Livraison des plats cuisinés

Uber Eats est un service de livraison de plats cuisinés lancé par Uber en 2015 et basé à San Francisco, en Californie. Les commandes sont prises via une application mobile ou web. Service permet de mettre en relation les consommateurs avec les professionnels de la restauration. En 2018, Uber Eats a réalisé un chiffre d'affaire dépassant 1.3 milliards de dollars [9].

1.5.5 Santé à domicile

Medicalib propose aux patients une solution simple et rapide pour trouver une infirmière près de chez eux. **L**a plateforme a pour objectif de simplifier la relation patient-soignant : le patient fait une demande de prise en charge, et l'équipe Medicalib s'occupe de trouver une infirmière disponible et qualifiée pour les soins en moins d'**une heure.** La startup travaille avec plus de 500 infirmiers libéraux et est présente dans une quinzaine de villes en France, dont Paris, Marseille, Lyon, Toulouse, Nice et Bordeaux. Depuis son lancement, elle a pris en charge 4000 patients, et elle aide tous les mois plus de 600 personnes à trouver un infirmier, ce qui en fait la première plateforme santé dédiée aux soins à domicile en France [10].

1.5.6 Travaux en tout genre

Ces derniers temps, plusieurs plateformes de mise en relation ont pu voir le jour à travers le monde dans le domaine du bâtiment notamment le nombre d'opportunités qui ne cesse de progresser. Une étude détaillée de l'évolution des strats up opérant dans ce secteur est présentée dans la section suivante.

1.6 Mise en relation artisan-particulier

Avec l'avènement d'internet, les artisans peuvent désormais profiter de l'outil digital pour s'adapter aux nouvelles habitudes de leurs clients issues des

technologies de l'information et communications, et permettre à ces artisans d'avoir plus d'opportunités de travaux de rénovation.

Également appelés plateformes d'artisans, ces plateformes de mise en relation artisans/ particuliers existent pour faciliter le contact entre les acteurs concernés pour la réalisation de travaux liés au monde de l'artisanat et de bâtiments, tout en fournissant la compétence appropriée.

1.6.1 Fonctionnement général du côté artisan

D'une manière générale, les artisans souhaitant offrir leurs services via les plateformes de mise en relation artisans/particuliers doivent d'abord s'inscrire sur ces plateformes en renseignant leurs coordonnées personnelles, domaines de compétences et éventuellement leurs réalisations afin de mettre en valeur leur candidature augmentant ainsi leur chance de décrocher des contrats de réalisations.

1.6.2 Fonctionnement général du côté particulier

Les particuliers souhaitant d'exprimer leur besoin de réaliser une ou plusieurs taches quelconques via les plateformes de mise en relation artisans/particuliers doivent d'abord s'inscrire sur ces plateformes en tant que client et cela en renseignant leurs coordonnées personnelles.

1.6.3 Evolution dans le monde

HomeAdvisor [11] apparue aux états unis d'Amérique en 1998 est le premier site mondial de mise en relation entre particulier et artisan. Avec plus de 30 millions utilisateurs il est le leader mondial dans ce domaine. Aujourd'hui, il offre ces services dans plusieurs pays dans le monde.

Ensuite vient l'apparition de plusieurs autres plateformes dans ce domaine tel que le site français Youpijob [12] qui est présents dans trois pays, la France, la Suisse et la Belgique et qui contient plus de 12 000 artisans et plus de 500 000 jobs réalisé, et le site Frizbiz [13] dédié au monde de l'habitat : petits travaux de bricolage, ménage, repassage, déménagement, peinture, pose de sol, fixation de tringle à rideau, changement d'ampoule avec plus de 200 000 utilisateurs.

1.6.4 Evolution en Algérie

En Algérie, d'après les recherches que nous avons effectué sur le net, nous pouvons dire que les plateformes de mise en relation artisans/ particuliers sont encore à leur début malgré l'immense potentiel d'opportunités qui se présentent dans le domaine

du bâtiment. En effet, les sites opérant dans ce domaine en Algérie se contentent juste de faire passer des annonces d'offre de services et/ou travaux sans pour autant garantir le suivi de l'artisan (i.e., établissement de la confiance entre le client et l'artisan) pour une meilleure qualité de service.

Plus précisément, la seule plateforme qui permet d'exprimer un service sous forme d'une annonce est le site dédié initialement pour la vente et l'achat « Ouedkniss » [14]. Et pour les autres plateformes qui ont essayé d'émerger dans ce domaine en Algérie, nous citons « Fixit » [15] et « Allo khedma » [16] qui n'ont pas eu le succès escompté.

1.7 Conclusion

Les plateformes de mise en relation artisans/ particuliers facilitent aux particuliers la démarche de recherche d'un artisan qualifié, et permettent de valoriser les compétences individuelles des artisans en les aidant à se faire visibles pour mieux développer leur activité. En Algérie, ce secteur est encore très prometteur vu le potentiel d'opportunités qui se présentent notamment dans le domaine du bâtiment. Ceci, nous a encouragés d'opter pour le développement d'une plateforme de mise en relation mobile offrant toute les fonctionnalités nécessaires pour atteindre les objectifs sus cités.

II. Analyse et Spécification des Besoins

2.1 Introduction

L'analyse et la spécification des besoins représentent la première étape dans le cycle de développement d'un système. Cette étape consiste à identifier les acteurs qui interagissent avec le système et associe à chacun d'eux l'ensemble de fonctionnalités qui lui permet de réaliser les objectifs fixés.

Dans ce chapitre, nous allons donc mettre en évidence les acteurs intervenant dans notre plateforme de mise en relations artisans/ particuliers tout en détaillant les fonctionnalités offertes pour chacun à travers l'utilisation des diagrammes de cas d'utilisation du langage UML et les scénarios associés.

2.2 Langage de modélisation UML

Le langage de modélisation unifié, de l'anglais *Unified Modeling Language* UML, issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système [17].

L'UML comprend un certain nombre de diagrammes utilisés pour l'analyse et la conception de logiciel, parmi lesquels nous citons :

- Diagramme de cas d'utilisation, utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés;

- Diagramme de séquence, représente graphiquement les interactions entre les acteurs et le système selon un ordre chronologique dans la formulation *Unified Modeling Language*. Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation;

- 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 (multi-threads ou multi-processus). Le diagramme d'activité est également utilisé pour décrire un flux de travaux (workflow);

- Diagramme de classe, est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations

entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

Dans notre cas, dans un premier temps, nous utiliserons essentiellement les diagrammes de cas d'utilisation pour clarifier les fonctionnalités de notre plateforme de mise en relation artisans/ particuliers.

2.3 Spécification des besoins

La spécification des besoins comprend, d'une part, l'aspect fonctionnel c'est-à-dire les opérations devant être assumée par le système, et d'autre part l'aspect non fonctionnel permettant la facilité d'utilisation.

2.3.1 Besoins fonctionnels

Les besoins fonctionnels de notre application mobile de mise en relation artisans/ particuliers peuvent être énumérés comme suit :

1. Le système doit permettre aux clients de s'inscrire pour pouvoir soumettre en ligne les travaux et/ou bricoles qu'ils souhaitent réalisés chez eux à la maison.

2. Le système doit permettre aux artisans de s'inscrire afin de proposer en ligne leurs offres de service en mettant en avant leurs compétences;

3. Permettre au client d'avoir un ou plusieurs devis estimatifs pour un travail soumis en ligne depuis son espace;

4. L'application doit signaler à l'artisan toutes les offres de travail qui portent sur son domaine de qualification et également lui fournir un moyen de soumettre des devis estimatifs aux demandeurs de service.

5. Le système, grâce au module GPS, doit permettre la localisation géographique du demandeur et fournisseur de service pour réaliser des contacts privilégiant la proximité entre artisan/ particulier;

6. L'application doit constamment aider les clients à trouver la meilleure compétence, en mettant en place un système de notation dynamique client-professionnel;

2.3.2 Besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement.

1. Ergonomie et souplesse : l'application doit offrir une interface conviviale et ergonomique exploitable par l'utilisateur en rendant simple l'usage de toute les fonctionnalités proposées par notre application;

2. Rapidité : l'application doit optimiser les traitements pour avoir un temps de génération d'une réponse à une demande de réalisation d'un travail très raisonnable pour le client;

3. Efficacité : l'application doit être fonctionnelle indépendamment de toutes circonstances pouvant entourer les utilisateurs des deux côtés

4. Maintenabilité et scalabilité : le code de l'application doit être lisible et compréhensible afin d'assurer son état évolutif et extensible par rapport aux besoins du marché.

2.4 Identification des acteurs

Un acteur représente un rôle joué par une personne externe (utilisateur humain, matériel ou autre système), qui interagissent directement avec le système étudié. Les acteurs de notre application sont :

2.4.1 Visiteur

Toute personne visitant l'application ayant pour intérêt l'inscription pour bénéficier des services offerts par l'application (i.e., réaliser des annonces pour la recherche d'artisans qualifier du côté client, et consultation d'annonces pour la soumission des offres du coté artisan).

2.4.2 Particulier demandeur de service (client)

Toute personne ayant un compte personnel sur l'application dédiée aux clients, il peut aussi jouer le rôle d'un visiteur. Un client accède à l'application pour exprimer une demande de réalisation d'un travail sans ambiguïté avec maximum d'avantages (gain de temps, argent et possibilité du choix).

2.4.3 Artisan (fournisseur de service)

Toute personne ayant un compte artisan sur l'application dédiée aux artisans, ces derniers peuvent mettre en ligne leurs services à tout temps au porté des utilisateurs qui souhaitent réaliser une ou plusieurs taches.

2.4.4 Administrateur

Toute personne chargé de faire respecter les règles d'utilisation du système. Souvent on parle aussi de l'autorité.

2.5 Cas d'utilisation

Les cas d'utilisations *use cases* sont des outils formels qui permettent de déterminer les besoins des utilisateurs (i.e., les acteurs) et de capturer les exigences fonctionnelles d'un système. Un cas d'utilisation doit exprimer ce que le système doit faire sans préjuger de la façon dont cela sera fait.

Dans ce qui suit, les fonctionnalités majeures de notre application de mise en relation seront représentées à l'aide de diagrammes d'utilisation.

2.5.1 Diagramme de cas d'utilisation côté «visiteur»

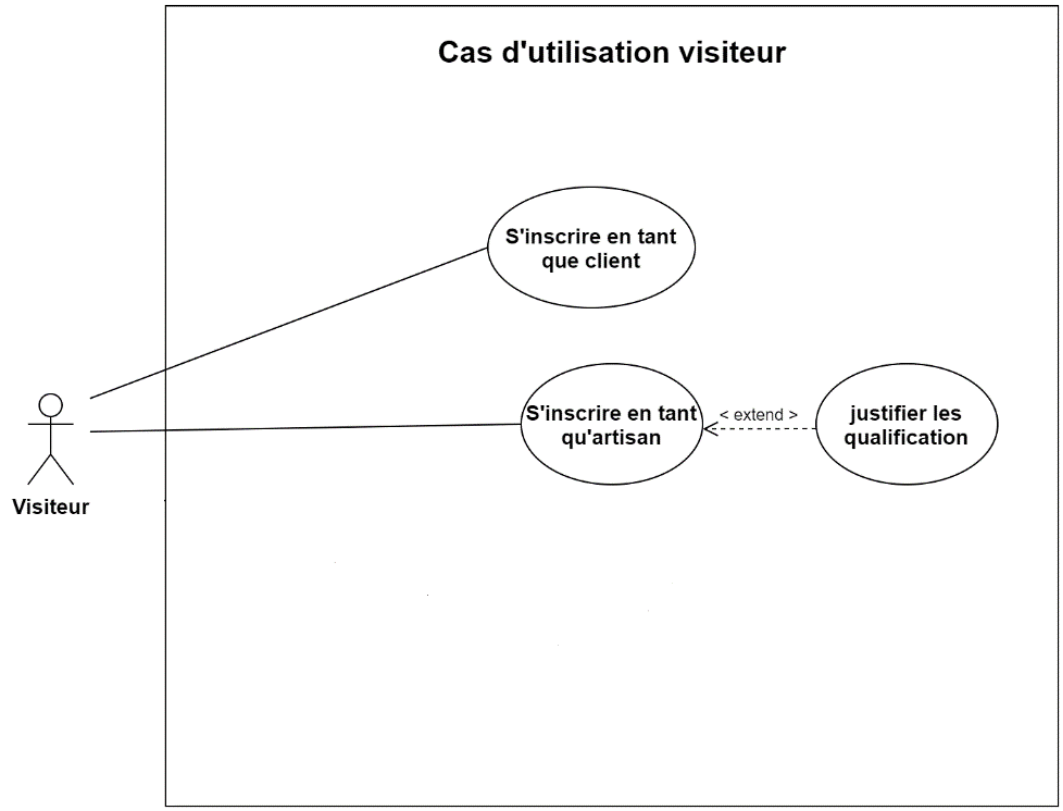

Figure II. 1 Diagramme de cas d'utilisation côté visiteur

Scénario du cas d'utilisation s'inscrire en tant que artisan

1. Le visiteur télécharge l'application artisan.
2. Le visiteur remplit le formulaire d'inscription en fournissant toute ses coordonnées personnelles et clique sur le bouton « Inscription ».
3. Une fois le bouton « Inscription» est cliqué, le système vérifie si le formulaire ne contient pas d'erreurs afin de rediriger le visiteur vers l'interface complément inscription.
4. Une fois le visiteur est sur l'interface complément inscription, il clique sur le bouton « ajouter documents » afin de fournir les documents qui justifient ses qualifications.
5. Une fois que les documents sont envoyés, le visiteur attendra qu'il soit approuvé par l'administrateur pour qu'il redevienne un artisan sur la plateforme.

2.5.2 Diagramme de cas d'utilisation côté «client»

L'utilisateur qui veut devenir un client accède à l'espace client en tant que visiteur pour s'inscrire. Une fois inscrit un compte client lui sera attribué.

L'objectif principal du client est d'exprimer ses besoins sous forme de taches en postant une annonce, une fois l'annonce est postée des devis de la part des artisans intéressés lui seront communiqués afin de pouvoir choisir l'artisan qui lui convient. Une fois le travail est accompli, le client pourra évaluer l'artisan. Le client peut aussi modifier les coordonnées de son profil à tout moment.

Dans ce qui suit nous présentons les scénarios pour les cas d'utilisation : consulter les réalisations des artisans, poster une annonce et mettre à jour son profil.

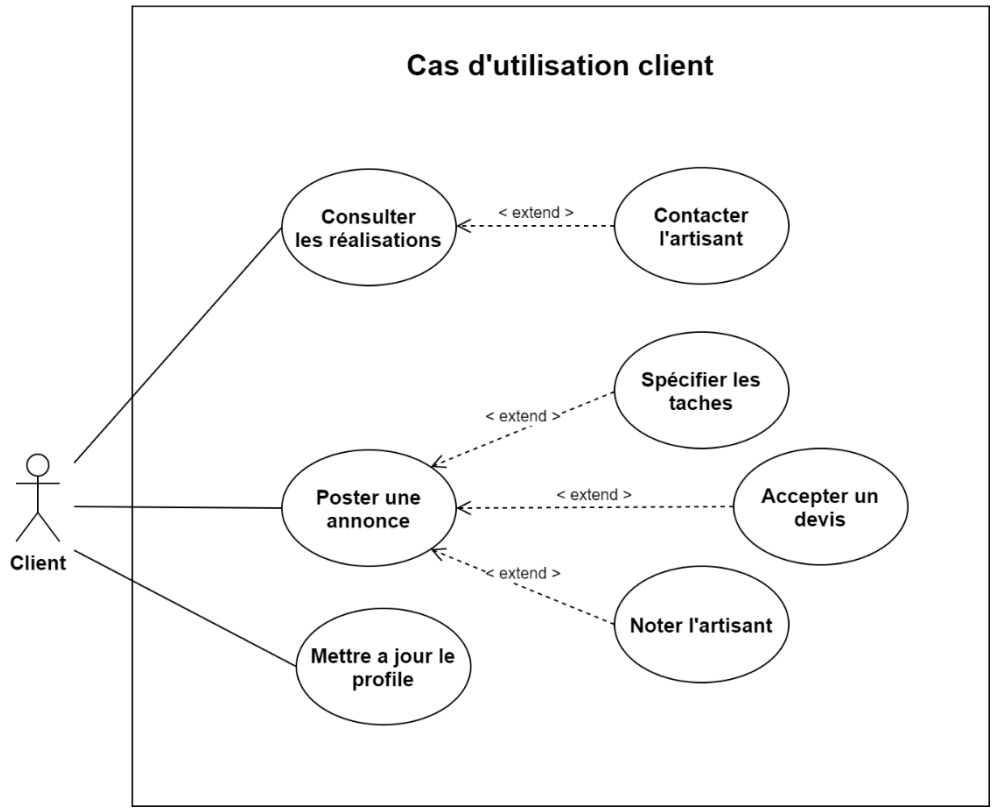

Figure II. 2 Diagramme de cas d'utilisation côté client

Scénario du cas d'utilisation consulter les réalisations des artisans

1. Le client se connecte avec son compte sur l'application client.
2. Une fois connecté, le client se retrouvera sur l'interface d'accueil qui permet de rechercher des artisans pour un travail quelconque.
3. Le client saisie les métiers et l'emplacement du travail qu'il souhaite réaliser et clique sur le bouton « Rechercher ».
4. Une fois que le bouton « Rechercher » est cliqué, une liste d'artisans apparaitra.
5. Le client choisit un artisan et cliques sur « consulter le profil » afin d'accéder au profil de l'artisan choisi.
6. Une fois accédé au profil de l'artisan, toutes les réalisations de ce dernier apparaitront.
7. Le client peut choisir une réalisation quelconque de l'artisan afin d'accéder à ses détails.

Scénario du cas d'utilisation poster une annonce

1. Le client se connecte avec son compte sur l'application client.
2. Une fois connecté, le client se retrouvera sur l'interface d'accueil ou se trouve un bouton-menu qui affiche un menu en cliquant dessus.
3. Une fois que le menu est affiché, le client clique sur « Ajouter une annonce » ainsi ce dernier sera redirigé vers l'interface « Poster annonce ».
4. Une fois sur l'interface « Poster annonce » le client doit fournir un titre pour l'annonce, l'emplacement du travail à effectuer, les métiers concernés par l'annonce, les taches à effectuer et les photos du chantier s'il possède.
5. Une fois que toutes les informations de l'annonce sont fournies, le client clique sur le bouton « Poster ».
6. Une fois que le bouton «Poster » est cliqué, l'annonce sera postée et visible aux artisans concernés.

Scénario du cas d'utilisation mettre à jour le profil

1. Le client se connecte avec son compte sur l'application client.
2. Une fois connecté, le client se retrouvera sur l'interface d'accueil ou se trouve un bouton-menu qui affiche un menu en cliquant dessus.
3. Une fois que le menu est affiché, le client clique sur «Profil » afin d'accéder à son profil.
4. Une fois que le client est sur son profil, il pourra modifier toutes ses informations en cliquant sur l'icône modifier qui se trouve à côté de chaque information.

2.5.3 Diagramme de cas d'utilisation côté «artisan»

L'utilisateur qui veut devenir un artisan accède à l'espace artisan en tant que visiteur pour s'inscrire. Une fois inscrit un compte artisan lui sera attribué.

L'objectif principal de l'artisan est de consulter les annonces à proximités postées par les clients afin de proposer des devis.

Une fois le travail fini, l'artisan pourra mettre à jour son profil en postant de nouvelles photos sa dernière réalisation.

L'artisan peut aussi modifier les coordonnées de son profil à tout moment.

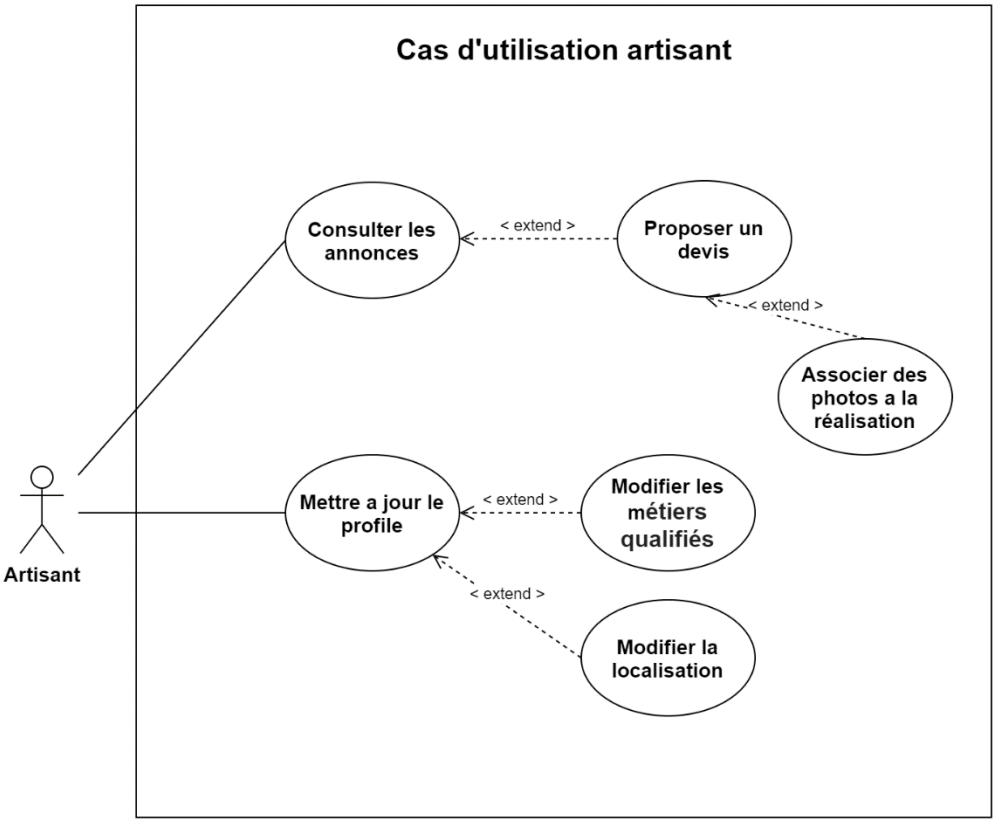

Figure II. 3 Diagramme de cas d'utilisation côté artisan

Scénario du cas d'utilisation proposer un devis

1. L'artisan se connecte sur son compte sur l'application artisan.
2. Une fois connecté, l'artisan se retrouve sur la page d'accueil ou se trouve les annonces postés par les clients.
3. L'artisan choisi l'annonce qu'il veut et clique sur « afficher détail » afin d'afficher les détails de l'annonce.
4. Une fois sur la page détail annonce, l'artisan peut proposer un devis pour le travail posté une cliquant sur le bouton « proposer un devis ».
5. Une fois le bouton « proposer un devis » est cliqué, un formulaire à remplir sera affiché pour l'artisan afin de fournir le montant proposé et la durée estimé par l'artisan.
6. Une fois que le formulaire est rempli, l'artisan clique sur le bouton «Proposer », ainsi un devis sera envoyé au client de la part de cet artisan.

Scénario du cas d'utilisation mettre à jour le profil

1. L'artisan se connecte avec son compte sur l'application artisan.

2. Une fois connecté, l'artisan se retrouvera sur l'interface d'accueil ou se trouve un bouton-menu qui affiche un menu en cliquant dessus.
3. Une fois que le menu est affiché, l'artisan clique sur «Profil » afin d'accéder à son profil.
4. Une fois que l'artisan est sur son profil, il pourra modifier toutes ses informations en cliquant sur l'icône modifier qui se trouve à côté de chaque information.

2.5.4 Diagramme de cas d'utilisation côté «administrateur»

L'administrateur qui accède à l'espace d'administration en tant que visiteur pour s'authentifier, une fois authentifié une session lui sera ouvert.

L'objectif principal de l'administrateur est de consulter les documents des artisans afin de les approuver ou les désapprouver.

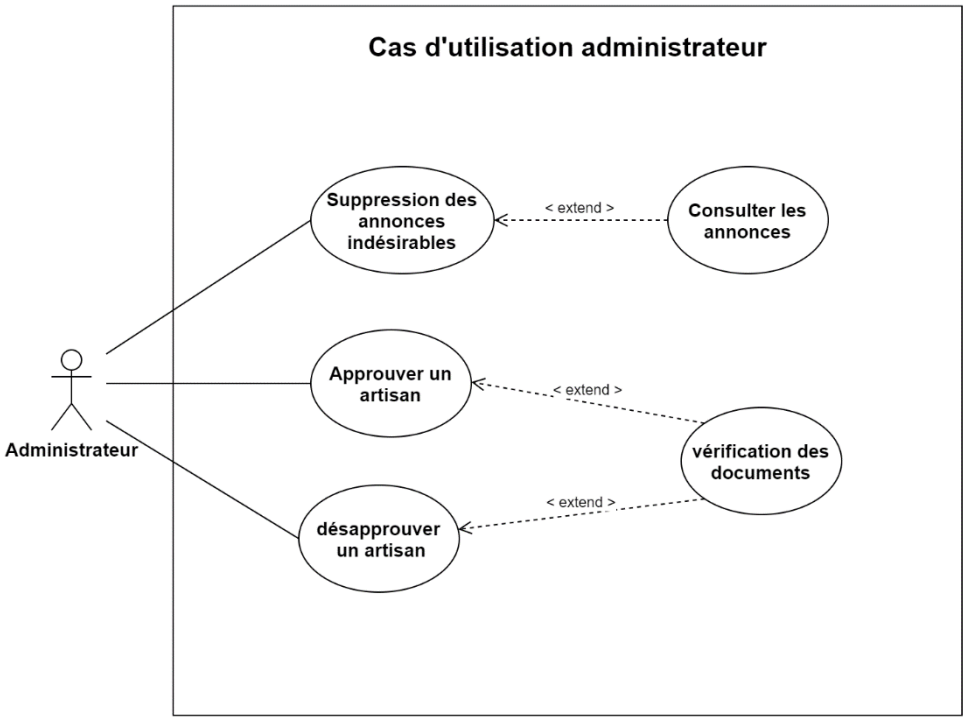

Figure II. 4 Diagramme de cas d'utilisation côté administrateur

Scénario du cas d'utilisation approuver un artisan

1. L'administrateur accède à la page d'accueil ou se trouvent toute les demandes d'approbations en attente.

2. L'administrateur choisit une demande et clique dessus afin d'afficher tous les documents et les informations à propos de la demande.
3. Une fois que les documents et les informations sont vérifiés par l'administrateur, ce dernier clique sur le bouton « Approuver » afin d'approuver l'artisan en question.

Scénario du cas d'utilisation supprimer une annonce indésirable

1. L'administrateur accède à la page d'accueil ou le bouton « Afficher les annonces ».
2. L'administrateur clique sur le bouton « Afficher les annonces » afin d'afficher toutes les annonces postées par les clients.
3. L'administrateur vérifie les annonces, et s'il trouve une annonce avec un contenu indésirable, tels que des termes déplacés ou des demande de services illégales, il clique sur « supprimer l'annonce » afin de supprimer cette dernière.

2.6 Conclusion

Ce chapitre constitue le lancement de la phase de développement, menant une étude fonctionnelle de notre plateforme de mise en relation artisans/ particuliers. Cette étude nous a permis de recenser les besoins fonctionnels et non fonctionnels ainsi que les acteurs de notre système. Dans le chapitre suivant, nous présentons la description opérationnelle permettant les réalisations des fonctionnalités offertes par ce système.

III. Conception

3.1 Conception en général

Avant de présenter la conception de notre application de mise en relation artisans/ particuliers, nous exposons brièvement dans cette section certains paradigmes liés à la conception d'applications mobiles développées sous le l'environnement Flutter [18] et destinées à fonctionner sous le système Android et IOS.

3.1.1 Conception architecturale

Il est primordial à la conception de tout système informatique de choisir le modèle d'architecture adéquat pour permettre un fonctionnement correct, des performances meilleurs, la réutilisabilité des modules et l'interopérabilité avec d'autres systèmes. C'est à cet effet que nous avons opté pour le modèle dit "Modèle, Vue et Contrôleur" MVC [19] qui sera également très pratique pour gérer l'interaction entre les différents composants de notre application Android. Nous décrivons cette architecture dans la section suivante.

3.1.2 Approche Orientée Objet

La programmation orientée objet est définie comme un paradigme de programmation informatique basé sur l'interaction de briques appelées objets, qui sont eux-mêmes des abstractions d'objets réels, via leurs relations et les propriétés qui leur sont accordées. Ce paradigme assure:

- Une modularité des programmes résolvant le problème de la complexité des codes en le structurant sous forme de classes qui seront utilisées dans les différentes parties du code afin d'éviter les répétitions.
- Une plus grande vitesse d'exécution et cela en chargeant une seule fois l'objet dans la mémoire de la machine et qui sera utilisé pendant toute l'exécution de l'application, et cela permet de diminuer le nombre de lectures et d'écritures, et aussi le nombre de requêtes au serveur.
- Une souplesse de réutilisation et d'évolution du code grâce à la facilité de repérer les classes concernées par la modification, et les changements seront effectués seulement sur ces classes sans réécrire les mêmes modifications pour chaque partie modifiée.

3.2 Conception détaillée

Dans cette section, nous présentons les détails de la conception de notre application de mise en relation artisans/ particuliers. Pour mener à bien notre conception, nous

avons utilisé les diagrammes de séquences et de classes du langage UML pour la modélisation des traitements de notre système. Par-ailleurs, au regard de sa simplicité et sa puissance démonstrative, nous avons adopté le modèle entité/association pour la construction du modèle conceptuel de données correspondant à notre application de mise en ligne.

3.2.1 Diagrammes de séquences

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

Dans le contexte de notre application de mise en relation artisans/ particuliers, nous présentons ici les diagrammes de séquences correspondant aux cas d'utilisation représentant les fonctionnalités principales de notre système.

Diagramme de séquence : cas d'utilisation inscription "visiteur client"

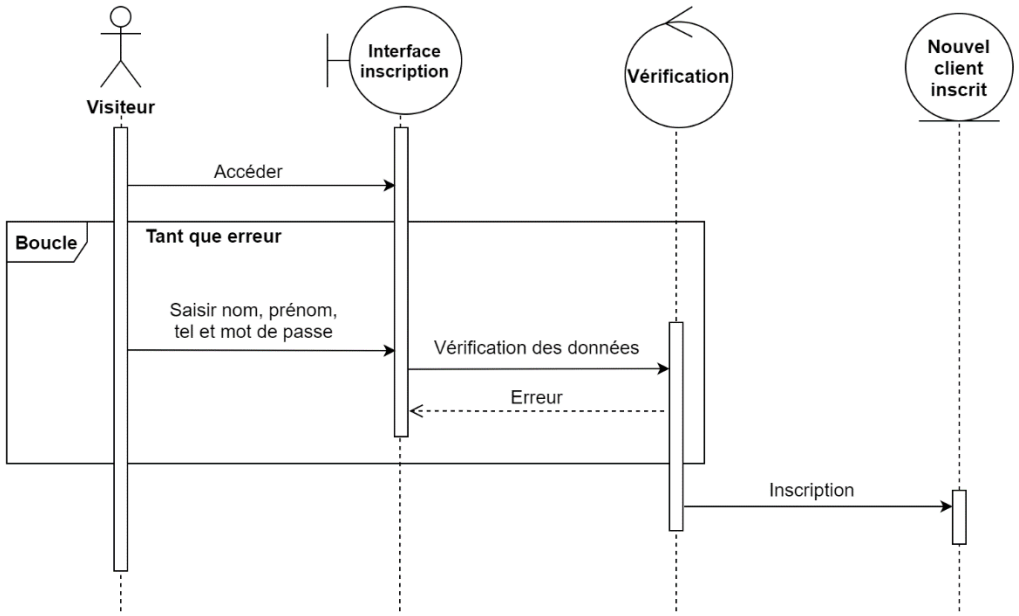

Figure III. 1 Diagramme de séquence cas d'utilisation inscription "visiteur/client"

Un visiteur souhaitant s'inscrire en tant que client accède à l'espace inscription client. Un formulaire portant les informations : nom, prénom, numéro de téléphone et le mot de passe doit être rempli et validé, Le système vérifie les informations communiquées, Une fois que tout est vérifié un nouveau client sera créé dans la base de données.

Diagramme de séquence : cas d'utilisation inscription "visiteur artisan"

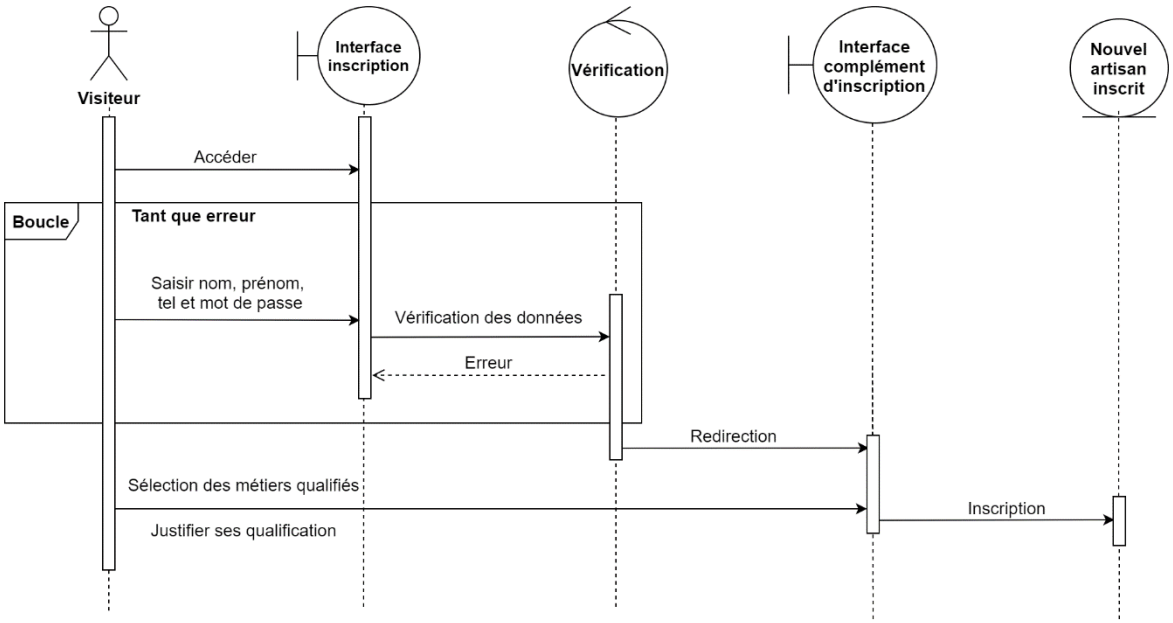

Figure III. 2 Diagramme de séquence cas d'utilisation d'inscription d'un artisan

Un visiteur souhaitant s'inscrire en tant que artisan accède à l'espace inscription artisan. Un formulaire portant les informations : nom, prénom, numéro de téléphone et le mot de passe doit être rempli et validé. Le système vérifie les informations communiquées et le redirige vers la page complément d'inscription pour fournir ses qualifications et son emplacement. Une fois que tout est vérifié un nouvel artisan sera créé dans la base de données.

Diagramme de séquence : cas d'utilisation poster une annonce "client"

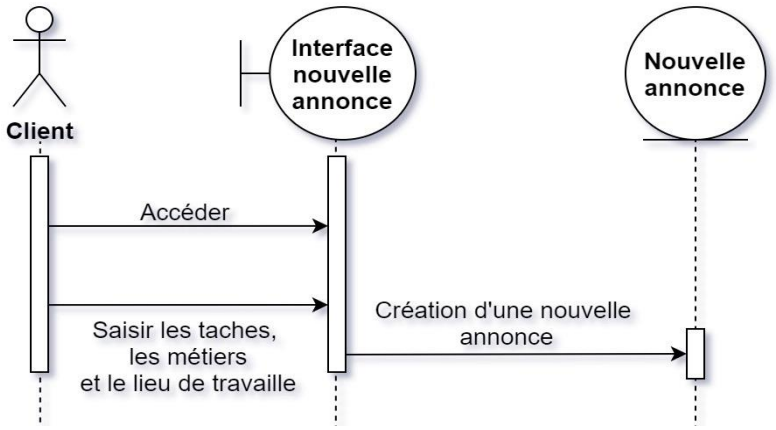

Figure III. 3 Diagramme de séquence cas insertion d'une annonce

Le client accède à l'interface qui permet de poster une annonce, ensuite spécifie les tâches à accomplir, les métiers concernés et l'endroit où se trouve le chantier.

Une fois que le client clique sur le bouton "poster", une nouvelle annonce est créée dans la base de données. Ainsi le système notifiera les artisans concernés par l'annonce à travers une notification. Ce travail sera proposé pour ces artisans en l'affichant sur leur page d'accueil.

Diagramme de séquence : cas d'utilisation consulter et valider un devis "client"

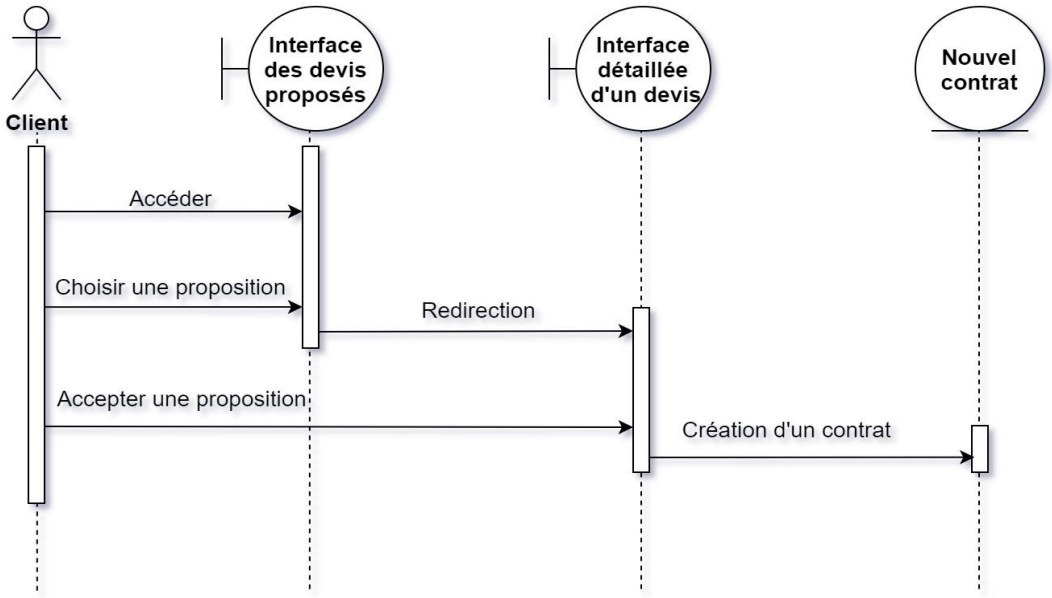

Figure III. 4 Diagramme de séquence cas d'acceptation d'un devis

Le client accède à la page des devis proposés suite à une annonce qu'il a posté pour consulter les détails de chaque devis.

Après consultation, le client valide le devis qui lui semble le plus intéressant. Un contrat est créer dans la base de données et l'artisan concerné sera notifié.

Diagramme de séquence : cas d'utilisation consulter les annonces "artisan"

L'artisan accède à l'interface annonces ou se trouve toute les annonces qui portent les métiers qui le concernent. L'artisan peut effectuer une recherche sur cette interface en choisissant l'emplacement et les métiers des annonces qu'il veut voir, ainsi un filtrage sera effectué sur les annonce à rechercher et seulement celles qui portent l'emplacement et les métiers fournis qui lui seront affichées comme résultat de recherche.

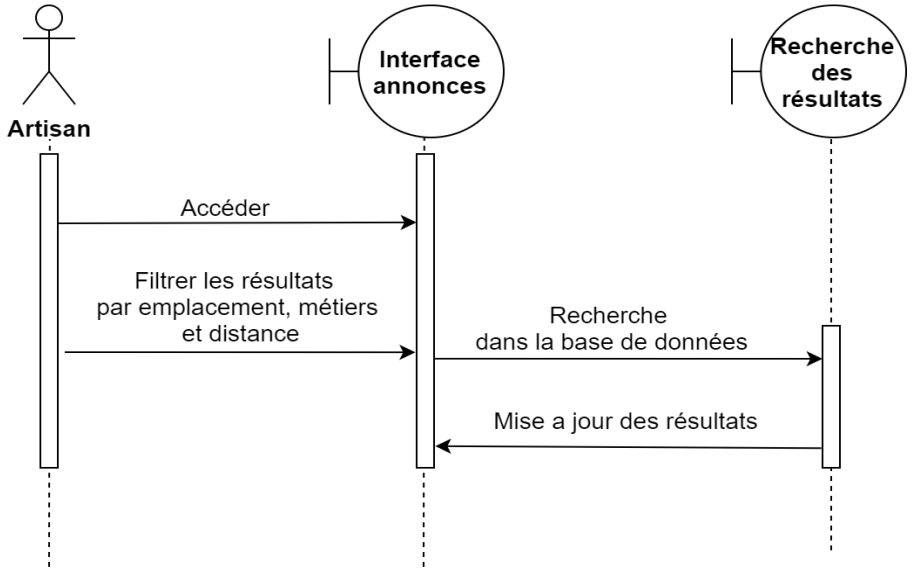

Figure III. 5 Diagramme de séquence : cas d'utilisation consulter les annonces

Diagramme de séquence : cas d'utilisation soumettre un devis "artisan"

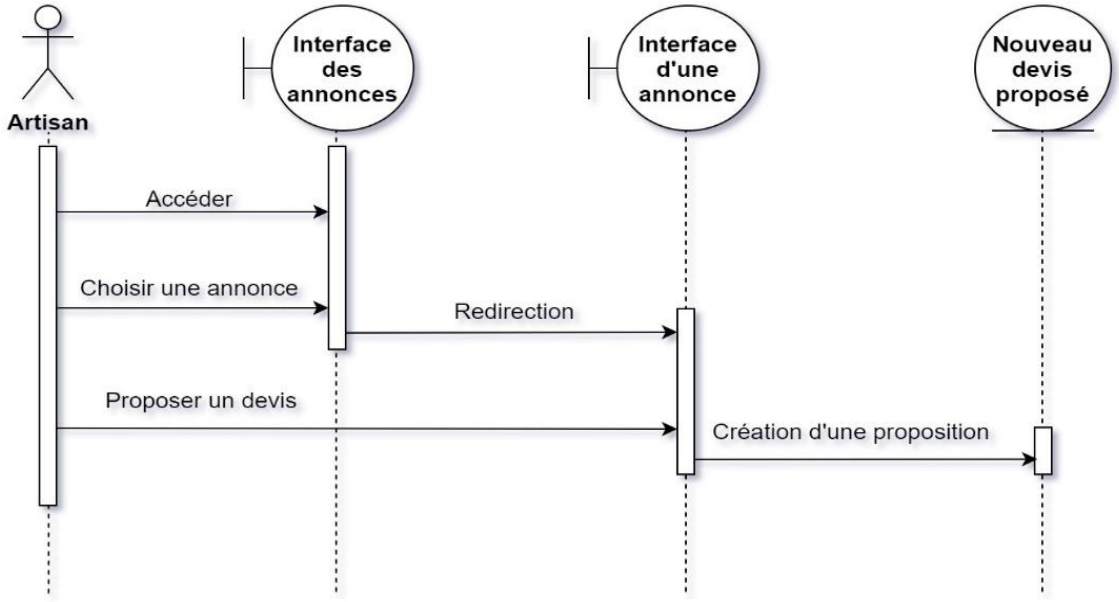

Figure III. 6 Diagramme de séquence, cas d'utilisation soumettre un devis

L'artisan accède à la page des annonces, puis pour chaque annonce, consulte son détail avant de soumettre un devis. Une fois soumis, un devis est créé dans la base de données et le client concerné sera notifié.

Diagramme de séquence : approuver une inscription "administrateur"

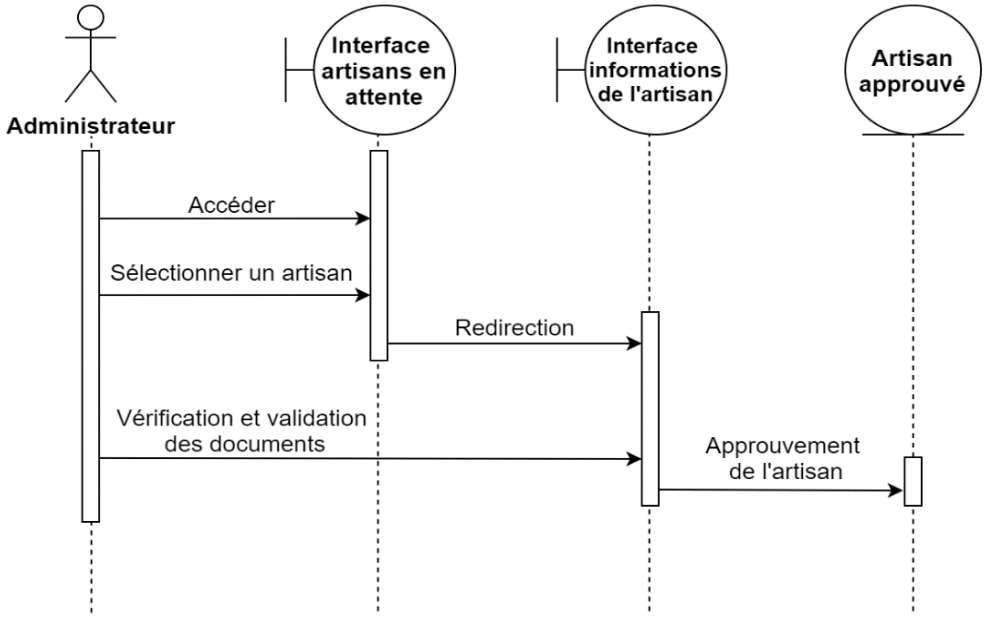

Figure III. 7 Diagramme de séquence cas d'approbation d'un artisan

L'administrateur accède à l'interface artisans en attente et choisit un artisan pour consulter ses informations, ses métiers et ses justificatifs. Une fois consultés, l'administrateur approuve l'artisan si toutes les informations fournis sont adéquates. Une fois l'artisan est approuvé son état sera actif dans la base de données.

3.2.2 Diagrammes de classes

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les cas d'utilisation ne réalisent donc pas une partition des classes du diagramme de classes. Un diagramme de classes n'est donc pas adapté (sauf cas particulier) pour détailler, décomposer, ou illustrer la réalisation d'un cas d'utilisation particulier.

Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du

domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de Programmation orientée objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

Diagramme de classe : cas inscription en tant que client

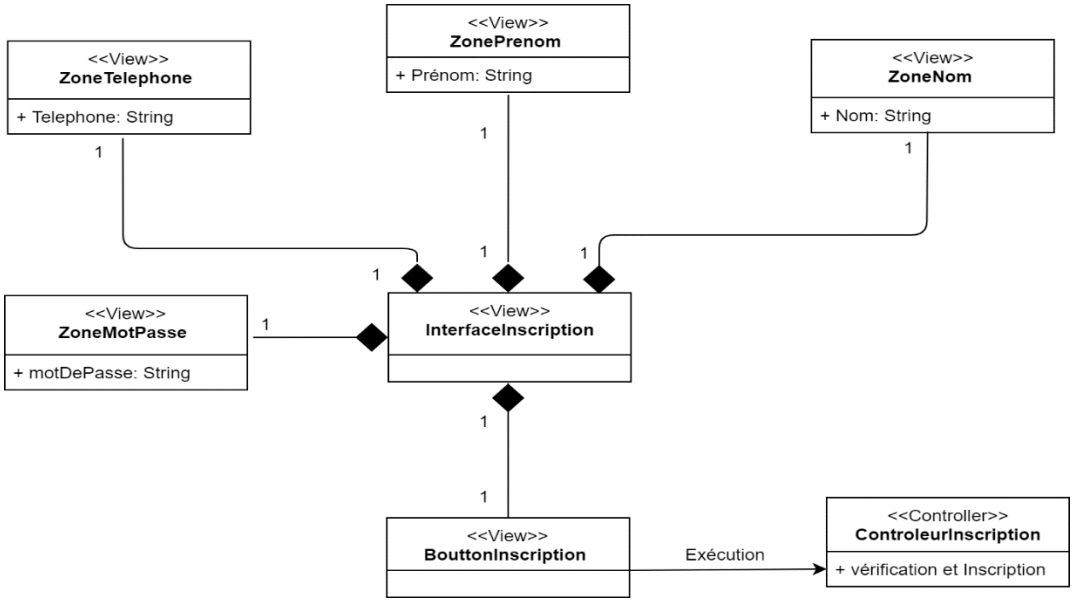

Figure III. 8 Diagrammes de classes cas d'inscription au tant que client

Dans l'interface inscription côté client de notre application nous avons les classes suivantes :

- **InterfaceInscription** est l'interface principale
- **ZoneNom**, **ZonePrenom**, **ZoneMotPasse** et **ZoneTelephone** représentent les champs de saisie des informations du client.
- **BouttonInscription** exécute **ControleurInscription** en cliquant dessus afin d'effectuer une vérification et insérer un nouveau client dans la base de données à travers ce dernier.

Diagramme de classe : cas inscription en tant que "artisan"

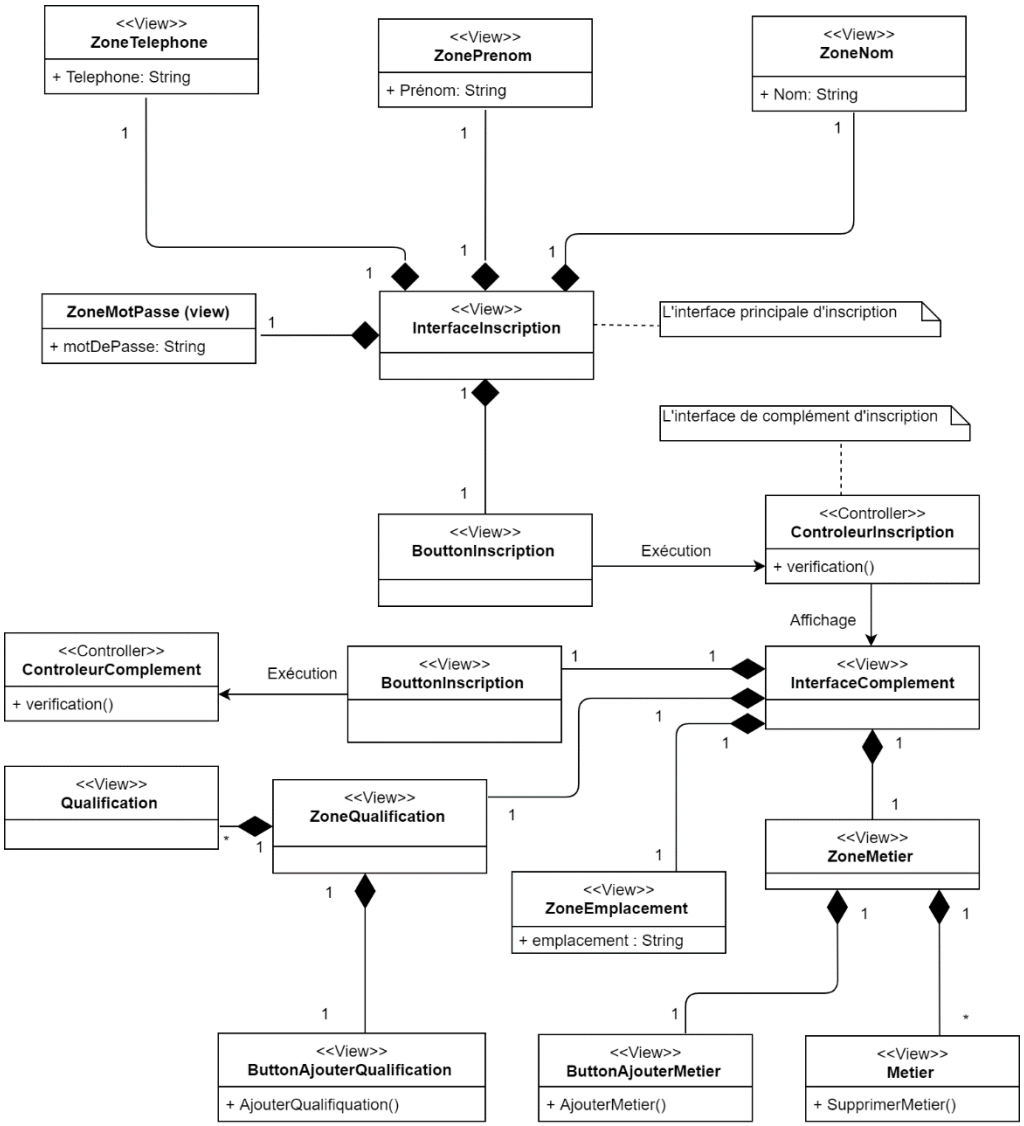

Figure III. 9 Diagramme de classes cas d'inscription d'un client

Dans l'interface inscription côté artisan de notre application nous avons les classes suivantes :

- **InterfaceInscription** est l'interface principale
- **ZoneNom**, **ZonePrenom**, **ZoneMotPasse** et **ZoneTelephone** qui représentent les champs de saisie des informations de l'artisan.
- **BouttonInscription** exécute **ControleurInscription** en cliquant dessus afin d'effectuer une vérification et rediriger l'utilisateur vers l'interface complément inscription.
- **InterfaceComplement** représente l'interface complément de l'inscription et qui se compose des classes : **ZoneMetier**, **ZoneQualification** et

ZoneEmplacement permettent respectivement de fournir les métiers maitrisés par l'artisan, les justificatifs de ces métiers sous forme de documents, et l'emplacement de l'artisan

- **BoutonAjouterMetier** représente le bouton qui permet d'ajouter un métier représenté par la classe **Metier** à la liste des métiers maitrisés par l'artisan.
- **BoutonAjouterQualification** représente le bouton qui permet d'ajouter un justificatif de qualification sous forme d'un document.

Diagramme de classe : cas approbation d'un devis par un client

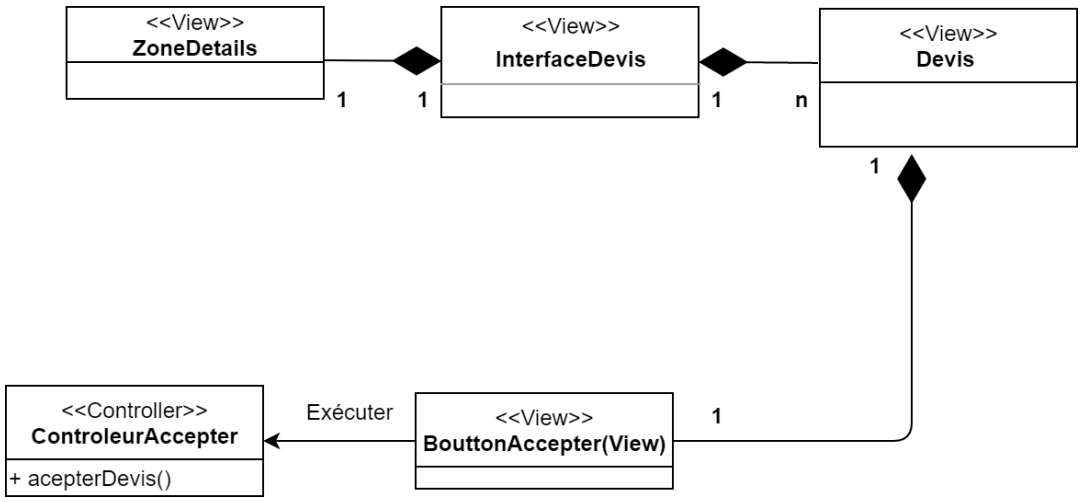

Figure III. 10 Diagramme de classes cas d'acceptation d'un devis

Dans l'interface acceptation d'un devis de notre application client nous avons les classes suivantes:

- **InterfaceDevis** représente l'interface principale ou se trouve tous les devis proposés par les artisans pour une annonce postée par le client.
- **Devis** représente un devis.
- **ZoneDetail** représente la partie détaillée du devis.
- **BouttonAccepter** représente le bouton qui permet d'accepter un devis.
- **ControlleurAccepter** est le contrôleur qui vérifie et exécute l'action de **BouttonAccepter**.

Diagramme de classe : cas poster une annonce par un client

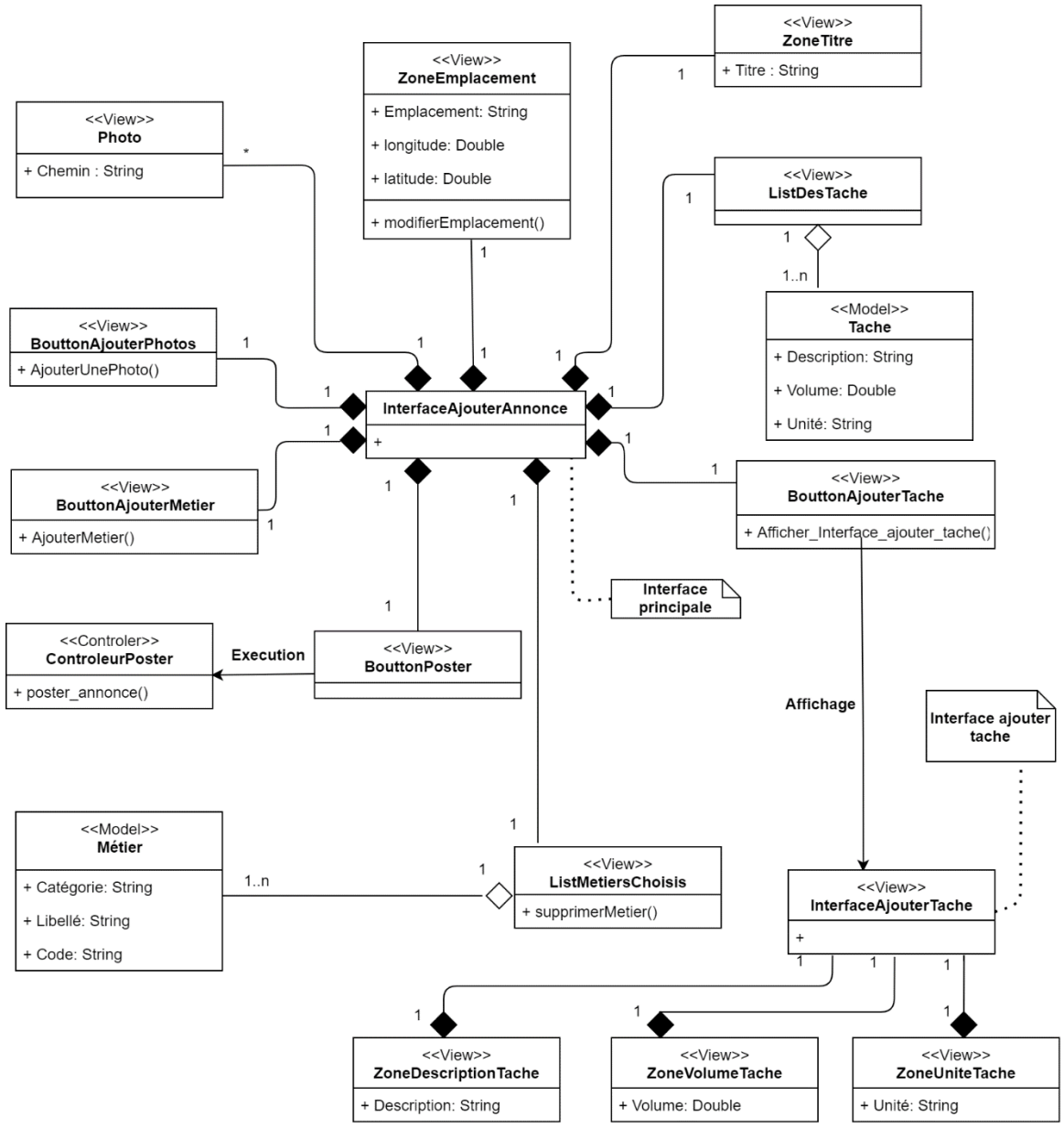

Figure III. 11 Diagramme de classes cas poster une annonce

Dans l'interface ajouter annonce de notre application côté client nous avons la classe qui représente l'interface principale intitulée **InterfaceAjouterAnnonce** et qui se compose de classes suivantes :

- **ZoneEmplacement**, **ZoneTitre**, **ListDesMetiers**, **ListDesTache** et **ListPhotos** qui permettent de fournir les informations nécessaires à propos de l'annonce.

- **Tache** et **Metier** qui forment respectivement les classes **ListDesTache** et **ListDesMetier**.
- **BoutonAjouterMetier** qui permet d'ajouter un métier à l'annonce en créant un objet de la classe **Metier**.
- **BoutonAjouterTache** qui envois vers la classe **InterfaceAjouterTache** qui affiche l'interface **AjouterTache** et qui permet de fournir les informations nécessaires à propos de la tâche à ajouter à travers les classes : **ZoneDescription** et **ZoneVolume** ainsi que **ZoneUnite** et créer un objet de la classe Tache.
- **BoutonAjouterPhoto** permet d'ajouter une photo à l'annonce.
- **BoutonValider** exécute **ControleurValider** afin de vérifier et de créer un objet de type Annonce et l'insérer à la base de données.

Diagramme de classe : cas consulter une annonce par un artisan

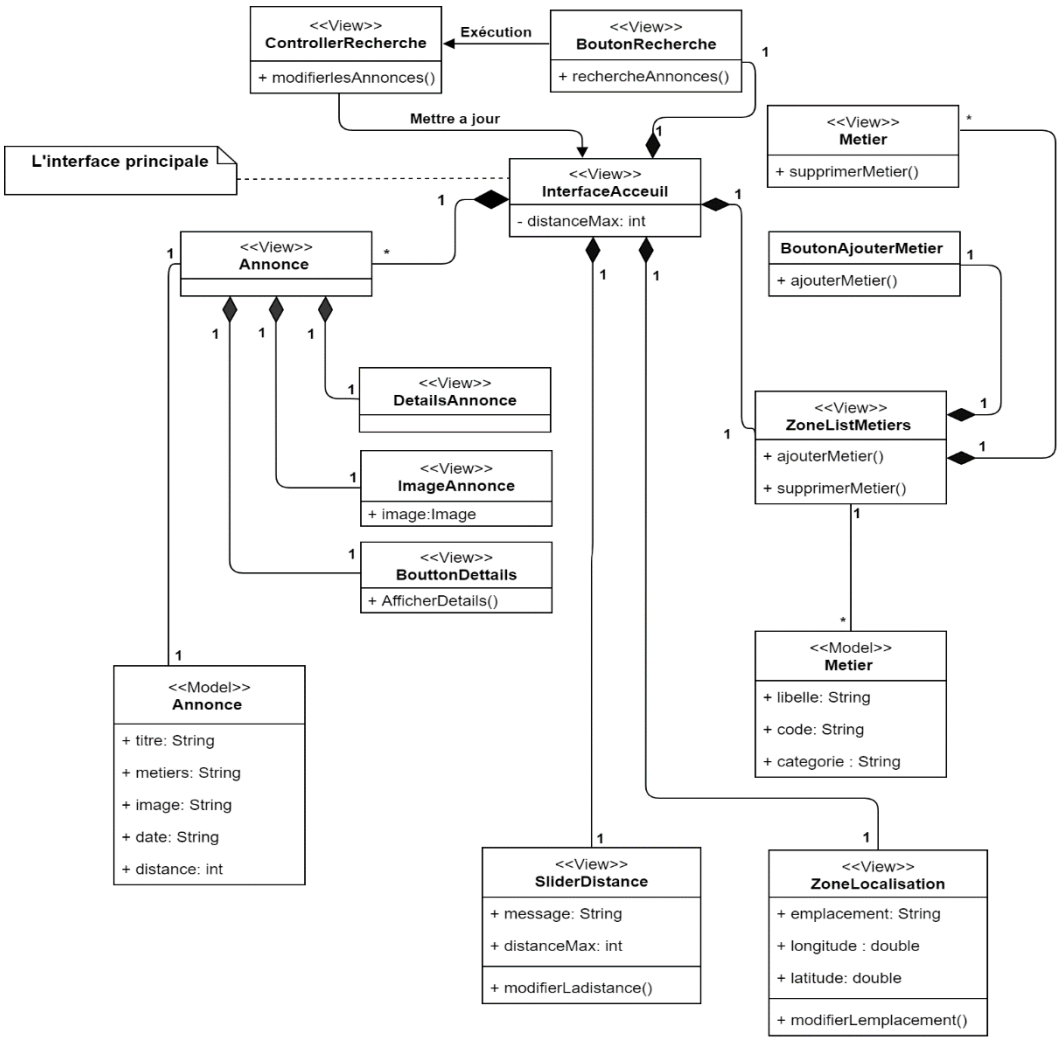

Figure III. 12 Diagramme de classes cas de recherche des annonces

Dans l'interface accueil de notre application côté artisan nous avons la classe **InterfaceAccueil** comme classe principale et qui représente l'interface d'accueil de l'application et celle-ci se compose d'autres classes qui sont :

- **Annonce** afin d'avoir plusieurs objets de type Annonce et que elle-même utilise quatre autres classes afin de fournir des informations à propos de l'annonce et qui sont : Annonce(Model), **DetailAnnonce** pour fournir les détails de l'annonce, **ImageAnnonce** pour afficher l'image principale de l'annonce et **BoutonDetail** pour accéder aux détails de l'annonce.
- **ZoneListMetiers** représente la liste des métiers présents dans l'annonce et qui utilise les deux classes Metier et **BoutonAjouterMetier** afin d'ajouter et de supprimer un métier.
- **ZoneLocalisation** permet d'indiquer la localisation de l'annonce.
- **Slider** indique la distance ou se situe le lieu de l'annonce.
- **BoutonRecherche** exécute **ControlerRecherche** afin de mettre à jour l'interface d'accueil.

Diagramme de classe : cas soumettre un devis par un artisan

Dans l'interface Annonce de notre application côté artisan nous avons la classe InterfaceAnnonce comme classe principale et celle-ci se compose d'autres classes qui sont :

- **DetailAnnonce** est la vue détaillée de l'annonce.
- **CarouselPhoto** représente le slide des photos de l'annonce et elle-même se compose d'une autre classe qui est la classe Photo.
- **BoutonProposer** renvoie vers la classe **InterfaceProposerDevis** qui affiche l'interface permettant de proposer un devis et elle-même se compose des classes **ZoneProposerMontant** afin de saisir le montant proposé et **BoutonProposer** qui exécute **ControlerProposer** afin de créer un objet de la classe devis et l'insérer dans la base de données.

Notre interface **InterfaceAnnonce** contient aussi des objets des classes **Metier**, **Tache** et **Photo** qui sont utilisées pour afficher les détails des annonces pour l'artisan.

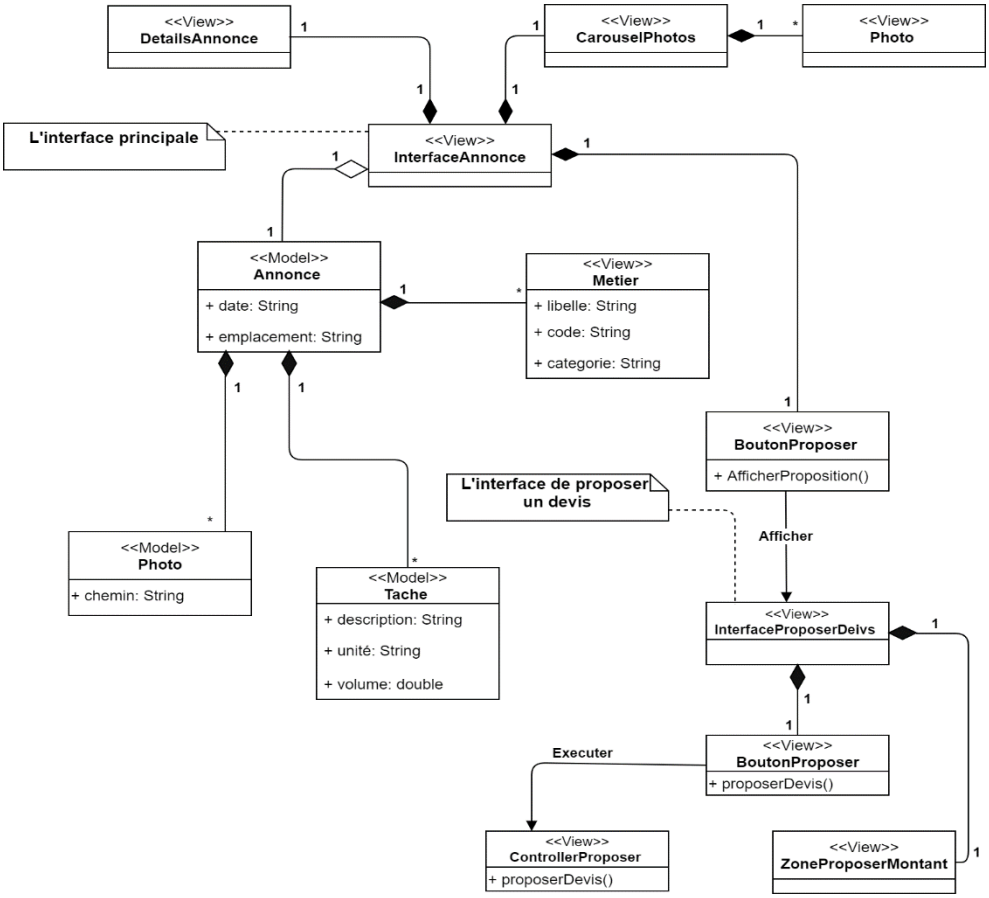

Figure III. 13 Diagramme de classes cas de proposer un devis

Diagramme de classe : approuver un artisan par un l'administrateur

Dans l'espace administration, nous avons la classe **InterfaceArtisanAttente** comme classe principale qui se compose d'autres classes qui sont :

- **TableauArtisanAttente** qui représente le tableau des artisans en attente d'approbation.
- **LigneArtisan** qui forme la Classe **TableauArtisanAttente** et qui se compose des deux classes **Artisan** et **BoutonInformations**.
- **BoutonInformations** affiche l'interface des informations et des documents de l'artisan.
- **InterfaceInformation** représente l'interface des informations et des documents de l'artisan et qui se compose des quatre classes suivantes : **ListMetier**, **Document**, **BoutonApprouver** et **BoutonDesapprouver**.

– Les deux classes **BoutonApprouver** et **BoutonDesapprouver** qui représentent respectivement le bouton approuver qui permettent d'approuver ou rejeter l'artisan.

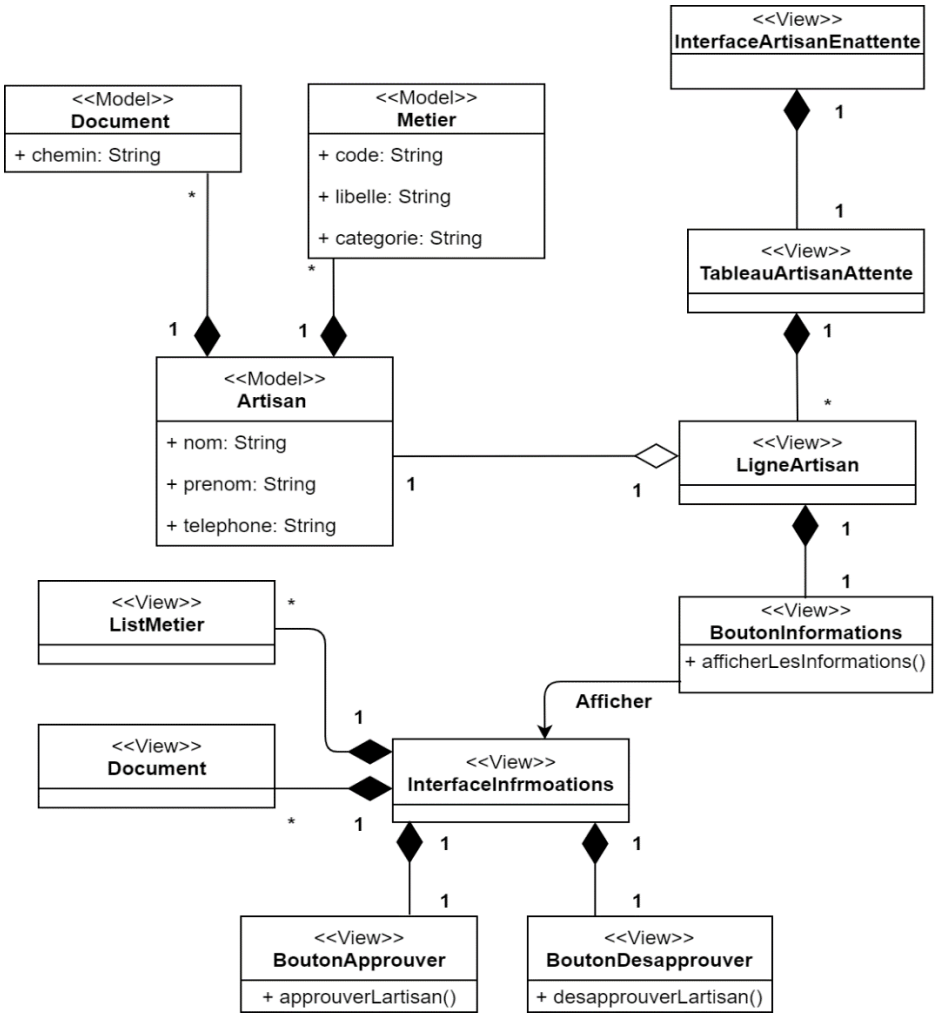

Figure III. 14 Diagrammes de classes cas d'approbation d'un artisan

3.2.3 Modèle conceptuel de données associé à notre système

Pour le modèle conceptuel de données MCD correspondant à notre application de mise en relation artisans/ particuliers, nous avons choisi le modèle entité-association de la démarche conceptuelle MERISE [20]. Notre choix est motivé par sa simplicité à comprendre les objets manipulés dans le système ainsi que les liens existants entre ces derniers.

Entités

Une entité est la représentation d'un élément matériel ou immatériel ayant un rôle dans le système que l'on désire décrire.

Dans le cas de notre application, nous avons recensé les entités suivantes :

- *ARTISAN*
- *CLIENT*
- *METIER*
- *ANNONCE*
- *TACHE*
- *DEVIS*
- *CONTRAT*
- *REALISATION*
- *PHOTO_REALISATION*
- *PHOTO_ANNONCE*

Associations et cardinalités

Une association représente les liens sémantiques qui peuvent exister entre plusieurs entités.

Les cardinalités permettent de caractériser le lien qui existe entre une entité et la relation à laquelle elle est reliée. La cardinalité d'une relation est composée d'un couple comportant une borne maximale et une borne minimale, interval dans lequel la contrainte d'une entité peut prendre sa valeur.

Dans le cas de notre application, nous avons identifié les associations et les cardinalités suivantes :

1. *QUALIFIER* entre : *ARTISAN* et *METIER*
   - Un artisan est qualifié par 1 ou plusieurs métiers.
   - Un métier peut être une qualification de 0 ou plusieurs artisans.

2. *REALISER* entre *: ARTISAN* et *REALISATION*
   - Un artisan peut avoir réalisé 0 où plusieurs réalisations.
   - Une réalisation est réalisée par un et un seul artisan.

3. *PROPOSER* entre *: ARTISAN et DEVIS*
   - Un artisan peut proposer 0 ou plusieurs devis.

- Un devis est proposé par un seul artisan.

4. *POSTER* entre *: CLIENT et ANNONCE*
   - Un client peut poster 0 ou plusieurs annonces.
   - Une annonce est postée par un et un seul client.

5. *CONSULTER* entre*: CLIENT et REALISATION*
   - Un client peut consulter 0 ou plusieurs réalisations
   - Une réalisation peut être consultée par 0 ou plusieurs clients.

6. *IMPLIQUER* entre *: ANNONCE et METIER*
   - Une annonce implique un ou plusieurs métiers.
   - Un métier est impliqué dans 0 ou plusieurs annonces.

7. *CONSISTER* entre*: ANNONCE et TACHE*
   - Une annonce consiste en une ou plusieurs taches.
   - Une tache est constituée par une et une seule annonce

8. *CONERNER* entre*: DEVIS et TACHE*
   - Un devis concerne une ou plusieurs taches.
   - Une tache est concernée par 0 ou plusieurs devis.

9. *CONCLURE* entre *: DEVIS et CONTRAT*
   - Un devis peut conclure 0 ou 1 contrat.
   - Un contrat peut être conclu à travers un et un seul devis.

10. *ASSOCIER* entre *: REALISATION et PHOTO_REALISATION*
    - Une photo est associée à une et une seule réalisation.
    - Une réalisation possède 0 ou plusieurs photos.

11. *ATTRIBUER* entre *: ANNONCE et PHOTO_ANNONCE*
    - Une photo est attribuée à une et une seule annonce.
    - Une annonce possède 0 ou plusieurs photos.

Schéma du modèle entité-association

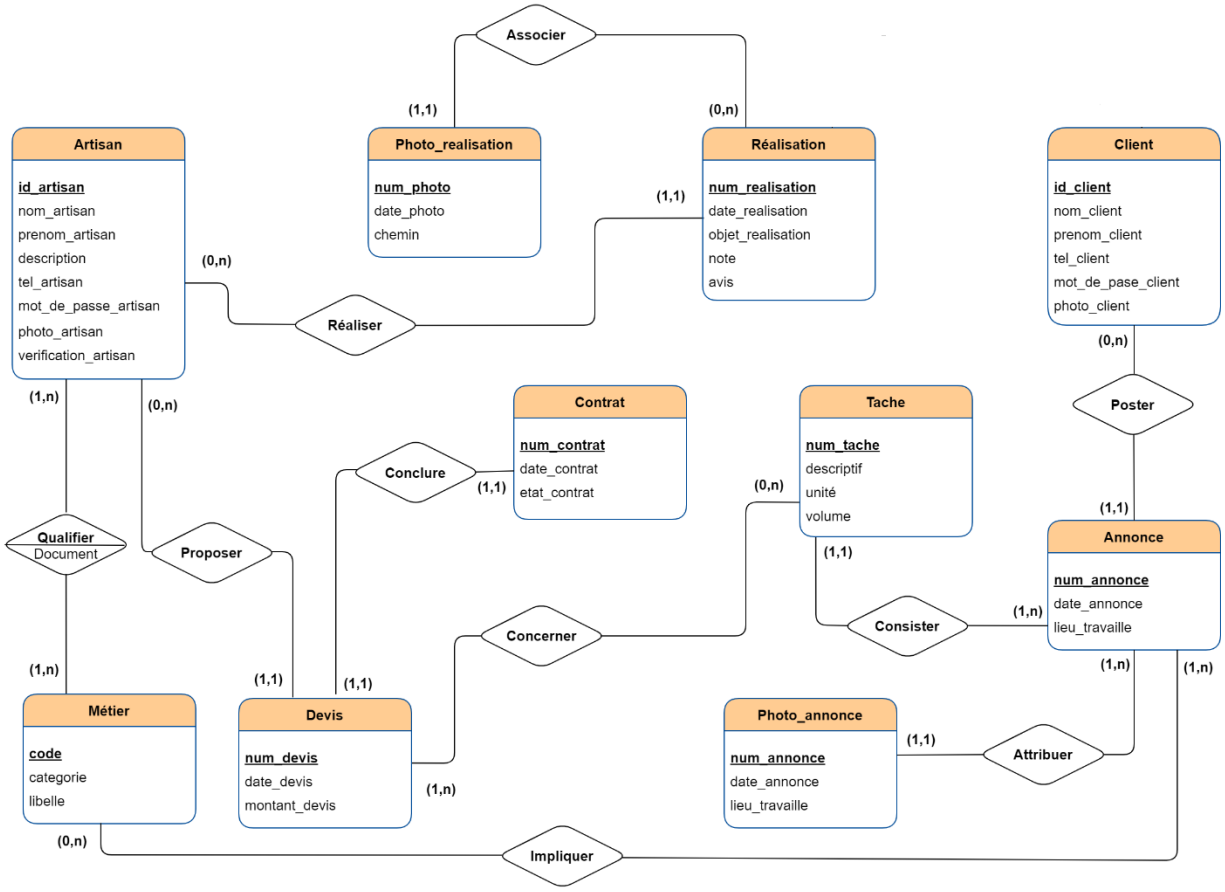

Figure III. 15 Modèle conceptuel de données

3.2.4 Modèle logique de données résultant

C'est la transformation de MCD en relationnel, et cela permet de modéliser la structure selon laquelle les données seront stockées dans la base de données.

- ***Client*** *(**id_client**, nom_client, prenom_client, tel_client, mot_de_passe_client,photo_client)*
- ***Artisan(id_artisan,*** *nom_artisan, prenom_artisan, description, tel_artisan, mot_de_passe_artisan, photo_artisan,lieu_artisan, longitude_artisan, latitude_artisan, verification_artisan)*
- ***Annonce*** *(**num_annonce**, **#id_client**,titre_annonce ,date_annonce, lieu_travaille, longitude_annonce, latitude_annonce)*
- ***Metier*** *(**code**, categorie, libelle)*
- ***Metier_artisan*** *(**#id_artisan, #code,** qualification)*
- ***Metier_annonce*** *(**#num_annonce, #code**)*

- ***Tache*** *(**num_tache**, **#num_annonce**, descriptif, unite, volume)*
- ***Devis*** *(**Num_devis**, **#id_artisan, #num_contrat**, date_devis, montant_devis,duree_devis)*
- ***Devis_tache(#num_devis,#num_tache)***
- ***Contrat*** *(**num_contrat**, **#num_devis**, date_contrat,etat_contrat)*
- ***Realisation*** *(**num_realisation**, **#id_artisan**, date_realisation, objet_realisation, note, avis)*
- ***Photo_realisation*** *(**num_photo**, **#num_realisation**, chemin)*
- ***Photo_annonce*** *(**num_photo**, **#num_annonce**, chemin)*

3.3 Conclusion

Dans ce chapitre, nous avons présenté la partie conception de notre application mobile de mise en relation artisans/ particuliers en adoptant les diagrammes d'UML pour la modélisation de l'aspect dynamique de notre systèmes, et l'outil entité/ association de la démarche MERISE pour la modélisation de la partie statique à savoir les données manipulées par l'application. Dans la suite de ce mémoire, nous allons présenter les détails de l'implémentation ainsi que des exemples d'interfaces de notre plateforme de mise en relation.

IV. Réalisation

4.1 Vue d'ensemble

Dans ce chapitre nous allons exposer le travail achevé. Nous présenterons en premier lieu l'environnement matériel et logiciel dans lesquels notre application a été développée en indiquant les technologies utilisées. Nous clôturons ce chapitre par un aperçu sur l'état d'avancement du projet et sur quelques captures d'écran traduisant le déroulement de l'application.

4.2 Environnement matériel

Pour la réalisation de notre application, nous avons travaillé sur une machine dont la configuration est la suivante :

- **Processeur :** Intel Core i5-7300U 2.7 GHz
- **Mémoire :** 8 Go DDR4
- **Disque dur :** Samsung SSD 256 Go

4.3 Environnement logiciel

Pour l'environnement logiciel, nous avons travaillé sous le système d'exploitation Windows 10 et un éditeur de code Visual Studio Code sur lequel nous avons installé le plugin Flutter. Notons que le langage de programmation utilisé est le Dart.

Dans ce qui suit, nous présentons en détail toutes les technologies logicielles utilisées pour la réalisation de notre application.

4.3.1 Flutter

Flutter est un framework open source créé par Google pour la réalisation d'applications multiplateformes. Il permet donc de construire en une seule fois des applications qui pourront être déployées pour différentes plateformes (Android/iOS), et même plus avec l'arrivée de Flutter pour web et desktop. En soit, Flutter pourrait fonctionner sur n'importe quel device possédant un écran plus ou moins intelligent. La première version a été réalisée en Décembre 2018 ce qui en fait un framework encore jeune. Les principales composantes de Flutter sont présentées dans ce qui suit :

Le language Dart

Google a créé de nombreux langages de programmation, dont Dart [21], il commençait à tomber dans l'oubli. Mais depuis l'émergence de Flutter, ce langage retrouve des couleurs et a même connu une version 2.0.

Si Google a opté pour Dart, c'est parce qu'il offre deux modes de fonctionnement. Le premier, nommé AOT (pour Ahead Of Time), permet de générer une application native pour chaque plateforme. L'avantage de Flutter, par rapport à ses concurrents, est donc fort, puisque le code sera optimisé directement pour l'architecture sur laquelle il fonctionnera.

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 ce problème en ne mettant plus que quelques millisecondes entre chaque modification. Le développement de son application est alors bien plus rapide.

Outre ces deux modes, Dart est aussi très performant pour gérer l'allocation et le "garbage collector", c'est-à-dire l'élimination des objets en mémoire qui ne sont plus utilisés.

```dart
import 'Metier.dart';

class Annonce {
  int numAnnonce, numClient;
  String date;
  String emplacement;
  String titre;
  List<String> images;
  List<Metier> metiers;
  List<Tache> taches;

  bool favorite;
  Annonce();

  Annonce.con(this.numAnnonce, this.numClient, this.titre, this.date,
      this.emplacement, this.taches, this.metiers, this.images);

}
```

Figure IV. 1 Le langage Dart

Moteur Flutter

Le moteur de Flutter, écrit principalement en C++, fournit un support de rendu de bas niveau en utilisant la bibliothèque graphique Skia de Google. De plus, il s'interface avec des SDK spécifiques à la plate-forme, tels que ceux fournis par Android et iOS. Le moteur Flutter est un runtime portable pour l'hébergement d'applications Flutter. Il met en œuvre les bibliothèques de base de Flutter, y compris l'animation et les graphiques, les entrées/sorties de fichiers et de réseau, le support de l'accessibilité, l'architecture des

plugins, et une chaîne d'outils d'exécution et de compilation Dart. La plupart des développeurs interagissent avec Flutter via le Flutter Framework, qui fournit un cadre réactif, et un ensemble de widgets de plate-forme, de mise en page et de foundation.

Bibliothèque Foundation

La bibliothèque Foundation, écrite en Dart, fournit des classes et des fonctions de base qui sont utilisées pour construire des applications utilisant Flutter, comme une API pour communiquer avec le moteur.

Widgets spécifiques à la conception

Le Framework Flutter contient deux ensembles de widgets qui sont conformes à des langages de conception (design) spécifiques. Les widgets Material Design implémentent le langage de conception de Google du même nom, et les widgets Cupertino implémentent les directives de l'interface humaine iOS d'Apple.

4.3.2 Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code et Git intégré.

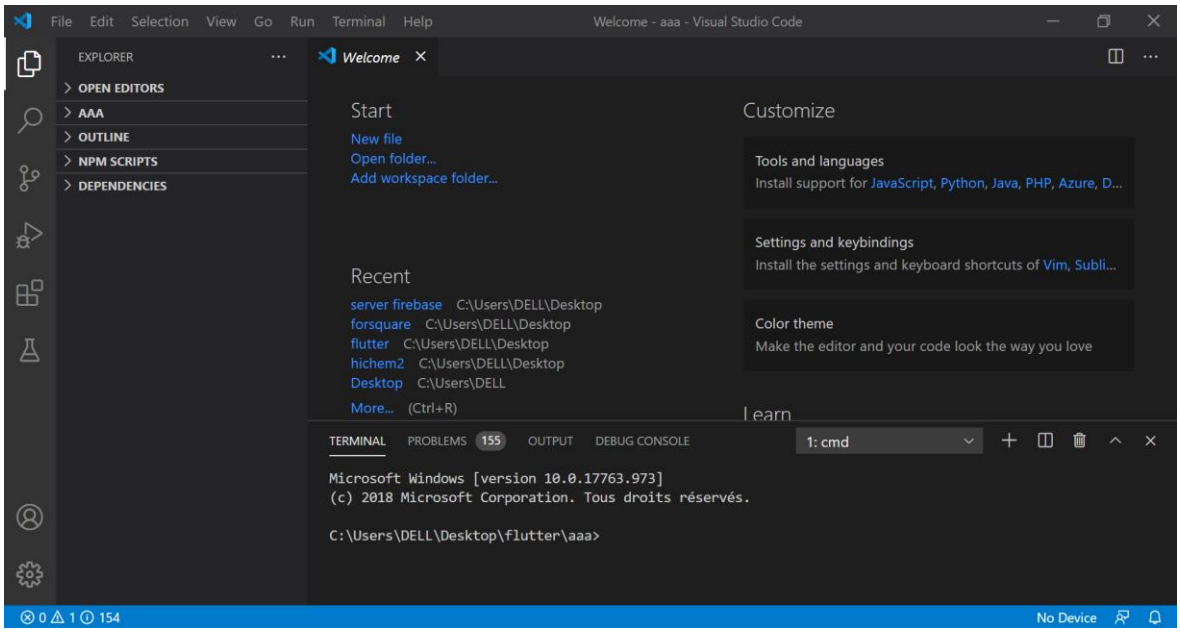

Figure IV. 2 Visual Studio Code

4.3.3 Plugin Dart pour VS Code

Pour coder notre application, nous avons installé le plugin Dart qui rajoutera à VS Code les fonctionnalités spécialisées dans le développement avec le langage Dart.

4.3.4 MySQL

MySQL [22] est un langage de manipulation de base de données relationnelle (SGBDR), apparue le 23 mai 1995, est l'œuvre d'une société suédoise MySQL AB, fondée par **David Axmark**, **Allan Larsson** et **Michael Widenius**. Est très employée sur le Web, souvent en association avec *PHP* (langage) et *Apache* (serveur web). MySQL fonctionne indifféremment sur tous les systèmes d'exploitation *(Windows, Linux, Mac OS notamment)*, il permet notamment :

- **La manipulation de base :** création, suppression et la copie de la base de données
- **La manipulation des tables :** création, suppression, modification de la structure des tables.
- **La manipulation des données** : insertion, sélection, modification et suppression d'enregistrements.

```
SELECT *,GROUP_CONCAT(DISTINCT libelle SEPARATOR ' ,') as metiers,
COALESCE( chemin ) as photo
FROM annonce, metier,metier_annonce,photo_annonce
where metier.code=metier_annonce.code
AND metier_annonce.num_annonce=annonce.num_annonce
AND photo_annonce.num_annonce=annonce.num_annonce
AND annonce.longitude>$minlong AND annonce.longitude<$maxlong
AND annonce.latitude<$maxlat AND annonce.latitude>$minlat
GROUP BY annonce.num_annonce
```

Figure IV. 3 Requête MySQL

4.3.5 Serveur Apache

Apache [23] s'agit d'une application qui fonctionne a la base sur les systèmes d'exploitation de type UNIX mais il a désormais été porté sur de nombreux systèmes, dont Microsoft Windows grâce à sa conception modulaire (nombreux de code) qui corresponds à différentes aspects ou fonctions de serveur.

4.3.6 PHP

PHP [24] est conçu en1994 par RasmusLerdorf Est un langage de programmation libre, principalement utilisé pour produire des pages web dynamique via un serveur. On l'a utilisé pour acceder à notre base de données à partir de nos applications en utilisant un intermédiaire qui est le vecteur JSON.

4.3.7 JSON

JSON [25] (JavaScript Objet Notation) est un langage léger d'échange de données textuelles. Pour les ordinateurs, ce format se génère et s'analyse facilement. Pour les humains, il est pratique à écrire et à lire grâce à une syntaxe simple et à une structure en arborescence. JSON permet de représenter des données structurées (comme XML par exemple).

Figure IV. 4 Exemple JSON

4.3.8 Interfaces de programmation utilisées

En informatique, API [26] est l'acronyme de *Application Programming Interface*, que l'on traduit en français par interface de programmation applicative ou interface de programmation d'application. L'API peut être résumée à une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données. Il s'agit en réalité d'un ensemble de fonctions qui facilitent, via un langage de programmation, l'accès aux services d'une application.

Firebase Authentification

C'est un API créé par Google, qui nous permet d'utiliser l'authentification Firebase pour connecter un utilisateur en envoyant un message SMS sur le téléphone de l'utilisateur. L'utilisateur se connecte à l'aide d'un code à usage unique contenu dans le message SMS.

Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) est une solution de messagerie multiplateforme qui nous permet d'envoyer des notifications aux utilisateurs de manière fiable et sans frais.

Here Maps APIs

« HERE Technologies » propose un grand nombre d'API, offrant un accès rapide aux données de localisation. Ces API peuvent fournir des cartes, des itinéraires, un géocodage, des lieux, des informations de positionnement, de circulation, de transit ou encore météorologiques.

4.4 Fonctionnement de l'application :

Dans cette partie nous allons expliquer les interactions entre les différentes technologies utilisées dans notre application

4.4.1 Interaction client-serveur

L'application se connecte au serveur via le package « http.dart » en envoyant une requête HTTP et recevant une réponse sous forme de « JSON ».

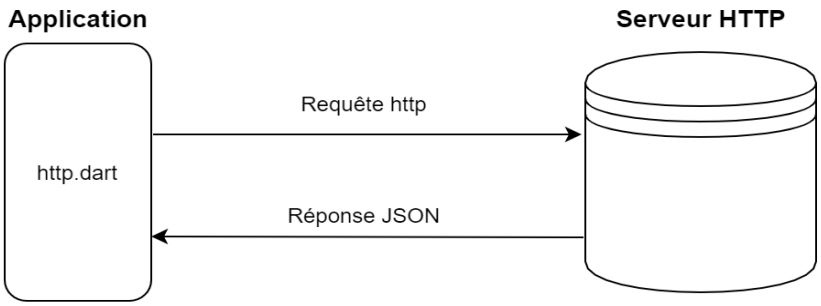

Figure IV. 5 Schéma de connexion http

4.4.2 Hachage des mots de passes des utilisateurs

Le hachage [27] est le processus de codification ou de chiffrement des données, il est l'une des pratiques de sécurité les plus basiques qui doit être effectuée. Sans cela, chaque mot de passe stocké peut être volé si le support de stockage (typiquement une base de données) est compromis .En appliquant un hachage sur le mot de passe avant de le stocker, nous rendons la tâche d'un attaquant très difficile pour connaitre le mot de passe original, et nous avons toujours la possibilité de comparer le mot de passe haché à une chaîne reçue.

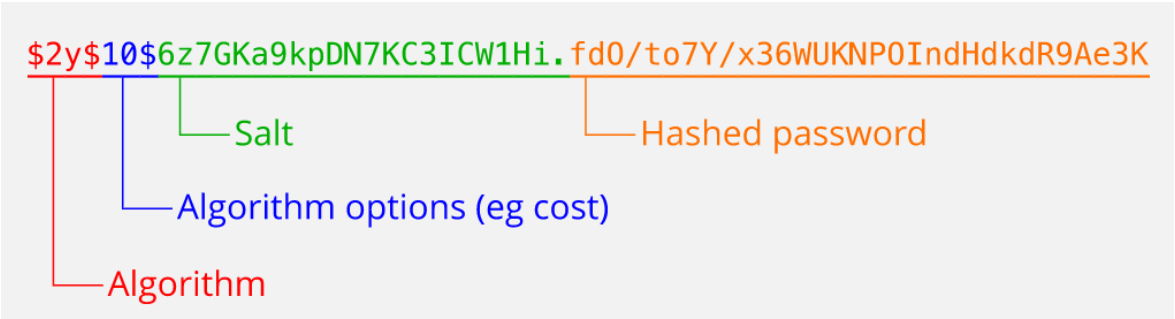

Figure IV. 6 Format de hachage « grain de sel »

4.5 Le Modèle physique des données

Cette étape permet de construire la structure finale de la base de données avec les différents liens entre les éléments qui la composent. Il s'agit de préparer l'implémentation dans un SGBDR.

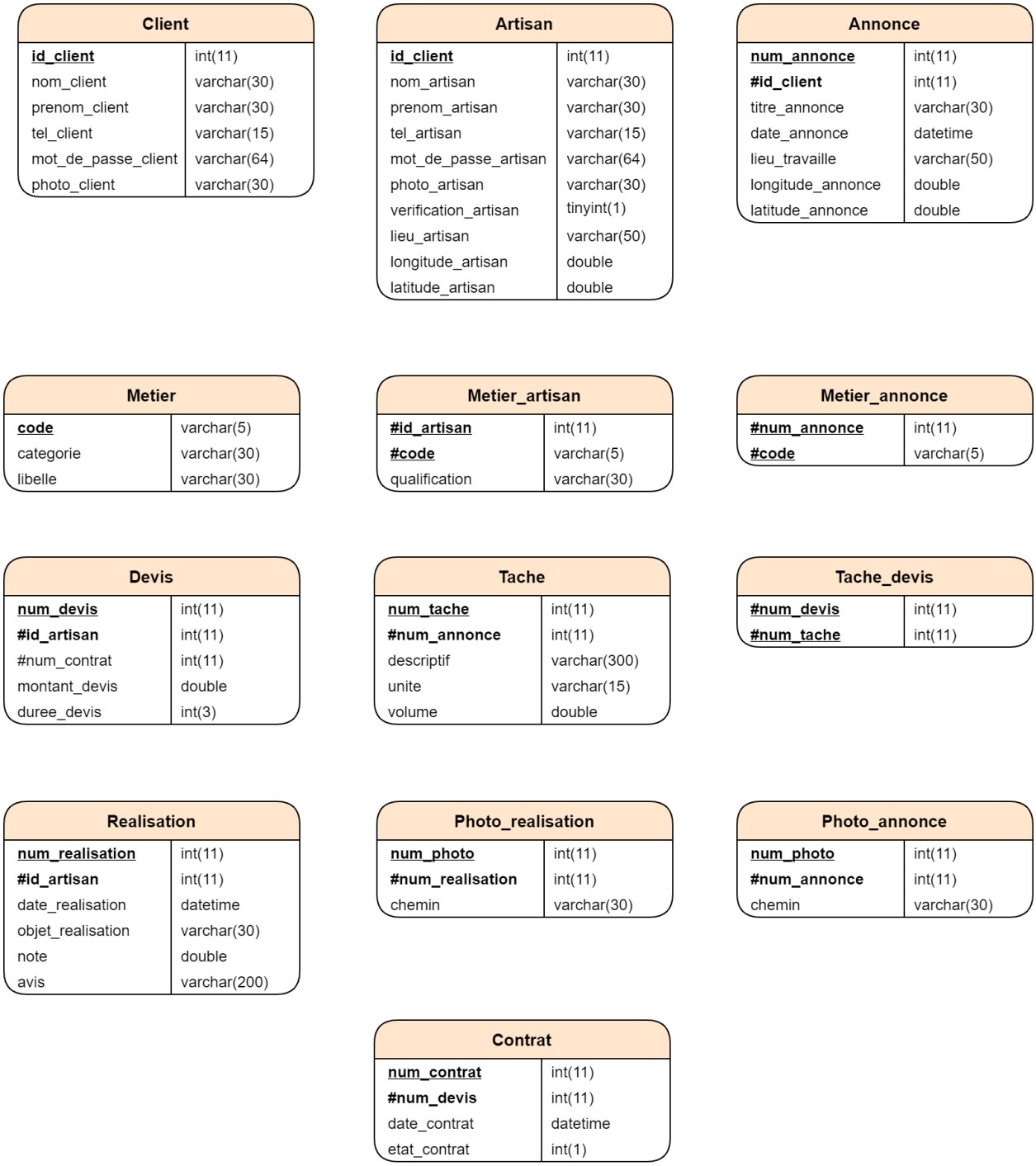

Figure IV. 7 Modèle physique de données

4.6 Interfaces hommes machine

Les interfaces homme-machine ou IHM sont les moyens et outils mis en œuvre afin qu'un humain puisse contrôler et communiquer avec une machine.

4.6.1 Coté "client"

Dans cette partie nous présentons quelques captures d'interfaces de notre application ***VieFacile*** coté « client ».

Interface inscription :

C'est l'interface qui permet au visiteur de saisir ses informations personnelles afin de s'inscrire sur notre plateforme en tant que client.

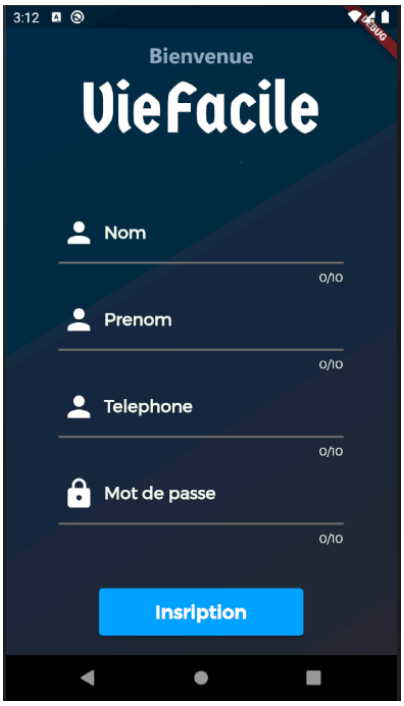

Figure IV. 8 Interface inscription client

Interface artisans de proximité :

C'est la page d'accueil du client après sa connexion à son espace. Dans cette page, il peut effectuer une recherche d'artisans à proximité de son emplacement en choisissant les métiers qu'il souhaite trouver.

Figure IV. 9 Interface artisans de proximité

Interface poster une annonce :

C'est l'interface qui permet au client de poster une annonce en saisissant le titre et l'emplacement de cette dernière, ainsi que les métiers concernés, les taches à effectuer et les photos du chantier.

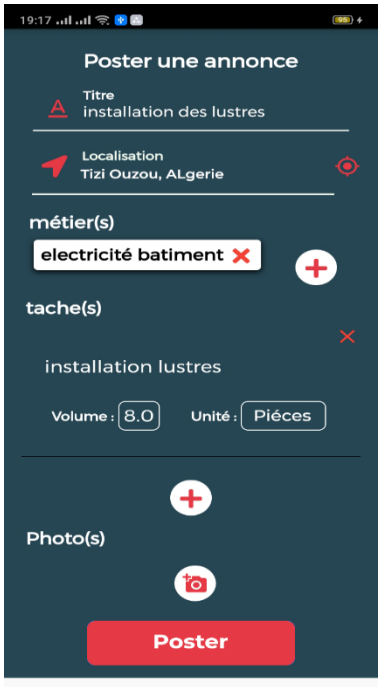

Figure IV. 10 Interface poster une annonce

Interface devis :

C'est l'interface ou se trouvent les devis proposés pour une annonce postée par le client. En cliquant sur le bouton « **Accepter** » le client doit prendre sa décision concernant le devis.

Figure IV. 11 Interface devis proposés

4.6.2 Coté "artisan"

Dans cette partie nous présentons quelques captures d'interfaces de notre application ***VieFacile*** coté « artisan ».

Interface annonces de proximité

C'est la page d'accueil d'un artisan après sa connexion à son espace, et c'est la page ou il peut effectuer une recherche des annonces de travail relatives à ses qualifications par rapport à la proximité de son emplacement. Ceci, en ayant la liberté d'évaluer la distance maximale du lieu de travail de son emplacement.

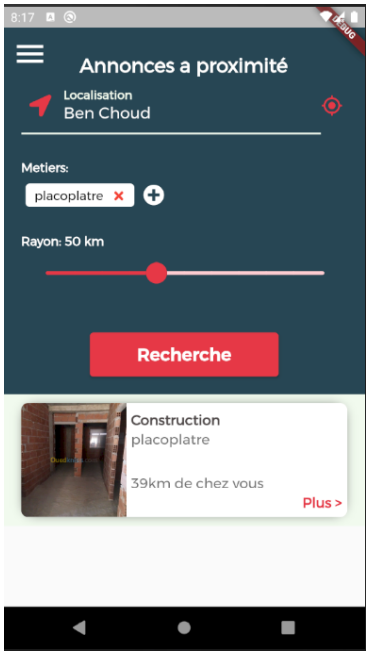

Figure IV. 12 Interface annonces à proximité

Interface détails d'une annonce

C'est l'interface qui s'affiche à l'artisan ayant choisi une annonce de travail. Cette interface comporte plusieurs photos du lieu de travail, les différentes tâches à effectuer, ainsi que son emplacement et la date de publication de l'annonce. L'artisan peut proposer un devis a cette annonce en appuyant sur le bouton « **Proposer un devis** »

Figure IV. 13 Interface détails d'une annonce

Interface profil d'artisan

C'est l'interface ou l'artisan peut éventuellement modifier ses informations personnelles ainsi que sa photo et ses qualifications. D'autre part, l'artisan peut consulter ses réalisations et ses travaux en cours.

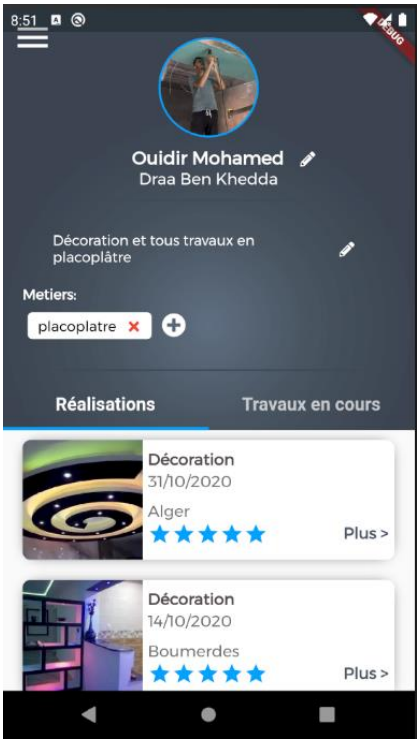

Figure IV. 14 Interface profil artisan

Interface détails d'une réalisation

C'est l'interface qui s'affiche à l'artisan après avoir choisi une de ses réalisations. Cet espace contient des photos de la réalisation ainsi que l'avis et la note du client.

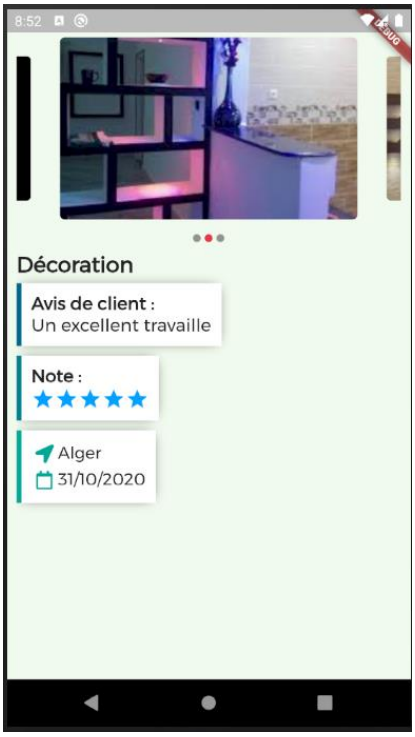

Figure IV. 15 Interface détails d'une réalisation

4.6.3 Coté "administrateur"

Dans cette partie nous présentons quelques captures d'interfaces de notre application ***VieFacile*** coté « administrateur ».

Interface artisans en attente

Cette l'interface permet à l'administrateur de consulter les nouveaux artisans inscrits à notre application. L'administrateur vérifie leurs qualifications et leurs métiers afin de les approuver ou de les rejeter.

Figure IV. 16 Interface artisans en attente

Interface gérer les annonces

Cette l'interface permet à l'administrateur de consulter les nouvelles annonces insérées par les clients afin de les étudier. Dans le cas où une annonce a un contenu indésirable, l'administrateur a la latitude de la supprimer.

Figure IV. 17 Interface gérer les annonce

4.7 Conclusion

Ce chapitre est dédié à la réalisation de notre plateforme de mise en relation ***VieFacile***. Nous avons, dans un premier temps, présenté l'environnement matériel et surtout logiciel adopté pour l'implémentation des différentes fonctionnalités offertes par notre système. Dans un second temps, nous avons expliqué l'implémentation du principe du fonctionnement de notre application et donné le schéma physique de notre base de données. Avant de conclure, certaines des interfaces de notre application ont été présentées en expliquant leurs utilités.

Conclusion générale

Le thème de notre mémoire s'est porté sur la réalisation d'une application mobile ***VieFacile*** en intégrant des services, des informations et des fonctionnalités qui vont au-delà de la notion habituelle du service de mise en relation artisans/particuliers.

Après avoir entamé l'Analyse & Conception de notre système en utilisant l'extension d'UML et la méthodologie MERISE ainsi que la conception et l'implémentation de la base de données, nous avons terminé par la présentation des différents outils de développement utilisés lors de la mise en œuvre de notre application, ainsi que la description de son fonctionnement illustré par ses différentes interfaces.

Nous estimons avoir réalisé un système répondant à l'objectif que nous nous sommes fixés au départ, à savoir la mise en place d'une application simple, efficace qui permet de mettre en relation à temps réel des artisans avec des particuliers demandeurs de services et l'amélioration des services offerts par les artisans ainsi l'élargissement de leurs champ de clientèle.

Ce travail étant un essai, n'est donc pas un modèle unique et parfait, c'est pourquoi nous restons ouverts à toutes les critiques et prêts à recevoir toutes les suggestions et remarques tendant à améliorer davantage cette initiative afin de lancer notre startup dans le futur proche. Et que le public puisse bénéficier des services offerts par notre application de mise en relation en lui facilitant la vie. Ainsi nous laissons l'inscription en tant qu'entreprise demandeuse ou fournisseuse de services comme perspective qui sera ajoutée à notre plateforme avant de la lancer.

En effet, la réalisation de ce projet nous a permis d'apprendre à nous organiser, à travailler en groupe, à améliorer nos connaissances et nos compétences dans le domaine de la programmation. Nous avons appris à mieux manipuler les langages et outils utilisés.

Enfin, ces cinq années d'étude à l'université nous ont permis d'acquérir de vastes connaissances non seulement dans le domaine d'informatique mais aussi dans la gestion et l'organisation du temps et du travail. Ces cinq années nous ont donné la chance de connaitre des amis étudiants et des enseignants respectables, et toutes ces personnes ont contribué à l'élargissement de notre vision, de notre façon de voir les choses et la construction de notre personnalité actuelle.

Références

[1] *L'essor des plateformes de mise en relation*, Publié le 2 Mai 2018, Lien : https://blog.hub-grade.com/essor-des-plateformes-de-mise-en-relation/ (Consulté le 28 Juillet 2020)

[2] *Internet number resource status report*, Publié le 31 Mars 2019 [PDF], Lien : https://www.nro.net/wp-content/uploads/NRO-Statistics-2019-Q1.pdf (Consulté le 28 Juillet 2020)

[3] *La naissance du web "birth web"*, Publié sur le site web du CERN, Lien : https://home.cern/fr/science/computing/birth-web (consulté le 28 Juillet 2020)

[4] *Annual number of global mobile app downloads 2016-2019*, Publié le 17 Janvier 2020, Lien : https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/ (consulté le 29 Juillet 2020)

[5] *Architecture web*, Université Libre de Bruxelles, Publié le 28 Avril 2004, Modifié le 27 Mars 2006, Lien : https://www2.ulb.ac.be/cours/acohen/travaux_2004_infodoc/projettechnologiesweb/html/architectures web.html (consulté le 28 Juillet 2020)

[6] Amazon, Publié sur Wikipédia, Date de dernière modification 29 juillet 2020, Lien : https://fr.wikipedia.org/wiki/Amazon (consulté le 29 Juillet 2020)

[7] Airbnb, Publié sur Wikipédia, Date de dernière modification 03 Juillet 2020, Lien : https://fr.wikipedia.org/wiki/Airbnb (Consulté le 29 Juillet 2020)

[8] Uber, Publié sur Wikipédia, Date de dernière modification 19 Juillet 2020, Lien : https://fr.wikipedia.org/wiki/Uber_(entreprise) (Consulté le 29 Juillet 2020)

[9] Uber Eats, Publié sur Wikipédia, Date de dernière modification 28 Juillet 2020, Lien : https://fr.wikipedia.org/wiki/Uber_Eats (consulté le 29 Juillet 2020)

[10] Medicalib, plateforme de mise en relation pour des soins à domicile, Lien : https://www.medicalib.fr/ (Consulté le 30 Juillet 2020)

[11] HomeAdvisor, le premier site mondial de mise en relation entre particulier et artisan, Lien : https://www.abouthomeadvisor.com/who-we-are (Consulté le 01 Octobre 2020)

[12] Youpijob, un site dédié aux services à domicile, Lien : https://youpijob.fr/qui-sommes-nous (Consulté le 01 Octobre 2020)

[13] Frizbiz, une plateforme de Jobbing et de services entre particuliers, Lien : https://www.frizbiz.com/fr/articles/fonctionnement-de-frizbiz (Consulter le 01 Octobre 2020)

[14] Ouedkniss, un site d'e-commerce et de petites annonces algérien, publié sur Geekyalgeria, Lien : http://geekyalgeria.com/ouedkniss (Consulté le 01 Octobre 2020)

[15] fixit, 1ère application mobile algérienne de mise en reletion entre articuliers et bricoleurs et/ou artisans, publié sur Algérie 360, Lien : https://www.algerie360.com/fixit-1ere-application-mobile-algerienne-de-mise-en-relation-entre-particuliers-et-bricoleurs-et-ou-artisans (Consulté le 01 Octobre 2020)

[16] Allo Khedma, une appli mobile pas comme les autres, publié sur MobileAlgérie, Lien : https://mobilealgerie.com/allo-khedma-une-appli-mobile-pas-comme-les-autres (Consulté le 01 Octobre 2020)

[17] *UML, c'est quoi* ? publié sur Openclassrooms, Mis à jour le 03 février 2020, Lien : https://openclassrooms.com/fr/courses/2035826-debutez-lanalyse-logicielle-avec-uml/2035851-uml-c-est-quoi (Consulté le 01 Octobre 2020)

[18] Qu'est-ce que Flutter, publié sur Frandroid, Lien :

https://www.frandroid.com/android/535194_quest-ce-que-flutter-loutil-permettant-de-creer-des-applications-android-et-ios (Consulté le 01 Octobre 2020)

[19] L'architecture logicielle MVC, publié sur Medium.com, Lien : https://medium.com/@belcaid.mehdi/larchitecture-logicielle-mvc-1a8bbb5cf6dc (Consulté le 02 Octobre 2020)

[20] MERISE : une méthode systémique de conception de SI, par Bernard Espinasse, Lien : http://www.lsis.org/dea/M6optionD/Exp-GL5Merise.pdf

[21] Dart (langage), publié sur Wikipédia, Date de dernière modification le 23 septembre 2020, Lien : https://fr.wikipedia.org/wiki/Dart_(langage) (Consulté le 01 Octobre 2020)

[22] MySQL, publié sur Wikipédia, Date de dernière modification le 7 septembre 2020, Lien : https://fr.wikipedia.org/wiki/MySQL (Consulté le 01 Octobre 2020)

[23] Qu'est ce qu'Apache ? publié sur Hostinger, Date de dernière modification le janvier 23, 2020, Lien : https://www.hostinger.fr/tutoriels/quest-ce-quapache-serveur-web-apache (Consulté le 01 Octobre 2020)

[24] PHP, publié sur Wikipédia, Date de dernière modification le 16 septembre 2020, Lien : https://fr.wikipedia.org/wiki/PHP (Consulté le 01 Octobre 2020)

[25] Présentation de JSON, publié sur json.org, Lien : https://www.json.org/json-fr.html (Consulté le 01 Octobre 2020)

[26] Interface de programmation, publié sur journaldunet.fr, Date de dernière modification le 20 janvier 2019, Lien : https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203559-api-application-programming-interface-definition-traduction (Consulté le 01 Octobre 2020)

[27] Hachage, publié sur lemagit.fr, Date de dernière modification aout 2016, Lien : https://www.lemagit.fr/definition/Hachage (Consulté le 01 Octobre 2020)