République Algérienne Démocratique et Populaire
Ministre de L'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane MIRA – Béjaïa
Faculté des sciences Exactes
Département d'Informatique

Mémoire de fin de cycle

Master 2

Option : Génie Logiciel

Présenté par :

KHIREDDINE Rahim Et LADJOUZE Salah

Thème

Conception et Réalisation d'une Platforme Freelance "SELEKENI"

Encadrer par : ELSAKAAN Nadim

MCA. AISSANi Sofiane	(Université-Béjaïa)	Président
Mme. BENNAI Sofia	(Université-Béjaïa)	Examinatrice
Dr. ELSAKAAN Nadim	(Université-Béjaïa)	Encadrant

Année 2019-2020

Remerciements

Nous remercions Allah le miséricordieux et tout puissant de nous avoir accordé le courage et la patience pour mener à terme le présent mémoire.

Nous tenons, également, à exprimer notre sincère reconnaissance et notre profonde gratitude à notre promoteur M. ELSAKAAN Nadim, pour sa disponibilité, ses orientations, ses précieux conseils et ses encouragements qui nous ont permis de mener à bien ce travail.

Nous tenons à exprimer notre gratitude aux membres du jury pour avoir accepté de juger ce travail. Nous tenons à exprimer notre gratitude aux membres du jury pour avoir accepté de juger ce travail. Nous exprimons nos profonds remerciements à l'encontre de nos parents qui nous ont enseigné la patience, la politesse, le sacrifice et qui ont toujours été là pour nous. Un énorme merci à nos familles et amis pour leurs éternels soutiens et la confiance qu'ils ont eu en nos capacités.

Dédicace

Ce travail est dédié :

à nos chers parents qui nous ont soutenus et encouragés durant toute notre scolarité.

à nos frères et sœurs.

à nos enseignants.

à nos amis.

et à toutes les personnes qui nous ont apprêtés de l'aide.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Actuellement, l'impact que prend le web et les technologies de l'informatiques ne cessent de se développer et être une partie indissociable dans notre vie, ce qui contribue à l'amélioration des services offerts et à faciliter les tâches qu'on peut réaliser quotidiennement notament, le travail, la vente et achat électronique, la communication et différents autres services...

De ce fait, les technologies du web ne cessent de se développer, ce qui Contribue à l'amélioration des services offerts de plus en plus, en l'occurrence le Commerce électronique, la publicité, le marketing et d'autres à venir dans l'avenir, L'impact que prend le web dans notre vie quotidienne est de plus en plus considérable Et ne cesse de s'accroitre quant à son application dans divers domaines et cela jusqu'à avoir un ascendant même dans l'exercice de notre travail quotidien.

De nos jours, le fait de travailler à distance est devenu la tendance qui se développe en fur et à mesure, vu qu'il nous épargne de se présenter quotidiennement au lieu du travail où l'on doit offrir des services. Avec cette nouvelle méthode de travail, qui révolutionne le monde professionnel, il devient possible de travailler tout en restant chez soi, ou même en changeant de lieu fréquemment, et ça se qu'on appelle le travailleur indépendant ou le freelance.

En se rendant compte de l'essont et de l'accessibilité des moyens technologiques dans notre pays, on peut se demander comment améliorer et apporter une nouvelle vision et l'adapter aux moyens disponibles.

Par ailleurs, l'objectif ultime de notre projet est d'offrir une Plateforme Freelance de qualité, qui se concentre davantage sur le nombre de fonctionnalités offertes, la simplicité et de la facilité avec une interface simple et intuitive tout en étant efficace pour pallier au problème cité.

Pour réaliser ce travail, il a fallu suivre un plan qui couvre l'ensemble des parties du projet, à commencer par la partie purement théorique jusqu'aux concepts de programmation et d'implémentation.

Cependant, notre plan comporte en tout quatre chapitres, comme suit :

• Le premier chapitre intitulé « Généralités » regroupe des généralités sur les différentes parties de la plateforme que nous allons développer.

• Le deuxième chapitre intitulé « présentation du projet» donne un premier aperçu sur les différentes facettes et fonctionnalités qui seront présentes dans notre plateforme

• Le troisième chapitre intitulé « Analyse et conception » donne plus de détails sur les fonctionnalités principales de notre système et c'est la base sur laquelle nous allons nous appuyer pour l'implémentation de ce dernier.

• Le quatrième et dernier chapitre intitulé « Implémentation » concerne la partie pratique c'est à dire les détails de la partie réalisation de notre plateforme.

Chapitre 1

Généralités

Introduction

Dans ce chapitre nous donnons en premier lieu des notions générales sur les Framework et les langages que nous utiliserons pour développer notre Plateforme Freelance, en second lieu, on présente les différents outils et logiciels ainsi que le langage de modélisation UML. On termine par une présentation explicative de l'architecture MVC.

1.1. Les technologies de développement web

1.2. Architecture MVC

Un des plus célèbre désigne pattern s'appelle MVC, qui signifie **Modèle-Vue-Contrôleur** c'est lui que nous allons découvrir maintenant :

Le pattern MVC permet de bien organisé son code source.il va vous aider à savoir quels fichiers crée, mais surtout à définir leur rôle. Le bute de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distinct.

— **Modèle :** Cette partie gérer les données de notre site. Son rôle est d'aller récupérer les informations «brutes» dans la base de données, de les organiser et de les assembler pour qu'elle est puissent ensuite être traité par le contrôleur. On y trouve donc entre autres les requêtes SQL.

— **Vue :** Cette partie ce concentre sur l'affichage. Elle ne fait presque aucune calcule

et se contant de récupérer des variables pour savoir ce qu'elle doit afficher on y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.

— **Contrôleur :** Cette partie gérer la logique du code qui prends des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte a afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voire la page ou non (gestion des droits d'accès).

La figure suivante schématise le rôle de chacune de ces éléments[1].

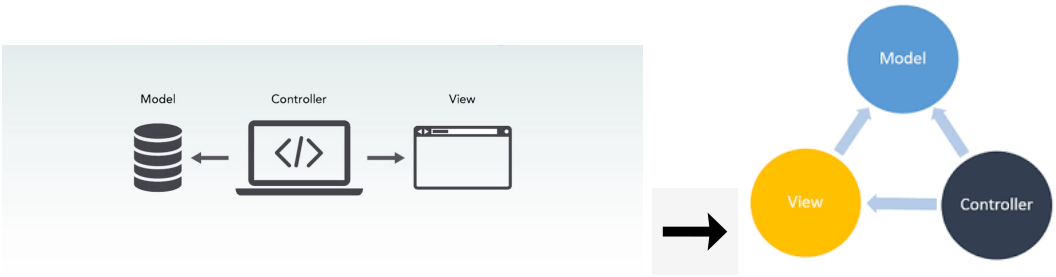

FIGURE 1.1 – Architecture du model MVC

Il est important de bien comprendre comment ces élément s'agencent et communiquent entre eux (observez bien la Figure-2).

Il faut tout d'abord retenir que le contrôleur est le chef d'orchestre : c'est lui qui reçoit la requête du visiteur et qui contacte d'autres fichiers pour échanger des informations avec eux.

Le fichier contrôleur demande les données au modèle sans lui soucier de la façon dont lui-ci va les récupérer. Une fois les données récupérées, le contrôleur les transmet a la vue qui se changera d'afficher la liste des messages.

En gros, le visiteur demandera la page au contrôleur et c'est la vue qui lui sera retournée, comme schématisé sur la figure suivante :

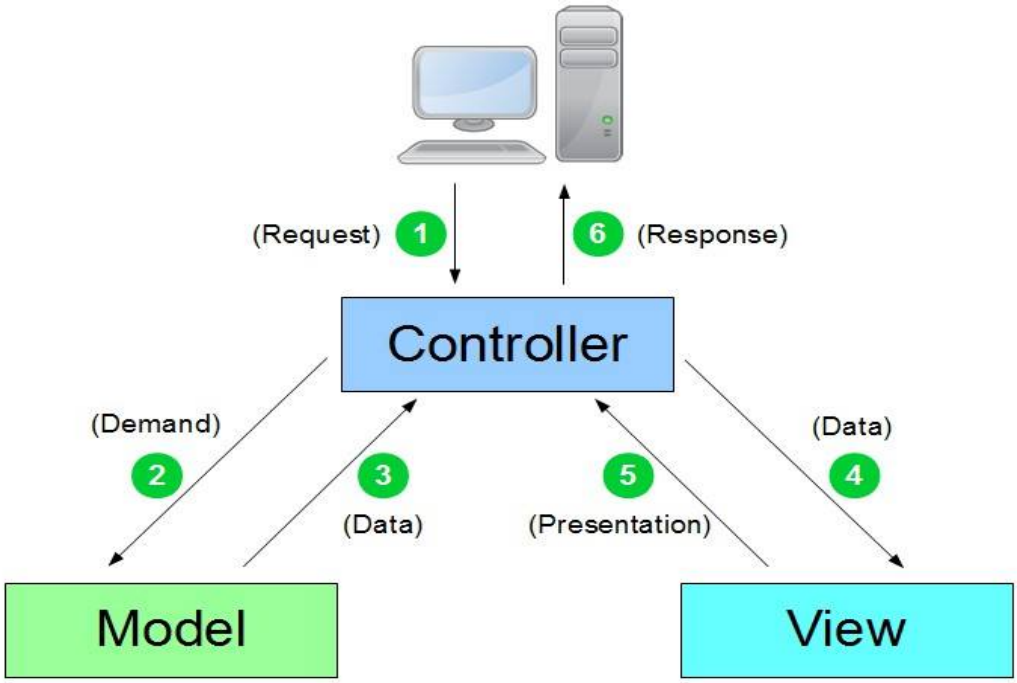

FIGURE 1.2 – Modèle MVC avec l'utilisateur

1.2.1. HTML 5

Pour HyperText Markup Language 5 est une version du célèbre format HTML utilisé pour concevoir les sites Internet. Celui-ci se résume à un langage de balisage qui sert à l'écriture de l'hypertexte indispensable à la mise en forme d'une page Web.[2]

1.2.2. CSS 3

Est un langage informatique utilisé sur l'Internet pour mettre en forme les fichiers HTML. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.[3]

1.2.3. JavaScript

JavaScript désigne un langage de développement informatique, et plus précisément un langage de script orienté objet. On le retrouve principalement dans les pages Internet. Il permet, entre autres, d'introduire sur une page web ou HTML des petites animations ou des effets. Le langage JavaScript se distingue des langages serveurs par le fait que l'exécution des taches est opérée par le navigateur lui-meme, sur l'ordinateur

de l'utilisateur, et non sur le serveur web[2].

1.2.4. Ajax

Le terme AJAX est l'abréviation de "Asynchronous JavaScript and XML". L'AJAX n'est pas un langage de programmation mais correspond plutôt à un ensemble de techniques utilisant des technologies diverses pour communiquer avec le serveur. Il désigne un type de conception de pages Web permettant l'actualisation de certaines données d'une page sans procéder au rechargement total de cette page[4].

1.2.5. PHP

Désigne un langage informatique, ou un langage de script, utilisé principalement pour la conception de sites web dynamiques. Sur un plan technique, le PHP s'utilise la plupart du temps cote serveur. Il génère du code HTML, CSS ou encore XHTML, des données (en PNG, JPG, etc.) ou encore des fichiers PDF. Il fait, depuis de nombreuses années, l'objet d'un développement spécifique et jouit aujourd'hui d'une bonne réputation en matière de fiabilité et de performances[2].

1.2.6. SQL

Est un langage informatique utilisé pour exploiter des bases de données. Il permet de façon générale la définition, la manipulation et le controle de sécurité de données. Dans la pratique, le langage SQL est utilisé pour créer des tables, ajouter des enregistrements sous forme de lignes, interroger une base de données, la mettre à jour, ou encore gérer les droits d'utilisateurs de cette base de données[2].

1.3. Frameworks

1.3.1. Bootstrap

Bootstrap est un framework développé par l'équipe du réseau social Twitter. Proposé en open source Ce framework est pensé pour développer des sites avec un design responsive, qui s'adapte à tout type d'écran. Il utilise les langages HTML, CSS et Ja-

vaScript.Il fournit des outils avec des styles déjà en place pour des typographies, des boutons, des interfaces de navigation et bien d'autres encore[5].

1.3.2. Symfony 5

C'est un ensemble de composants PHP ainsi qu'un framework MVC libre écrit en PHP. Il permet de réaliser des sites internet dynamiques de manière rapide, structurée, et avec un développement clair. Les développeurs peuvent travailler sur ce Framework très facilement, seuls ou en équipe, grâce à la facilité de prise en main.

Symphony propose entre autre :

— Une séparation du code en trois couches,selon le modèle MVC,pour une plus grande maintenabilité et évolutivité.

— Un templating simple, basé sur PHP et des jeux de "helpers", ou fonctions additionnelles pour les gabarits... Comme alternative au PHP, on peut aussi utiliser le moteur de templates Twig dont la syntaxe est plus simples[6].

— Des performances optimisées et un système de cache pour garantir des temps de réponse optimums et préchargement qui a pour but de compiler et charger des classes au démarrage[6].

— Le support de l'Ajax pour renforcer la réactivité du moteur de recherche et le rendre plus vivant.

— Une gestion des URL parlantes (liens permanents), qui permet de formater l'URL d'une page indépendamment de sa position dans l'arborescence fonctionnelle.

— Un système de configuration en cascade qui utilise de façon extensive le langage YAML.

— Un générateur de back-office et un "démarreur de module" (scaffolding).

— Un support de l'I18N - Symfony est nativement multi-langue.

— Une couche de mapping objet-relationnel (ORM) et une couche d'abstraction de données (cf. Doctrine et son langage DQL).

— La sécurité dans Symfony est robuste et très poussée, vous pouvez la contrôler-facilement. Filtrage des inputs, authentification, gestion des sessions, protection contreles injections et les failles XSS, CSRF, Cette série d'outils aborde les différentes mé-caniques que tout développeur Symfony peut utiliser, afin d'assurer la sécurité et lafiabilité de l'application à développe.

— Une architecture extensible, permettant la création et l'utilisation de plugins.

— Assetic : gestionnaire de fichiers .css et .js [7].

1.3.3. Doctrine 2

C'est un ORM (Object-Relational Mapping) pour PHP, intégré par défaut dans Symfony, il permet de donner l'illusion de travailler avec une base de données objet alors que l'on est sur une base de données relationnelle. Grâce à cela le développeur ne fait plus de requête SQL (sauf cas spécifiques rares) et travailles avec ses objets[8].

1.4. Environnement de développement

1.4.1. Visuel studio code

Visual Studio Code est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et possède un riche écosystème d'extensions pour d'autres langages (tels que C C++, C, Java, Python, PHP, Go) et des runtimes (tels que .NET et Unity)[9].

1.4.2. WampServer

C'est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données[10].

1.4.3. MySQL

Le terme MySQL, pour My Structured Query Language, désigne un serveur de base de données distribué sous licence libre GNU (General Public License). Il peut être utilisé sur de nombreux systèmes d'exploitation (Windows, Mac OS, etc.). Il supporte les langages informatiques SQL et SQL/PSM. Dans la pratique, le serveur MySQL peut se résumer à un lieu de stockage et d'enregistrement des données[2].

1.4.4. Apache

Le projet de serveur HTTP Apache vise à développer et à maintenir un serveur HTTP à code source ouvert pour les systèmes d'exploitation modernes. L'objectif de ce projet est de fournir un serveur sécurisé, efficace et extensible qui fournit des services HTTP synchronisés avec les normes HTTP actuelles[11].

1.5. Model de développement UP

Le processus unifié est une méthode générique de développement logiciel, son principal intérêt est de limiter les risques lors du développement des projets[12].

1.5.1. Principes du processus unifié

- **itératif et incrémental :** Le projet est découpé en itérations ou étapes de courte durée qui permettent de mieux suivre l'avancement globale. A la fin de chaque itération une partie exécutable du système finale est produite, de façon incrémentale[12].
- **Centré sur l'architecture :** Tout système complexe doit être décomposé en partie modulaire afin d'en faciliter la maintenance et l'évolution[12].
- **Guidé par les cas d'utilisation d'UML :** Le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés[12].
- **Piloté par les risques :** Les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout les levers le plus rapidement possibles[12].

1.5.2. Cycle de vie du processus unifié

1.5.2.1. Architecture bidirectionnelle

UP gère le processus de développement par deux axes :

- **L'axe vertical :** représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités, d'enchaînements, d'artefacts et de travailleurs.[12]

- **L'axe horizontal :** représente le temps et montre le déroulement du cycle de vie du processus cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en terme de cycles, de phases, d'itérations et de jalons.[12]

1.5.2.2. Activités du processus UP

- **Expression des besoins :** UP propose d'appréhender l'expression des besoins en se fondant sur une bonne compréhension du domaine concerné pour le système à développer et une modélisation des procédures du système existant. Ainsi, UP distingue deux types des besoins :
    1. Les besoins fonctionnels.
    2. Les besoins non fonctionnels.[13]
- **Analyse** : L'analyse permet une formalisation du système à développer en réponse à l'expression des besoins formulée par les utilisateurs.[13]
- **Conception** La conception prend en compte les choix d'architecture technique retenus pour le développement et l'exploitation du système.[13]
- **Implémentation** : Cette phase correspond à la production du logiciel sous forme de composants, de bibliothèques ou de fichier.[13]
- **Implémentation** : Cette phase correspond à la production du logiciel sous forme de composants, de bibliothèques ou de fichier.[13]
- **Les tests permettent de vérifier :**
    1. La bonne implémentation de toutes les exigences (fonctionnelles et techniques).
    2. Le fonctionnement correct des interactions entre les objets.
    3. La bonne intégration de tous les composants dans le logiciel. [13]

1.5.2.3. Phases du processus UP

- **Phase d'initialisation :** Consiste à définir la vision du projet, sa portée, sa faisabilité, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.[12]
- **Phase d'élaboration :** La phase d'élaboration poursuit trois objectifs principaux en parallèle :[12]
    1. Identifier et décrire la majeure partie des besoins des utilisateurs.
    2. Construire l'architecture de base du système.
    3. Lever les risques majeurs du projet.

- textbfPhase de construction : Consiste concevoir et implémenter l'ensemble des éléments opérationnels (autres que ceux de l'architecture de base). C'est la phase la plus consommatrice en ressources et en effort[12].
- textbfPhase de transition : Permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux[12].

1.6. Langage UML (Unified Model ingLangauage)

Il se définit comme un langage de modélisation graphique et textuel destiné à comprendre et à définir des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML modélise l'ensemble des données et des traitements en élaborant des différents diagrammes[13].

1.6.1. Présentation générale des diagrammes

UML dans sa version 2 propose treize diagrammes qui peuvent être utilisés dans la description d'un système. Ces diagrammes sont regroupés dans deux grands ensembles[13].

1.6.1.1. Diagrammes Structurels

Ces diagrammes, au nombre de six, ont vocation à représenter l'aspect statique d'un système.

— **Diagramme de classes**.

— **Diagramme d'objets**.

— **Diagramme de packages**.

— **Diagramme de structure composite**.

— **Diagramme de composants**.

— **Diagramme de déploiement**.

1.6.2. Diagrammes comportementaux

Ces diagrammes représentent la partie dynamique d'un système réagissant aux événements et permettant de produire les résultats attendus par les utilisateurs. Sept diagrammes sont proposés par UML :

— **Diagramme de cas d'utilisation**

— **Diagramme global d'interaction**

— **Diagramme de séquence**

— **Diagramme de temps**

— **Diagramme d'activité**

— **Diagramme d'état-transition**

1.7. Model du business B to B

On pourrait dire que le Freelancing a beaucoup évolué. Mais surtout le digital a fait évoluer, muter notre univers. Nous le constatons dans notre quotidien. Le Freelance a su saisir les opportunités créatives et relationnelles pour se créer un écosystème digital très développé, qui donne du sens à la politique dont il fonctionne. Exploiter ces fabuleuses opportunités est également un enjeu pour les marques B2B.

De par leur nature inter-entreprises, la démarche B2B restent moins visibles que les marques B2C « business to consommateur ». Pourtant, en France, le B2B occupe une place prépondérante, avec 1 million de salariés, 120 000 entreprises, 50 % du commerce français en valeur.

B2B2B (Business to Business to Business) : L'entreprise fournisseur vend un produit à l'entreprise cliente (un distributeur), qui le revend tel quel à ses propres clients professionnels.

Conclusion

A travers ce chapitre nous avons vu quelque notion sur les technologies web ainsi les Frameworks et la gestion de l'architectures MVC. Dans le chapitre suivant on vous présente le cahier des charges après on passe à l'objectif de notre projet et a la description du système de notre plateforme Freelance passant sur le choix du logo et des couleurs utilisées.

Chapitre 2

Présentation de projet

Introduction

Comme tout projet informatique, nous devons passer impérativement par les étapes classiques, au premier lieu on vous présente le concept de notre système, par la suite une étude concurrentielle puis nous allons nous initier à la charte graphique, et à la fin nous passerons à l'expression des besoins fonctionnels et non fonctionnels.

2.1. Objectif du projet

Notre Approche a ce projet consiste à réaliser une Plateforme Freelance qui a pour but de publier un besoin et attendre les compétences, puis suite à ¸ça faire un choix parmi ces compétences et l'implication des facteurs tel que le temps, le coût, la performance ou la perfection en toute confiance à la plateforme intermédiaire.

Notre Plateforme Freelance cible Deux catégories de clients demandeurs de service et les réalisateurs du service Freelance.

— Les demandeurs de services sont des clients particuliers (public), entreprises ou société.

— Les Freelancer, sont les réalisateurs de service qui sont des experts dans leur domaine (Photoshop, montage vidéo, latex, programmation $\cdots$).

2.2. Architecture de notre systéme

L'architecture de «Selekeni », la plateforme comportera quatre parties, décrites ci-dessous :

— **Front Office :** Vitrine de la plateforme, accessible à tous les utilisateurs, elle leurs autorise l'effectuation d'une recherche avancé e. Permettra aux clients et aux freelancers de s'authentifier. Il est important que cette partie soit pratique et esthétique.

— **Middle Office :** Accessible qu'après authentification, il permettra aux clients et aux freelancers de modifier leurs informations.

— **Administration :** Permettra aux administrateurs simples d'administrer les clients et les freelancers, et au super administrateur d'administrer les administrateurs simples.

— **Back Office :** Permettra aux administrateurs de voir les statistiques des nombres de freelancers et des clients inscris et les nombres des besoins publiés. Les principaux objectifs qu'on vise 'à atteindre sont :

- Créer de l'emploi à des jeunes diplômes.
- Passer du mode de travail traditionnel au mode Freelance (travail indépendant).
- Le client aura le choix de choisir entre les offres qui ont répondu à son service.
- La réalisation du service dans les brefs délais avec le moindre coût, Grâce au mode de concurrence.
- Donner de la liberté totale au Freelancer (expert) pour réaliser sa mission n'importe où.

2.3. La vue de notre plateforme par rapport à l'étude concurrentielle

Pour mettre sur pied une plateforme efficace, il est nécessaire de réaliser des études comparatives des méthodes et stratégies utilisées par nos concurrents pour pouvoir les devancer, et atteindre les besoins des utilisateurs. C'est pourquoi nous avons effectué des recherches pour avoir une idée à propos de notre secteur en ligne. Alors nous avons sélectionne deux plateformes de freelance pour leur clarté, visibilité, et les services proposés, l'un au niveau national Freehali, et l'autre au niveau international khamsat .

Nom de la plateforme	Freehali	khamsat
URL de la plateforme	www.freehali.com	www.khamsat.com
Pays de service	Algérie	Egypte
Langue	Français	Arabe
Public visé	Freelancers algériens/ entreprises et agences Algérienne/utilisateurs Algériens voulant un service de freelance.	Freelancers arabophone/ entreprises et agences arabophone/utilisateurs arabophone voulant un service de freelance.
Points positifs	• Plateforme ergonomique. • Facilité de publier un besoin. • Choix multiple des freelances.	• Plateforme ergonomique. • Responsivité bien gérée. • Plus de 850 services proposés.
Points négatifs	• Responsivité de la plateforme mal gérée. • Manque de contacte direct entre les freelancers et les clients. • Le dossier du freelance n'est pas vérifié. • Possibilité d'utilisateurs malveillants de s'inscrire.	• Manque d'information sur les freelancers. • L'évaluation de chaque freelancer n'est pas indiqué. • Le dossier du freelancer n'est pas vérifié.

Table 2.1 – Tableau comparatif entre Freehali et khamsat

Les solutions apportées aprés l'étude concurrentielle faite sur deux plateformes freelance au niveau national et international (Freehali et Khamsat), nous nous sommes mis d'accord sur une structure de départ pour notre travail.

La plateforme freelance Selekeni sera une combinaison des points forts des deux plateformes déjà existantes, et l'amélioration des points négatifs notamment la simplicité d'utilisation et qui proposera des freelancers pour réaliser des services de toute categorie en toute sécurité et de bonne qualité dans les brefs délais.

2.4. Description générale

2.4.1. Environnement

On a opté pour les choix suivants :

— **Un serveur web :** Apache.

— **Un SGBD :** MySQL.

— Le client aura le choix de choisir entre les offres qui répondent à son service.

— Un moyen pour l'envoie des e-mails.

2.4.2. Caractéristiques de la platforme

— **Responsive :** L'utilisation de Bootstrap est recommandée, pour l'adaptation à tout type de support (mobile, tablette et ordinateur).

— **Compatible avec les dernières versions des navigateurs :** Chrome, Opéra, Firefox et Brave.

— **Disponibilité :** Disponible (24h/24h) et (7/7) et depuis n'importe ou dans le monde (sans vpn).

— **Flexibles et scalables :** Peut-être amélioré et avoir plus de fonctionnalités.

— Bonne ergonomie, se fixer sur l'aspect Esthétique, toute en respectant les points qui seront cités dans la charte graphique.

2.4.3. Caractéristiques de l'utilisateur

Notre plateforme criblera toutes les entreprises publiques ou privées et méme l'individu (une personne), qui auront un service ou une mission à faire que ce soit en informatique ou autre domaine (exemple : architecture, design $\cdots$).

2.5. Charte graphique

2.5.1. Logo

FIGURE 2.1 – Logo fond blanc et écriture Bleu.

FIGURE 2.2 – Logo fond noir et écriture blanche.

2.5.2. Couleurs

2.5.2.1. Couleurs primaires

Sont les couleurs qui dominent la plateforme, notre choix s'est porté sur ces deux couleurs pour attirer l'attention de l'utilisateur avec une certaine douceur.

— Code RVB : 54,169,225 (#36a9ff). Cette couleur est un bleu canard. Une couleur qui a pour but d'attirer l'attention du visiteur de notre site.

— Code RVB :255,255,255 (#ffffff). Cette couleur est le blanc. En tant que nuance plutôt claire, elle est mise en fond, pour faire ressentir les éléments placés sur les pages et les mettre en valeur.

2.5.2.2. Couleurs secondaires

— Code RVB : 202,202,202(#cacaca). Cette couleur est le gris claire. On va l'utiliser comme couleur de texte sur fond blanc, la lisibilité du texte est parfaitement correcte.

2.5.3. Typographie

2.5.3.1. Typographie des titres

— Police : Helvetica

— Font size : 40px

2.5.3.2. Typographie des paragraphes

— Police : Helvetica

— Font size : 35px

2.5.4. Maquetes

Le maquettage est une méthode de conception d'interface qui permet de proposer des inter- faces conformes aux attentes et besoins du l'utilisateur. Elle permet également à l'agence web de s'assurer que les besoins de l'utilisateur sont adaptés ou non au projet.[14]

La maquette ci-dessous représente l'interface responsive (adapté aux déférents supports ordinateur, tablette et mobile) de la page d'accueil de notre plateforme web.

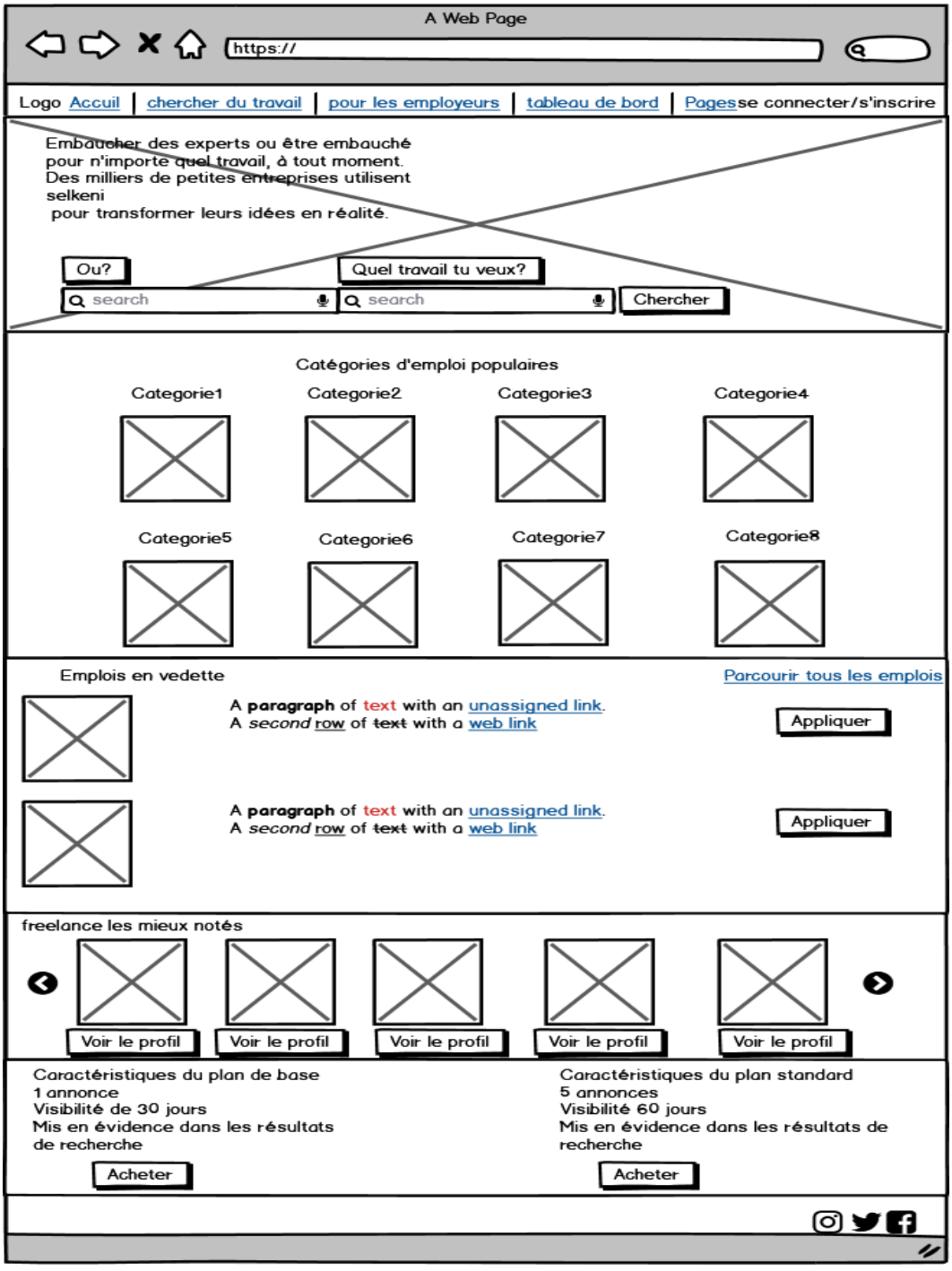

FIGURE 2.3 – Maquette de la page d'accueil - sur ordinateur

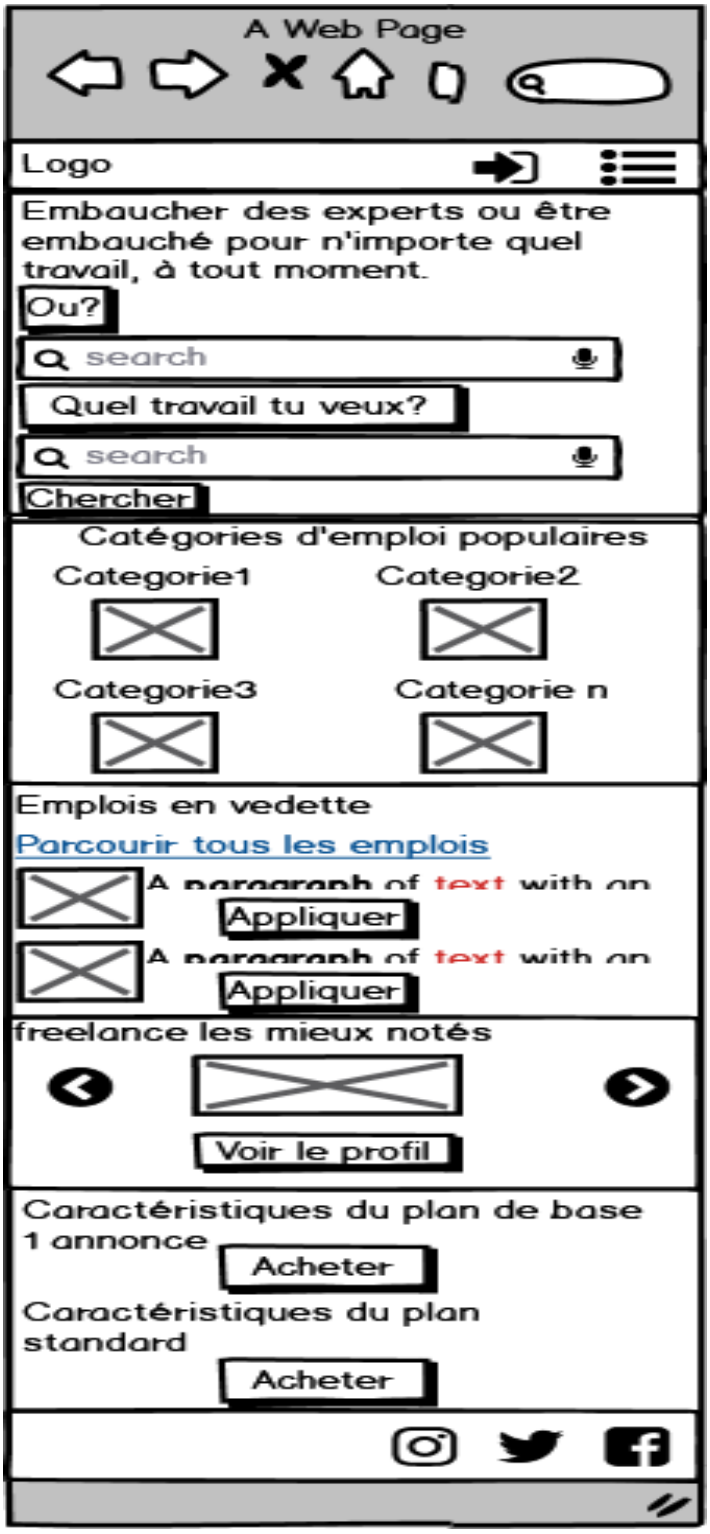

FIGURE 2.4 – Maquette de la page d'accueil - sur smartphone

2.6. Expression des besoins

Dans cette phase nous traitons l'analyse des besoins de notre projet, par identifications des acteurs impliqués dans le système et les besoins fonctionnels et non fonctionnels pour procéder au diagramme de cas d'utilisation.

2.6.1. Identification des acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié[12].

Les différents acteurs qui peuvent interagir avec notre système sont décrit ci-dessous.

- **Visiteur :** Représente toutes personnes qui accèdent à notre plateforme dans le but de visiter et de découvrir le contenu et les fonctionnalitésés de notre système (plateforme).
- **Commanditaire :** C'est une personne ou entreprise qui s'est déjà inscrie et authentifie pour accéder à son compte dans le but d'interagir soit en déposant une annonce d'un service ou commenter ou bien chercher un freelancer.
- **Freelancer :** C'est une personne qui est un expert dans son domaine (designer, programmeur, ceux qui font le montage vidéo ou Photoshop $\cdots$) à un compte avec le statut Freelancer, et peut commenter ou enchérir sur le service publiée par le client.
- **Administrateur simple :** C'est le responsable de la gestion des annonces de services postés par les commanditaires de les valider ou non, et il est chargé de surveiller les contenues postés (les avis, annonces $\cdots$).
- **Super administrateur :** Possède les privilèges d'un administrateurs simple et est en charge en plus, de gérer les administrateurs.

2.6.2. Identification des besoins

Les besoins fonctionnels expriment une action que doit exécuter le système en réponse à une demande de l'administrateur.

2.6.2.1. Besoins fonctionnels

La plateforme doit être en mesure de :

- **L'inscription :** Offrir une interface qui permet de créer les comptes des utilisateurs (commanditaires,freelancers, administrateurs).
- **La Recherche :** Pour un internaute la recherche est très importante, elle permet de mieux comprendre la politique et la gestion de la plateforme, permet d'effectuer une recherche par des critères soit par le titre du service ou lieu ou se trouves les services.
- **Le Contact :** Un Freelancer peut contacter un client en lui envoyant un message peut l'informer de son intérêt et lui donner le coût et la durée pour réaliser son service, et il peut même enchérir et donné le délai du développement du service sur la plateforme.
- **La Gestion :** Un commanditaire peut publier un service mais seulement après avoir créé un compte, autrement dit cette opération nécessite une inscription ou une authentification, en outre il peut mettre à jour son service ou son compte même la possibilité de le supprimer.
- Afficher les freelancers en vedette (ceux qui ont satisfaits le plus de tâches publier par des

2.6.2.2. Besoins non fonctionnels

Ces besoins sont généralement qualitatifs, de ce que suit on vous les explique en détails :

- **Interfaces ergonomiques :** faciles à manipuler pour tout un chacun.
- **Plateforme résponsive :** la plateforme doit s'adapter à toutes tailles d'écrans.
- **Formulaires simples :** Les formulaires d'inscriptions, de dépôt de service, et de connexion doivent être simples et rapide pour le remplissage, il ne faut pas qu'il y ait plus de deux pages pour valider un formulaire.
- **Aide et assistance** : Les utilisateurs doivent être assistés lors du remplissage des formulaires.
- **Sécurité :** Sécuriser la plateforme contre les différentes failles : faille XSS, injection SQL, etc.
- **Abonnement :** Chaque freelancer doit payer son abonnement.

2.6.2.3. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation montre les interactions fonctionnelles entre les acteurs et le système à l'étude. Un cas d'utilisation correspond à un certain nombre d'actions que le système devra exécuter en réponse à un besoin d'un acteur[12].

2.6.2.4. Diagramme de cas d'utilisation –visiteur

La figure ci-dessous représente le diagramme de toutes les fonctionnalités offertes au visiteur sur notre plateforme, on cite par exemple créer un compte, consulter les freelancers, consulter les tâches publier et effectuer une recherche.

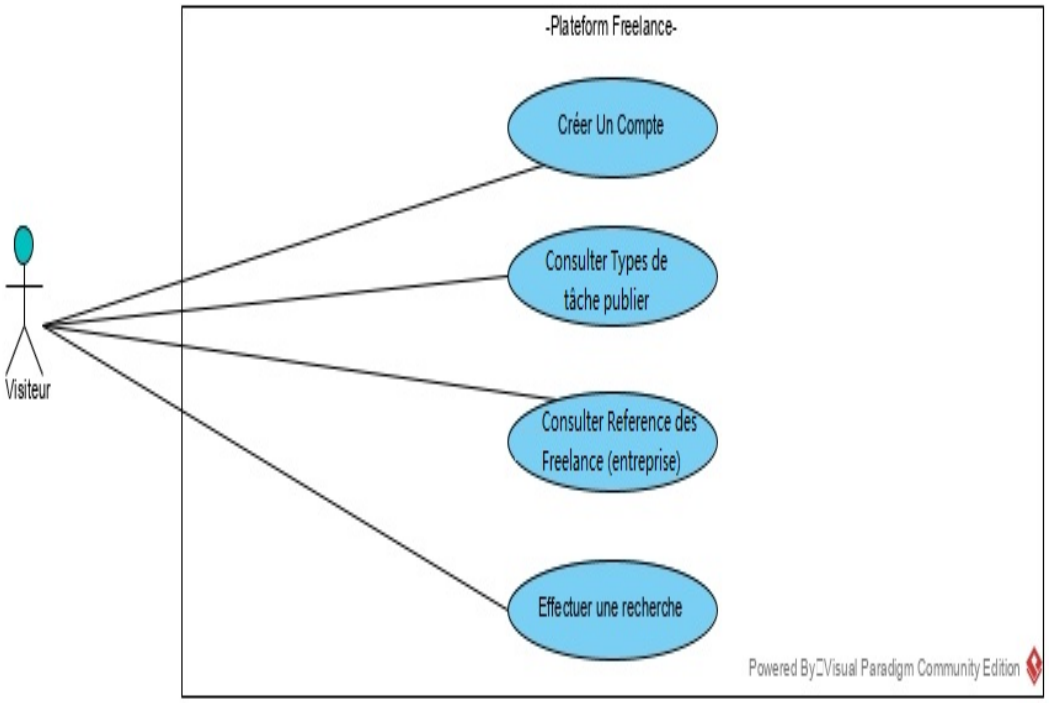

FIGURE 2.5 – Diagramme de cas d'utilisation –visiteur.

2.6.2.5. Diagramme de cas d'utilisation –commanditaire

La figure ci-dessous représente le diagramme de toutes les fonctionnalités offertes au commanditaire (client simple, entreprise) sur notre plateforme, on cite par exemple noter un freelancer, publier une tâche, contacter un freelancer et gérer son compte.

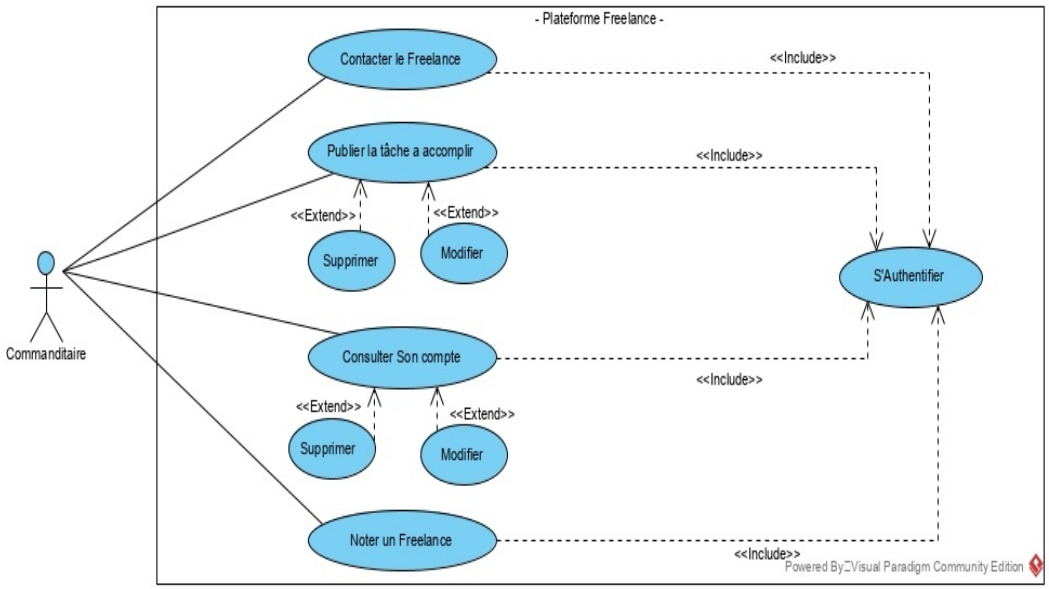

FIGURE 2.6 – Diagramme de cas d'utilisation –commanditaire

2.6.2.6. Diagramme de cas d'utilisation –freelance

La figure ci-dessous représente le diagramme de la fonctionnalité supplémentaire du freelancer sur notre plateforme, par rapport aux commanditaires, qui est l'enchérissement sur une tâche.

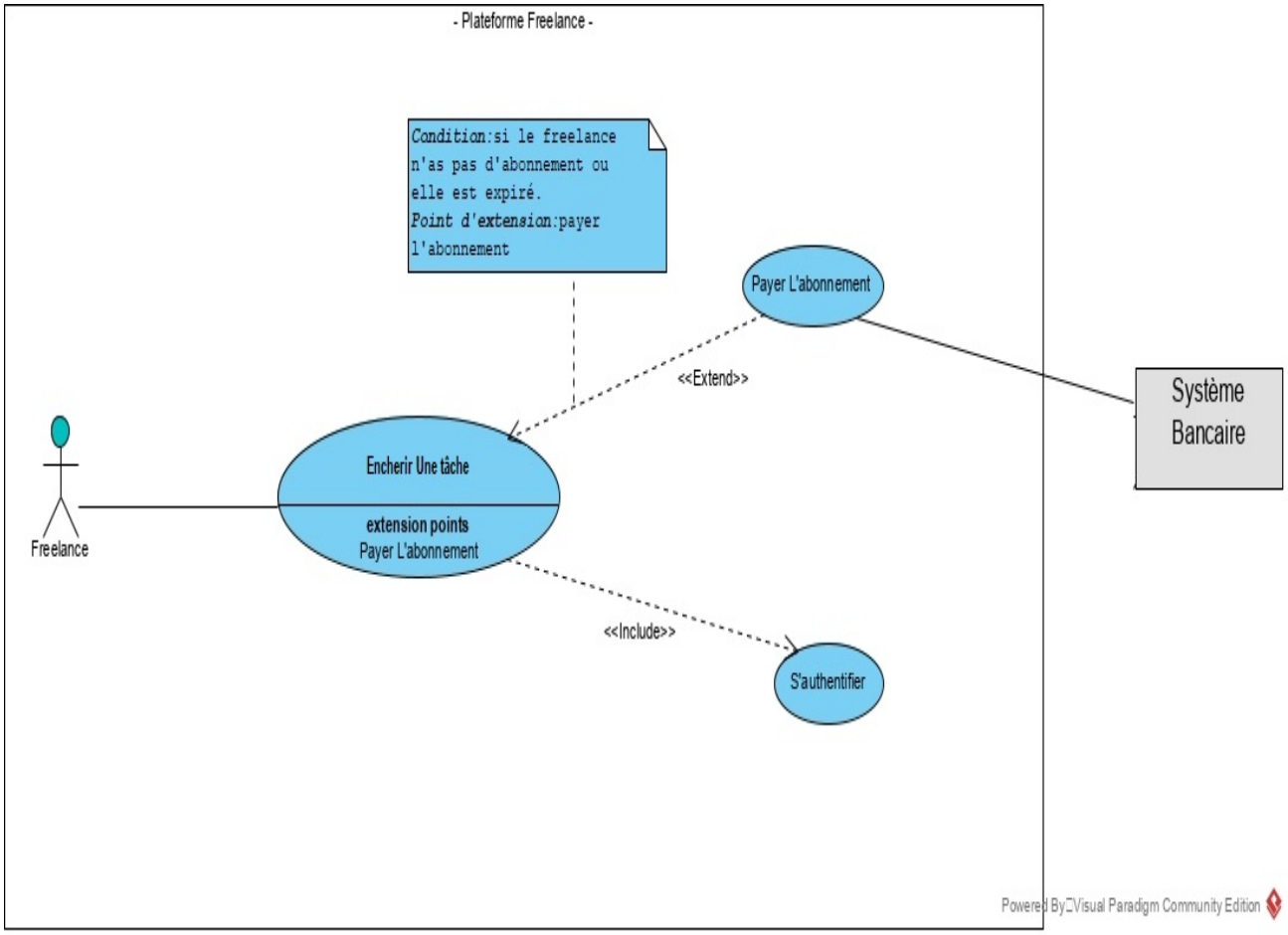

FIGURE 2.7 – Diagramme de cas d'utilisation –freelance

2.6.2.7. Diagramme de cas d'utilisation –administrateur simple

La figure ci-dessous représente le diagramme des fonctionnalités de l'administrateur de notre plateforme, on cite par exemple la gestion des tâches et des commanditaires ainsi que les freelances.

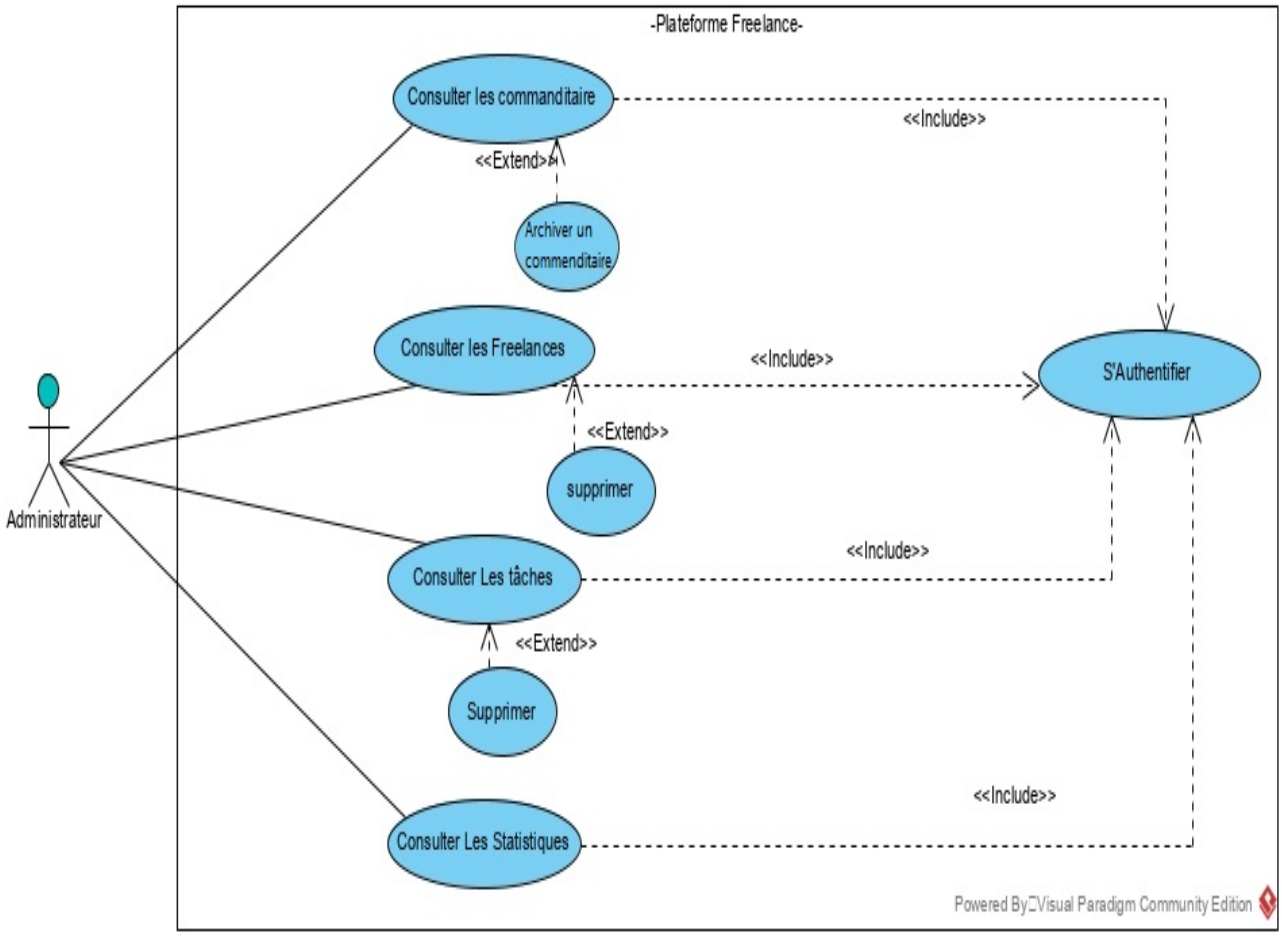

FIGURE 2.8 – Diagramme de cas d'utilisation –administrateur simple

2.6.2.8. Diagramme de cas d'utilisation –administrateur principal

La figure ci-dessous représente le diagramme de la fonctionnalité supplémentaire de l'administrateur principal de notre plateforme, par rapport aux administrateurs simple, qui est la gestion de ces derniers.

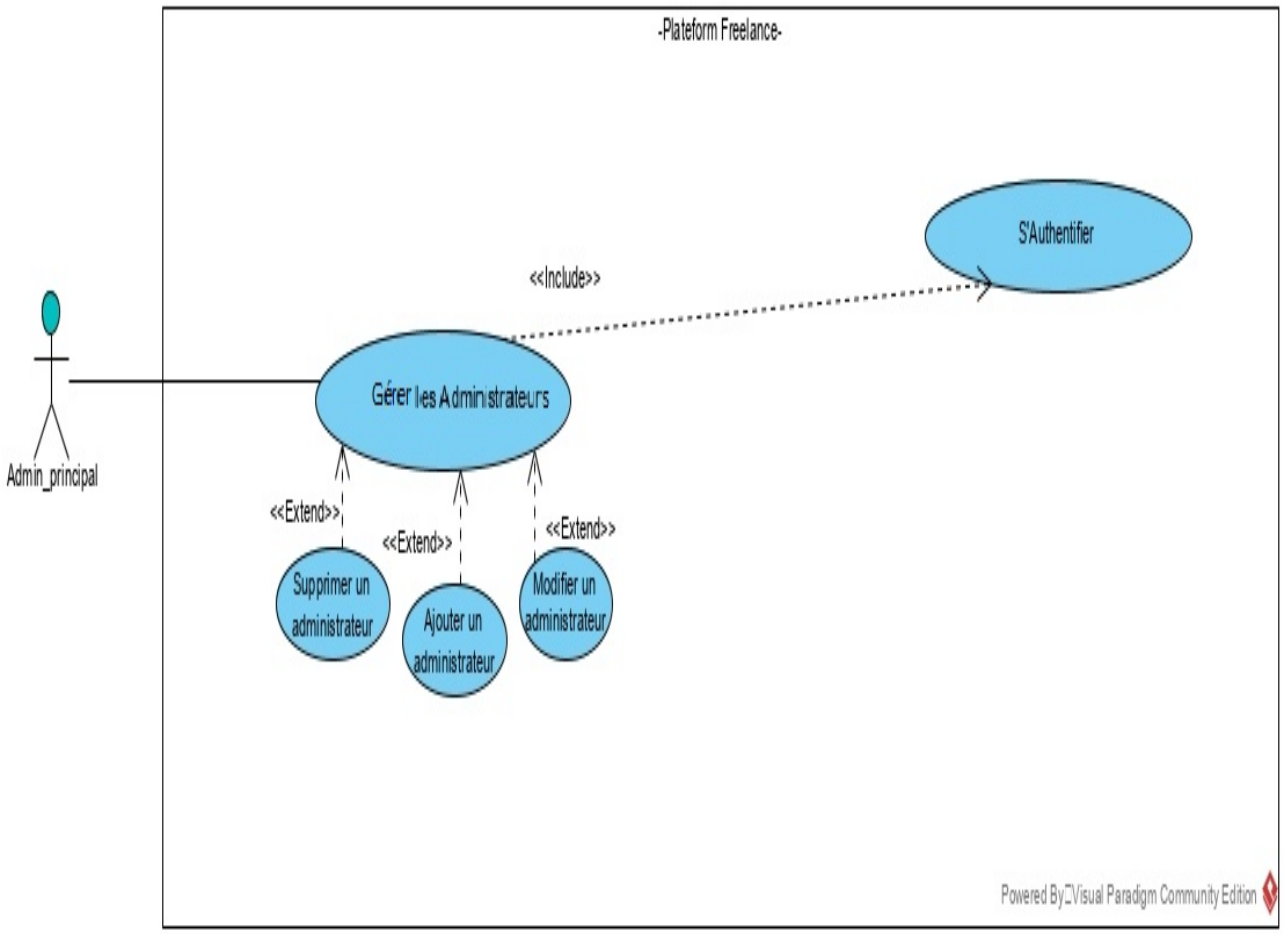

FIGURE 2.9 – Diagramme de cas d'utilisation –administrateur principal

2.6.2.9. Diagramme de cas d'utilisation –Globale

La figure ci -dessous représente le diagramme de cas d'utilisation global de notre système.

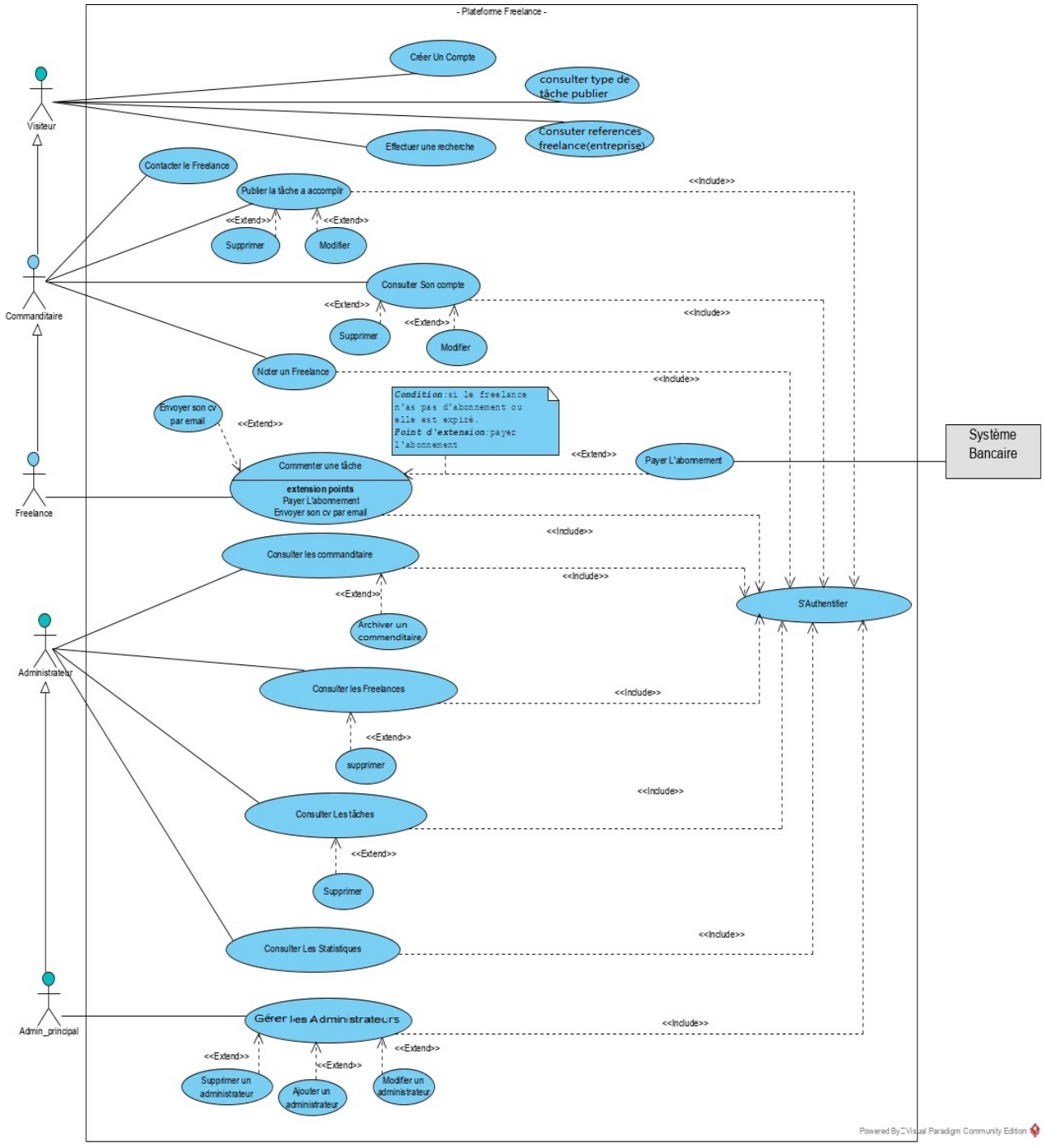

FIGURE 2.10 – Diagramme de cas d'utilisation –Globale

2.6.2.10. Description textuelle de quelques cas d'utilisation

Dans le tableau ci-dessous sont décrits les cas d'utilisation que nous avons choisi pour la suite de la modélisation.

Cas d'utilisation	Acteur	Description
Authentification.	Freelancer, Commanditaire, Administrateur, Administrateur principale.	Permets aux différents utilisateurs de s'authentifier à notre système.
Publier une tâche.	Commanditaire, Freelancer.	Permets au commanditaire et au freelance de publier leurs tâches selon la catégorie de leurs besoins et leurs budgets et les délais de réalisation de cette tâche.
Commenter la tâche.	Freelancer.	Permets au freelance d'enchérir (sous forme d'un commentaire) sur la tâche de son choix en spécifiant le prix et le délai de réalisation et les technologies à utiliser.
Consulter les tâche.	Freelancer, Commanditaire.	Permets au freelance et au commanditaire de consulter toutes les tâches publiées sur notre plateforme, ou de filtrer les tâches selon leurs catégories.
Consulter les enchères.	Freelancer, Commanditaire.	Permets aux freelance et commanditaire de consulter les enchères qui leur sont proposées par les freelances. Et les enchères seront affichées sauf si la tâche leur appartient.
Consulter la liste des administrateurs.	Administrateur principal.	Permets à l'administrateur principal de consulter, d'ajouter, modifier, supprimer les comptes des autres administrateurs.

TABLE 2.2 – Tableau décrit les cas d'utilisation choisi pour la suite de la modélisation.

Conclusion

A ce stade, nous avons rassemblé tous les éléments nécessaires pour entamer la conception des différentes parties de notre plateforme, tout ¸ça nous allons l'aborder dans le prochain chapitre.

Chapitre 3

Analyse et Conception

Introduction

Dans ce chapitre, nous allons procéder à l'établissement de deux phases importantes du de développement de notre système qui sont l'analyse des besoins et la conception. En effet, de ces dernières que dépendent la qualité et la cohérence du produit réalisé au développement. Dans ce qui suit, nous allons modéliser les différentes fonctionnalités du système sous forme des digrammes, qui vont nous permettre de dégager le modèle relationnel de ce dernier.

3.1. Analyse

3.1.1. Digramme de séquence système

Les diagrammes de séquences sont la représentation graphique des échanges entre les acteurs et le système selon un ordre chronologique. Pour passer du diagramme de cas d'utilisation aux diagrammes d'interaction, on doit passer par une étape intermédiaire : diagramme de séquence système[12].

Nous avons choisi de modéliser des diagrammes de séquences système où est décrit le scénario nominal du cas d'utilisation.

3.1.1.1. Diagramme de séquence système -Authentification

La figure décrit le diagramme de l'enchaînement séquentiel des échanges entre un utilisateur et le système lors de l'authentification (connexion) dans le cadre du scénario nominal.

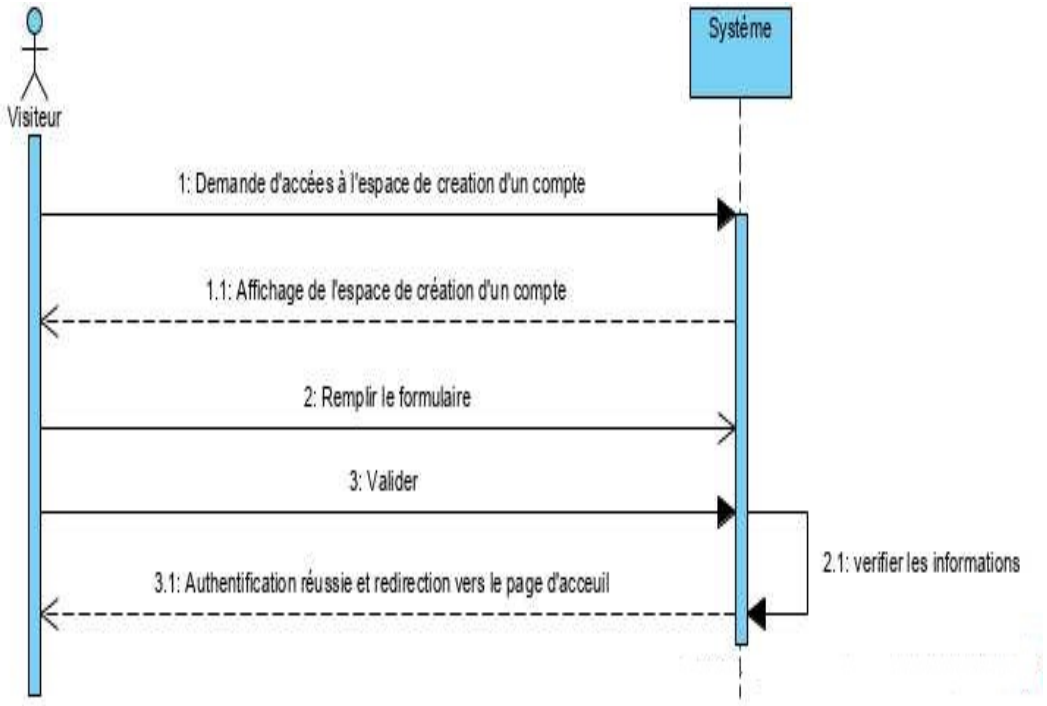

FIGURE 3.1 – Diagramme de séquence système -Authentification

3.1.1.2. Diagramme de séquence système – Créer un compte

La figure ci-dessous décrit le diagramme de l'enchainement séquentiel des échanges entre le système et un visiteur lorsque ce dernier veut créer un compte sur notre plateforme.

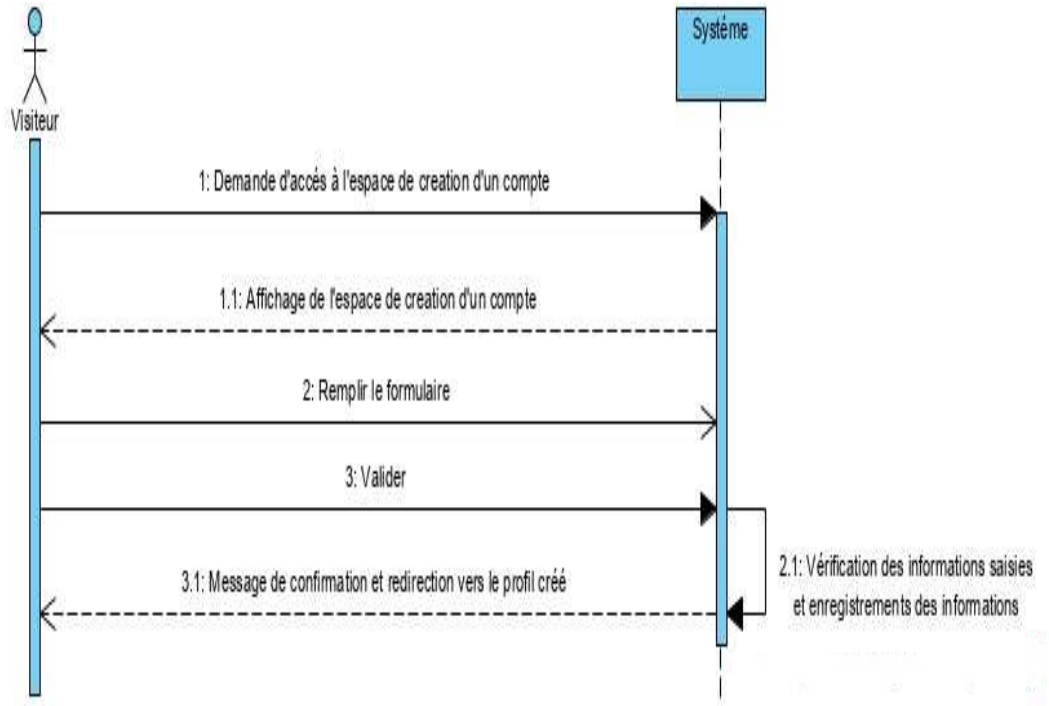

FIGURE 3.2 – Diagramme de séquence système – Créer un compte

3.1.1.3. Diagramme de séquence système -Publier une Tâche

La figure ci-dessous décrit le diagramme de l'enchaînement séquentiel des échanges entre un commanditaire et le système lors de la publication d'une tâche.

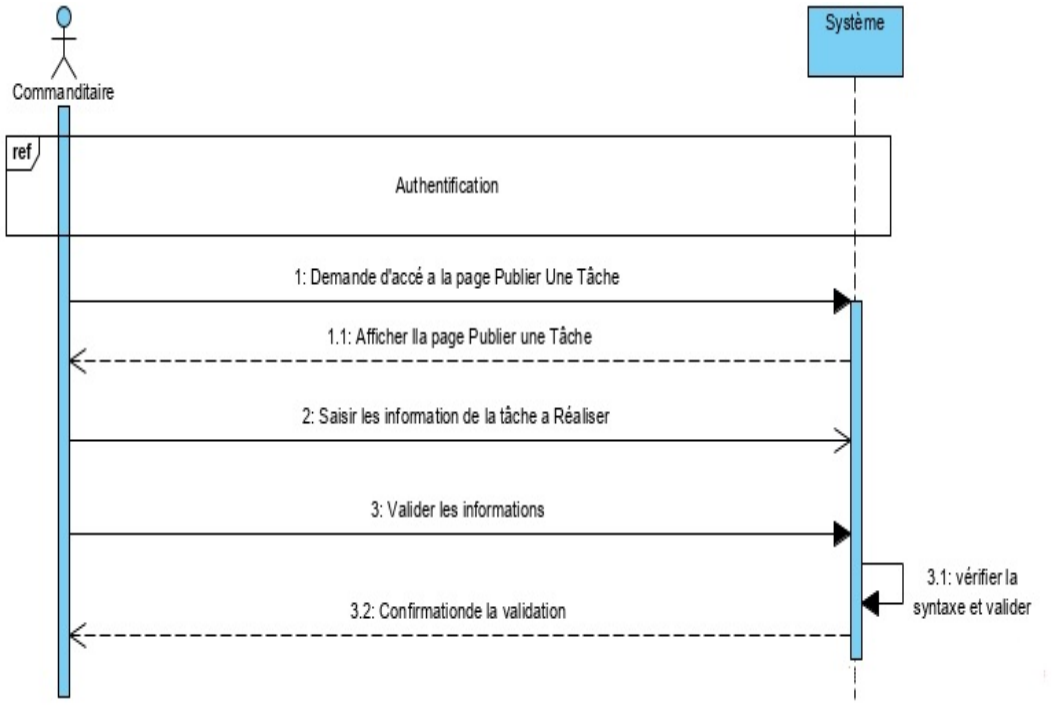

FIGURE 3.3 – Diagramme de séquence système -Publier une Tâche

3.1.1.4. Diagramme de séquence système -Commenter une Tâche

La figure ci-dessous décrit le diagramme de l'enchaînement séquentiel des échanges entre le système et un freelance lorsque ce dernier souhaite enchérir une tâche qui veut réaliser dans le cadre du scénario nominal.

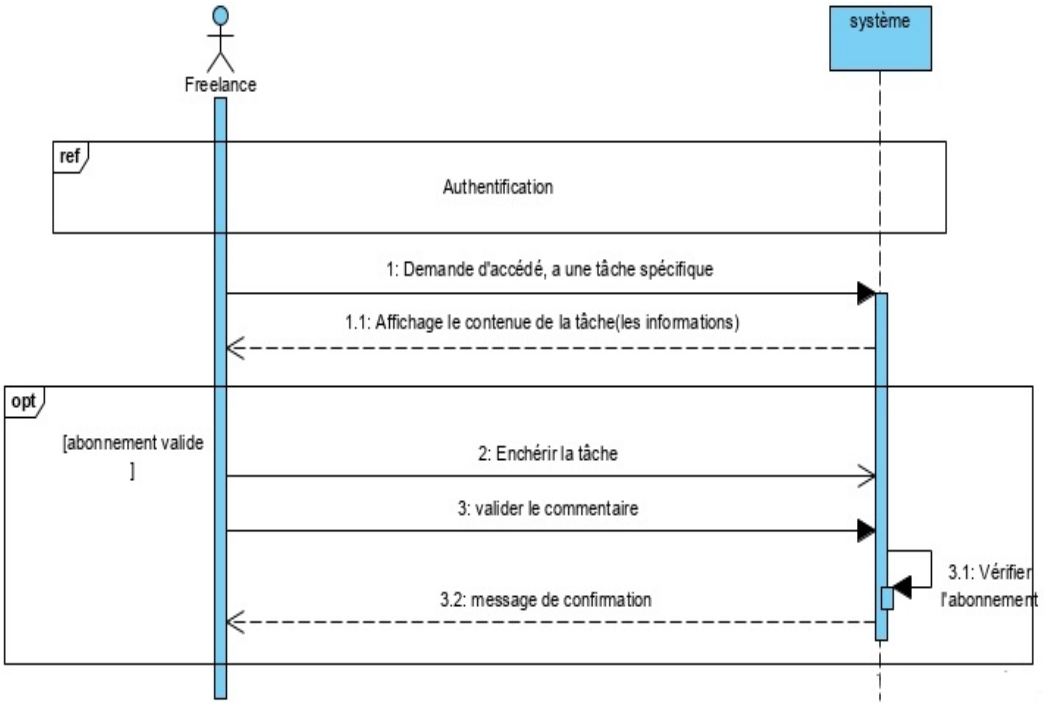

FIGURE 3.4 – Diagramme de séquence système -Commenter une Tâche

3.1.1.5. Diagramme de séquence système -Consulter Les Commentaires

La figure ci-dessous décrit le diagramme de l'enchaînement séquentiel des échanges entre un commanditaire et le système lors de la consultation des commentaires de sa tâche.

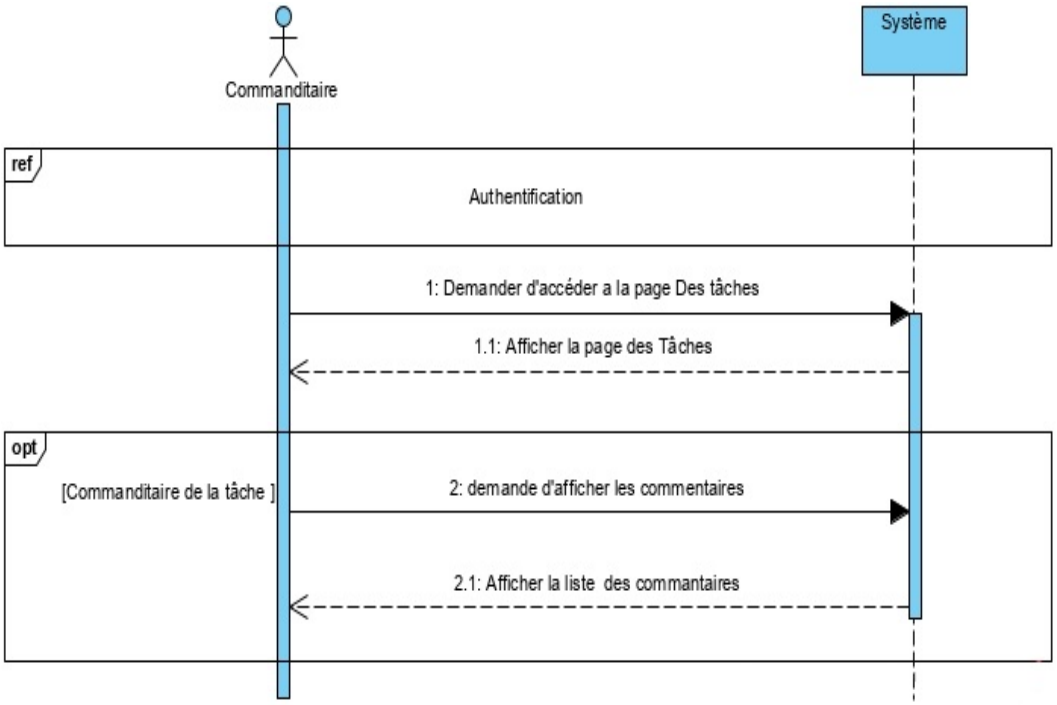

FIGURE 3.5 – Diagramme de séquence système -Consulter Les Commentaires

3.1.1.6. Diagramme de séquence système -Supprimer une tâche par administrateur -

La figure ci-dessous décrit le diagramme de l'enchaînement séquentiel des échanges entre un administrateur et le système lors de la suppression d'une tâche.

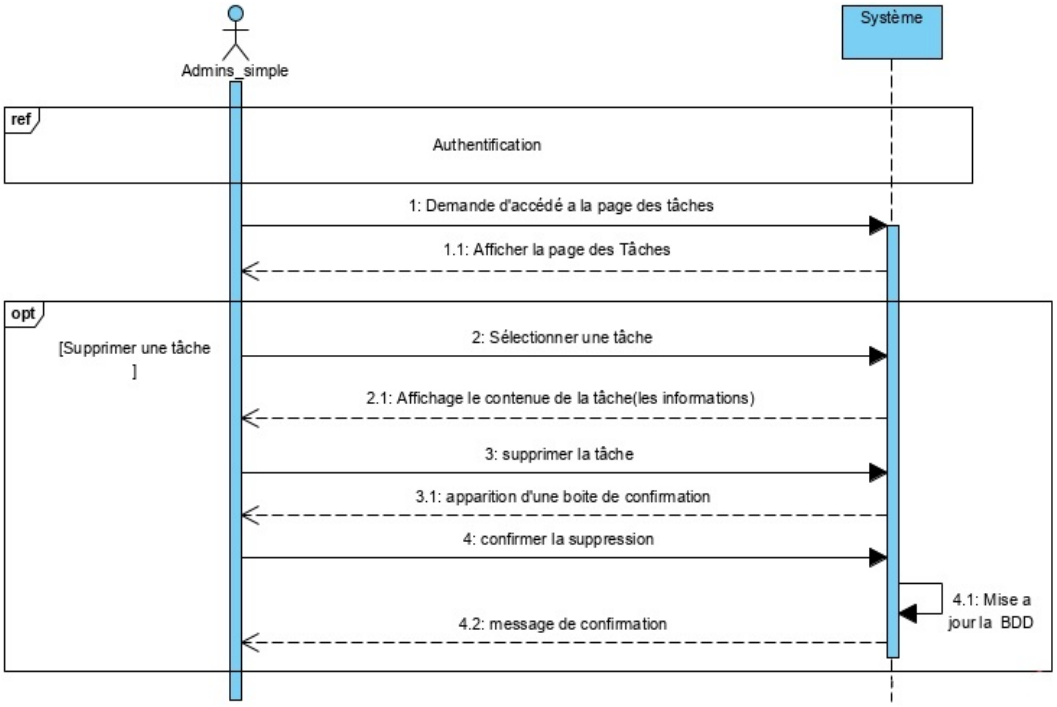

FIGURE 3.6 – Diagramme de séquence système -Supprimer une tâche par administrateur -

3.1.1.7. Diagramme de séquence - Gérer les administrateurs -

La figure ci-dessous décrit le diagramme de l'enchaînement séquentiel des échanges entre le système et l'administrateur principal lorsque ce dernier consulte la liste des administrateurs et éventuellement ajoute, modifie ou supprime un administrateur.

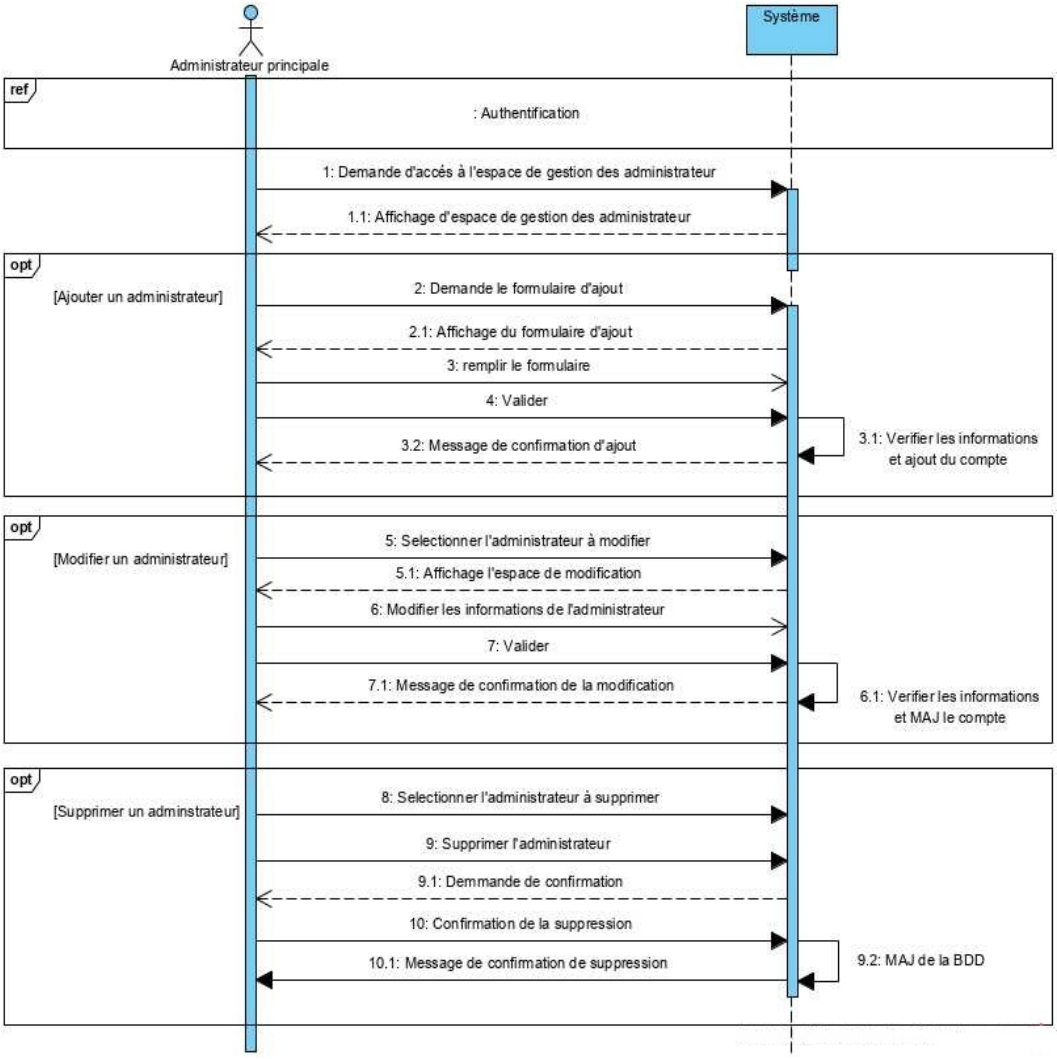

FIGURE 3.7 – Diagramme de séquence - Gérer les administrateurs -

3.2. Conception

3.2.1. Diagramme d'interaction

Un diagramme d'interaction est un diagramme de séquence système détaillé ou le système représenté par une boite noire est remplacé par un ensemble d'objets en interaction à savoir les dialogues, les contrôles et les entités tels que :[12]

— Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.

— Les dialogues peuvent interagir avec les contrôles.

— Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.

— Les entités ne peuvent interagir qu'entre elles.

Dans la suite seront présentés des diagrammes d'interactions ou seront décrits tous les scénarios possibles.

3.2.1.1. Diagramme d'interaction-Authentification

Ce diagramme ci-dessous décrit l'enchaînement des échanges entre l'utilisateur et les différents objets participant au cas d'utilisation traité « authentifier ».

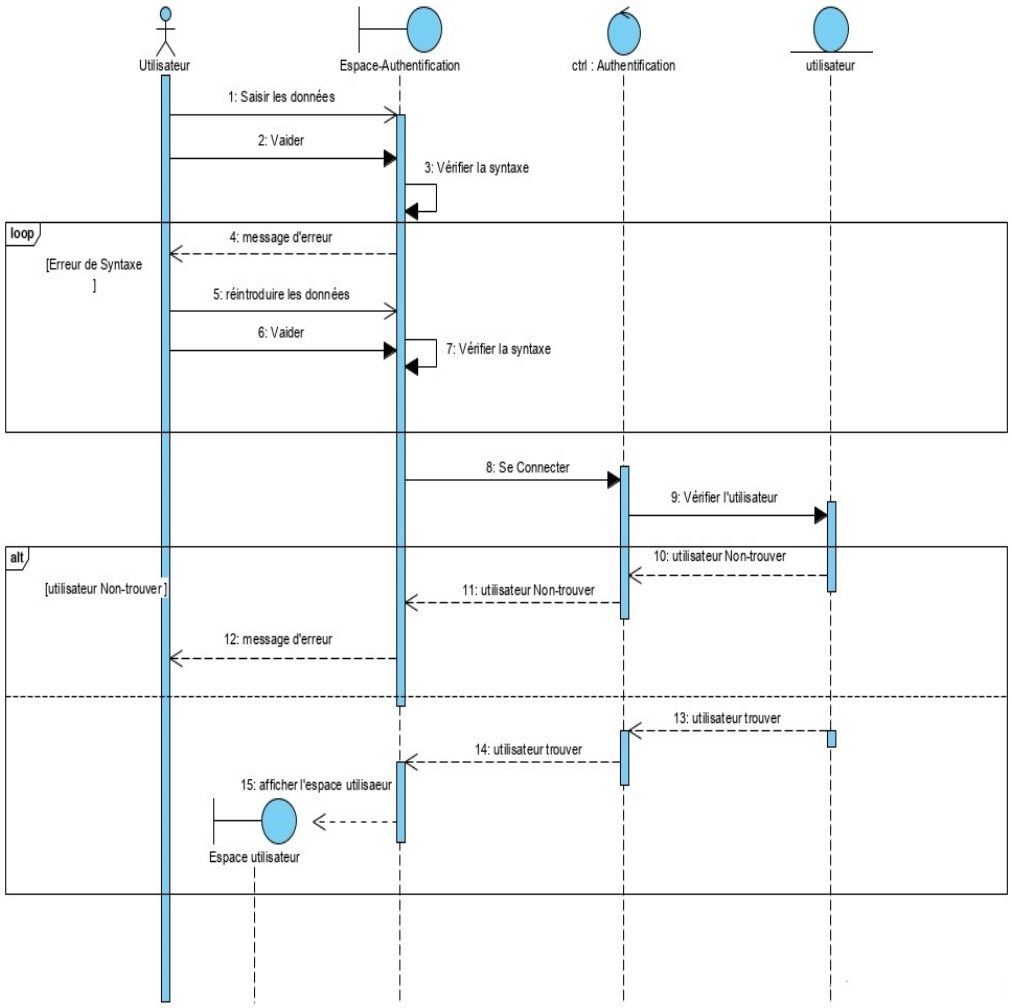

FIGURE 3.8 – Diagramme d'interaction-Authentification

3.2.1.2. Diagramme d'interaction- Gérer les administrateurs -

Le diagramme ci-dessous décrit l'enchaînement des échanges entre l'administrateur principal et les différents objets participant au cas d'utilisation «gérer les administrateurs».

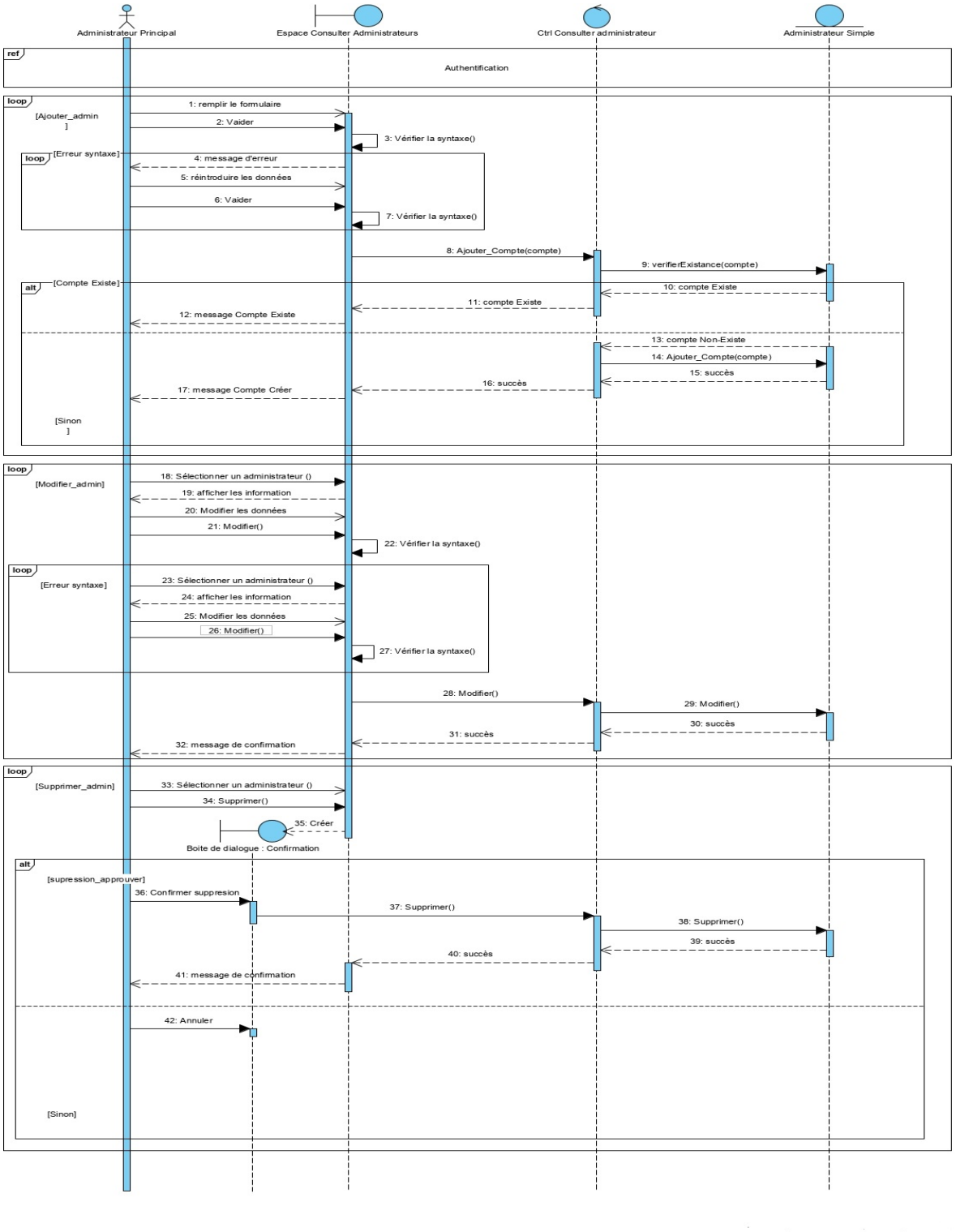

FIGURE 3.9 – Diagramme d'interaction- Gérer les administrateurs -

3.2.1.3. Diagramme d'interaction-Publier une tâche

Ce diagramme ci-dessous décrit l'enchaînement des échanges entre le commanditaire et les différents objets participant au cas d'utilisation traité « publier une tâche ».

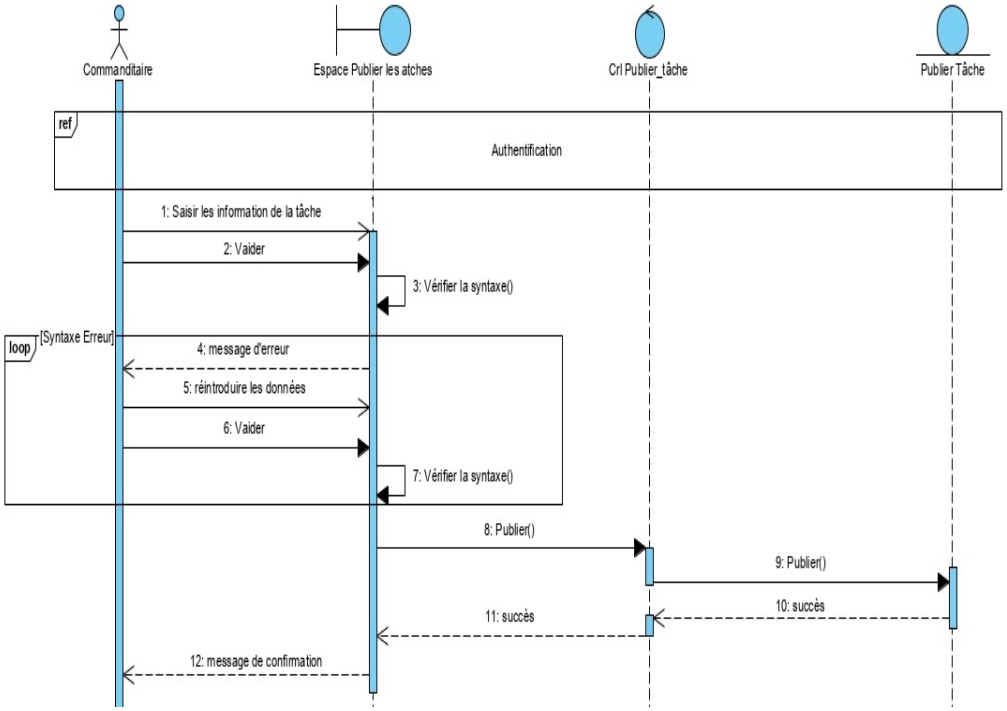

FIGURE 3.10 – Diagramme d'interaction-Publier une tâche-

3.2.1.4. Diagramme d'interaction-créer un compte-

Ce diagramme ci-dessous décrit l'enchaînement des échanges entre l'utilisateur et les différents objets participant au cas d'utilisation « créer un compte »

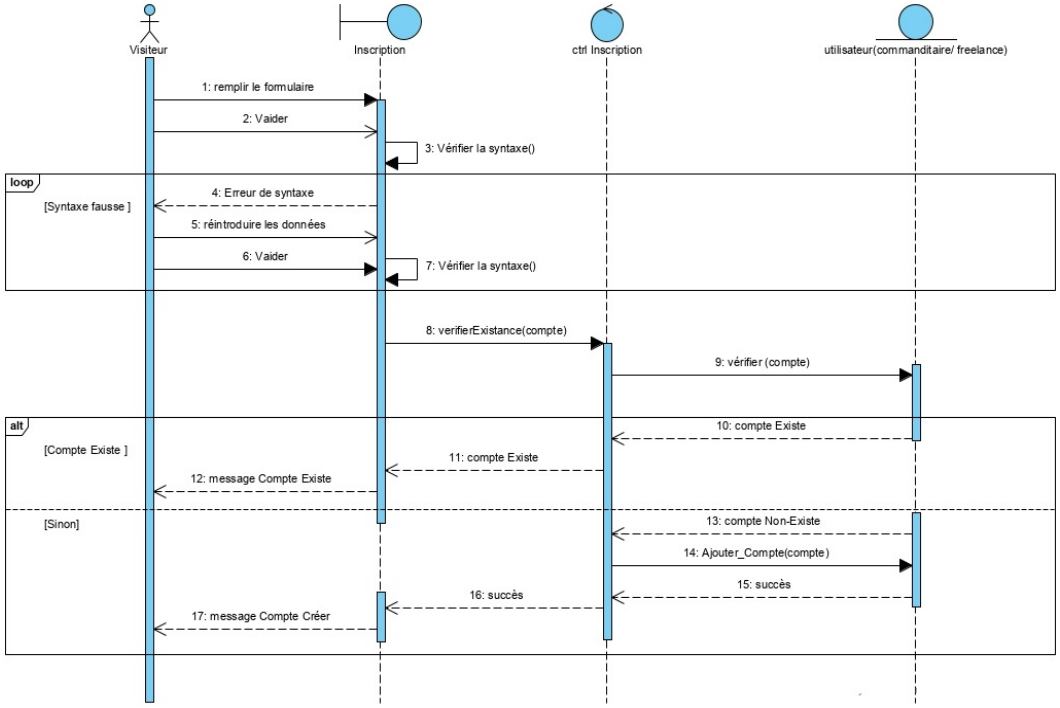

FIGURE 3.11 – Diagramme d'interaction-créer un compte-

3.2.1.5. Diagramme d'interaction -Commenter une tâche -

Le diagramme ci-dessous décrit le diagramme d'interaction entre un freelance et les différents objets participant au cas d'utilisation traité « enchérir sur une tâche ».

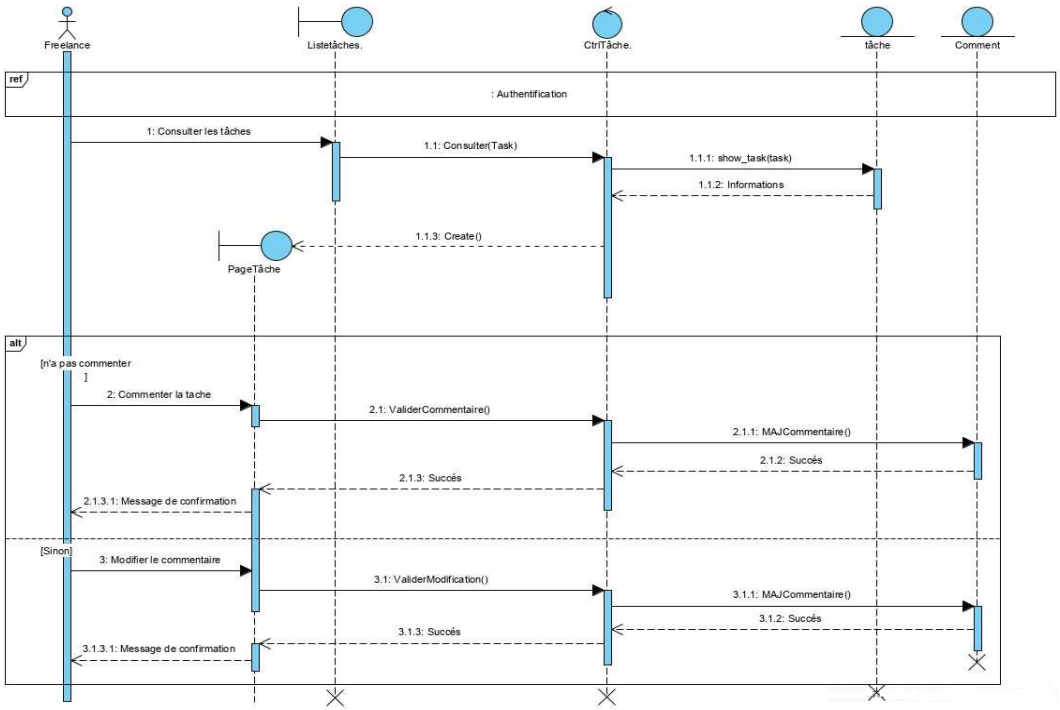

FIGURE 3.12 – Diagramme d'interaction -Commenter une tâche -

3.2.2. Diagramme de classe

Le diagramme de classe représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système[12].

3.2.2.1. Description des classes

Administrateur principal	Représente la classe de l'administrateur qui gère tous les administrateurs de l'application.
Administrateur simple	Regroupe les administrateurs qui gèrent la plateforme.
Utilisateur	Regroupe les informations des freelancers et des commanditaires ayant interagi avec notre plateforme.
Tâche	Regroupe les informations des tâches publiées pour être réalisées.
Catégorie	Regroupe les catégories liées aux tâches à publier.
Commentaire	Regroupe les commentaires des freelancers sur chaque tâche.
Abonnement	Regroupe les informations sur l'abonnement de chaque freelancer.
Message	Regroupe les messages échangés entre les utilisateurs.

TABLE 3.1 – Tableau représente la description des classes.

3.2.2.2. Règles de passage du diagramme de classe au modèle relationnel

Le passage du diagramme de classe au modèle relationnel se fait selon des règles suivantes :

— **Règle 1(Transformation des classes) :** Chaque classe du diagramme de classe devient une relation, il faut choisir un attribut de la classe pouvant jouer le rôle de clé (le rôle de l'identifiant).

Transformation des associations : Nous distinguons trois familles d'associations :

— **Règle 2 (Association un-à-plusieurs) :** Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.

— **Règle 3 (Association plusieurs-à-plusieurs) :** La classe-association devient une relation. La clé primaire de cette relation est la concaténation des identifiâtes

des classes connectées à l'association, chaque attribut devient clé étrangère si la classe connectée dont il provient devient une relation en vertu de la règle 1. Les attributs de l'association (classe-association) doivent être ajoutés à la nouvelle relation. Ces attributs ne sont ni clé primaire, ni clé étrangère.

— **Règle 4 (Association un-à-un) :** Il faut ajouter un attribut de type clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association. Si les deux multiplicités minimales sont à un, il est préférable de fusionner les deux classes en une seule

— **Règle 5 (Transformation de l'héritage) :** Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- **Décomposition par distinction :** Il faut transformer chaque sous-classe en une relation, la clé primaire de la surclasse, migre dans la relation issue de la sous-classe(s) et devient à la fois clé primaire et clé étrangère.
- **Décomposition descendante :** S'il existe une contrainte de totalité ou de partition Sur l'association d'héritage, il est possible de ne pas traduire la relation issue de la Surclasse. Il faut alors faire migrer tous ses attributs dans la(les) relation(s) issue(s) de la(des) sous-classe(s).
- **Décomposition ascendante :** Il faut supprimer la relation issue de la sous-classe et Faire migrer les attributs dans la relation issue de la surclasse.

3.2.2.3. Règles de passage du diagramme de classe au modèle relationnel

Le passage du diagramme de classe au modèle relationnel se fait selon des règles suivantes :

— **Règle 1(Transformation des classes) :** Chaque classe du diagramme de classe devient une relation, il faut choisir un attribut de la classe pouvant jouer le rôle de clé (le rôle de l'identifiant).

Transformation des associations : Nous distinguons trois familles d'associations :

— **Règle 2 (Association un-à-plusieurs) :** Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.

— **Règle 3 (Association plusieurs-à-plusieurs) :** La classe-association devient une relation. La clé primaire de cette relation est la concaténation des identifiâtes des classes connectées à l'association, chaque attribut devient clé étrangère si la classe connectée dont il provient devient une relation en vertu de la règle 1. Les attributs de l'association (classe-association) doivent être ajoutés à la nouvelle relation. Ces attributs ne sont ni clé primaire, ni clé étrangère.

— **Règle 4 (Association un-à-un) :** Il faut ajouter un attribut de type clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association. Si les deux multiplicités minimales sont à un, il est préférable de fusionner les deux classes en une seule

— **Règle 5 (Transformation de l'héritage) :** Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- **Décomposition par distinction :** Il faut transformer chaque sous-classe en une relation, la clé primaire de la surclasse, migre dans la relation issue de la sous-classe(s) et devient à la fois clé primaire et clé étrangère.
- **Décomposition descendante :** S'il existe une contrainte de totalité ou de partition Sur l'association d'héritage, il est possible de ne pas traduire la relation issue de la Surclasse. Il faut alors faire migrer tous ses attributs dans la(les) relation(s) issue(s) de la(des) sous-classe(s).
- **Décomposition ascendante :** Il faut supprimer la relation issue de la sous-classe et Faire migrer les attributs dans la relation issue de la surclasse.

Le diagramme ci-dessous représente le diagramme de classe de notre Plateforme Freelance :

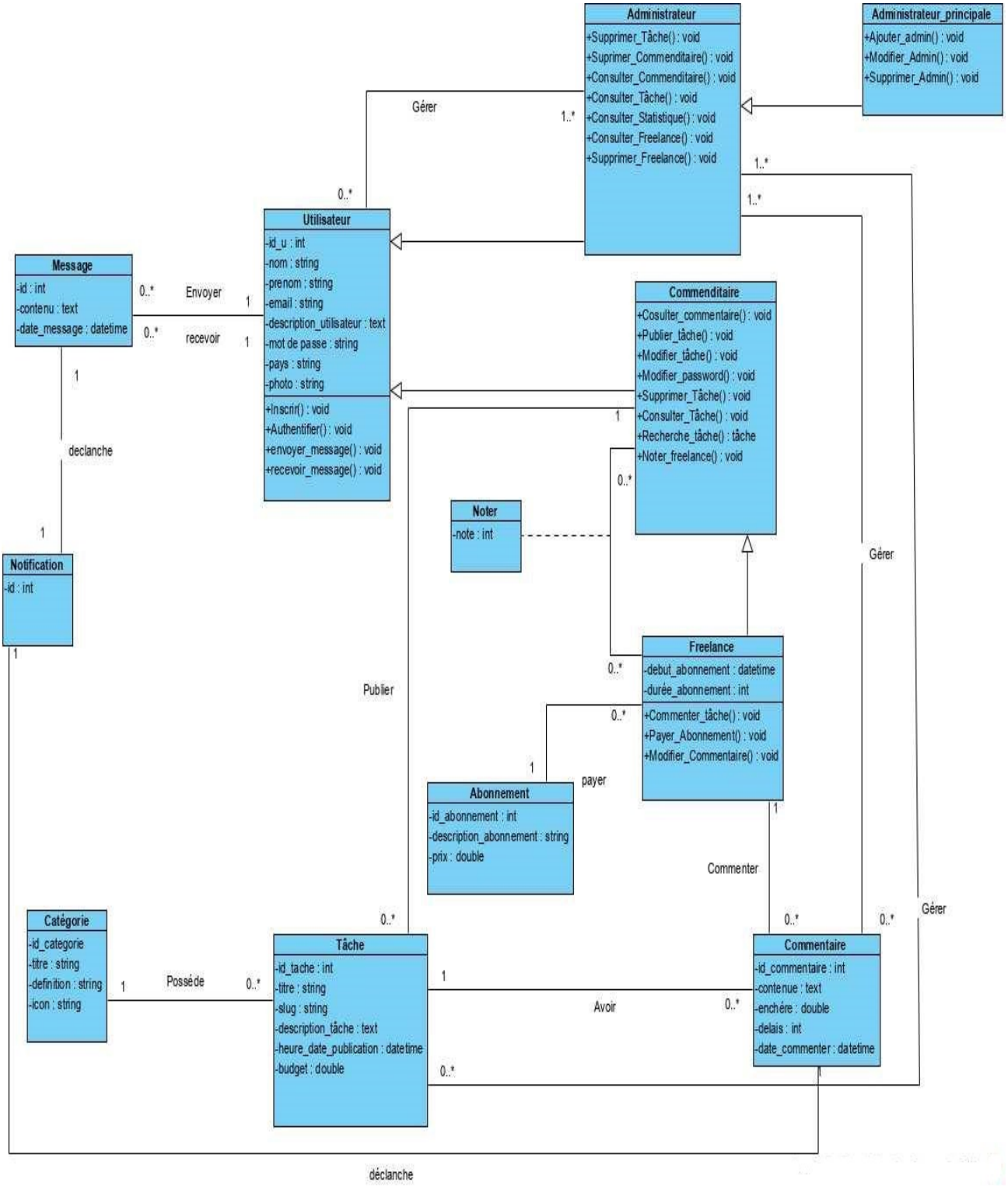

FIGURE 3.13 – Diagramme de classe

3.2.2.4. Dictionnaire de données

Dans le tableau ci-dessous sont décrites et expliqués toutes les données qui sont relatives aux classes de notre système.

Classe	Attribut	Description	Type	Taille
Utilisateur	Id_u	Identifiant de l'utilisateur	Int	11
	Nom	Nom de l'utilisateur	Varchar	30
	Prénom	Prénom de l'utilisateur	Varchar	30
	Email	Adresse email de l'utilisateur	Varchar	50
	Description_utilisateur	Description de l'utilisateur (ses compétences, ses besoin . . .)	Varchar	300
	Mot de passe	Mot de passe de l'utilisateur	Varchar	20
	Pays	Nationalité de l'utilisateur	Varchar	20
	Photo	Photo de profil de l'utilisateur	Varchar	250
Freelance	Debut_abonnement	La date du paiement de l'abonnement	DateTime	jj/mm/aaaa
	Durée_d'abonnement	Durée de l'abonnement (nombres de mois)	Int	10
Abonnement	Id_abonnement	Identifiant de l'abonnement	Int	11
	Description	Description d'un abonnement	Varchar	250
	Prix	Prix a payer pour un abonnement	Float	6
Tâche	Id_tâche	Identifiant de la tâche	Int	11
	Titre	Titre de la tâche	Varchar	15
	Slug	Liens de la tâche	Varchar	15
	Description	Description de la tâche	Varchar	/
	Heure_date_publication	Date et heure de la publication de la tâche.	DateTime	/

Classe	Attribut	Description	Type	Taille
	Budget	Le budget que le commanditaire peut offrir au freelance qui réalise la tâche	Float	10
Catégorie	Id_categorie	Identifient de la catégorie	Int	11
	Titre	Titre de la catégorie	Varchar	15
	Définition	Définition de la catégorie	Varchar	100
	Icône	Icône relative à une catégorie	Varchar	200
Commentaire	Id_commentaire	Identifiant du commentaire	Int	11
	Contenue	Le texte du commentaire (le freelance définit comment il va réaliser la tâche et les technologies qui va utiliser)	Varchar	300
	Enchère	Prix que le freelance propose au commanditaire pour réaliser sa tâche	Float	10
	Délais	Délais de la réalisation de la tâche (par jours)	Int	5
	Date_commentaire	Date de création du commentaire	DateTime	/
Notification	Id_notification	Identifiant de la notification	Int	11
Message	Id_message	Identifiant du message	Int	11
	Contenu	C'est le contenu du message	Varchar	300
	Date_message	Date de l'envoi et de réception du message	DateTime	/

TABLE 3.2: Tableau qui représente les données qui sont relatives aux classes de notre système.

3.2.2.5. Model relationnel du diagramme de classe

— Utilisateur (id_U, nom, prenom, email, description_U, mote_de_passe, pays, photo, debut_abnnement, fin_abnnement, #rôle, # id_msg, #id_abnnement).

— Message (id_msg, id_msg, contenu, date_msg, #id_notification).

— Tâche (id_tache, titre, slug, description_tache, heur_date_publication, budget, #id_categorie, #id_U).

— Catégorie (id_categorie, titre, definition, icôn).

— Note (id_U, note, #id_u).

— Abonnement (id_abonnemet, description_abonnmnt, prix).

— Commentaire (id_commentaire, Enchérir, délais, date_commentaire, #id_notification).

3.2.3. Organigramme de la plateforme

La figure ci-dessous représente le schéma de navigation sur la plateforme.

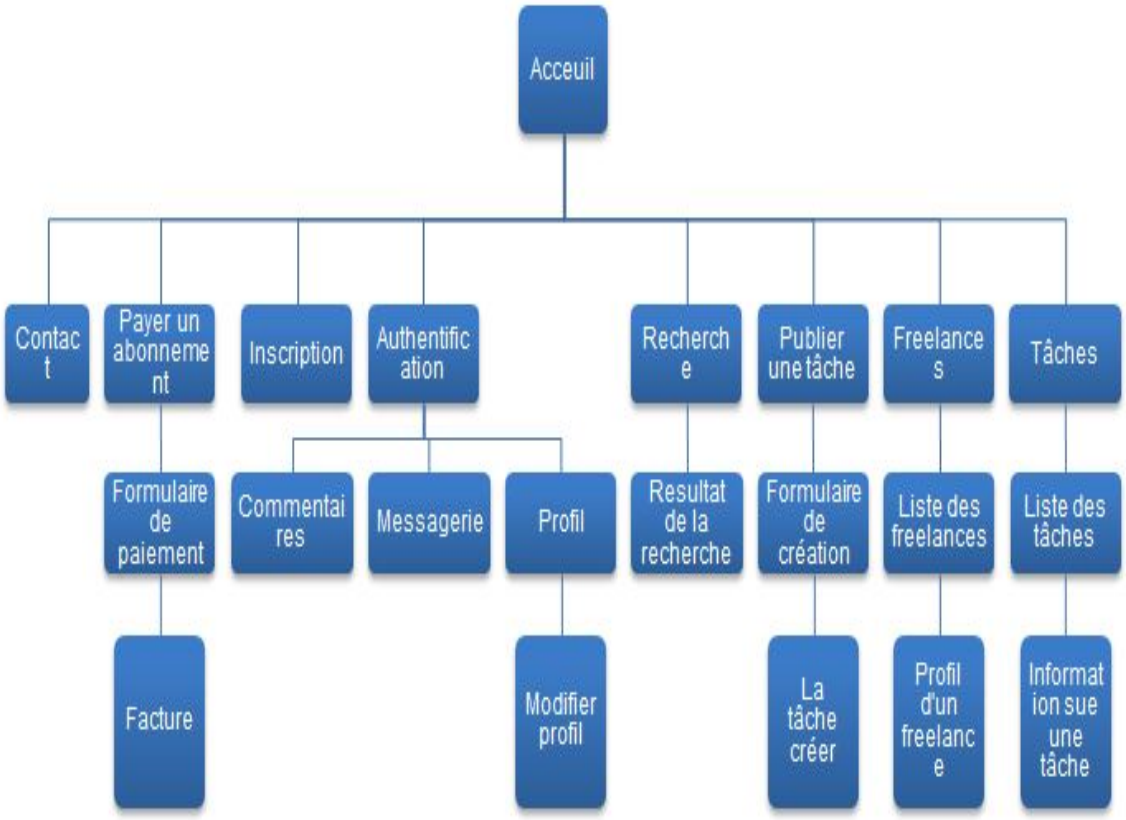

FIGURE 3.14 – Organigramme de la plateforme

Conclusion

Dans ce chapitre on a vu les différents diagrammes de la conception commençant par le diagramme de séquence système passant au diagramme plus détaller qui est le diagramme d'interaction, et en dernier le diagramme de classe après avoir appliqué les règles de passage du diagramme de classe au modèle relationnel et a la fin on a schématiser Notre platforme par un organigramme.

Chapitre 4

Réalisation

Introduction

Dans ce chapitre nous allons présenter les captures d'interfaces et on va expliquer brièvement les interfaces principales suivie d'une description pour chaque capture.

4.0.1. Principales captures d'interfaces

4.0.1.1. Interface D'Accueil

La figure ci-dessous représente l'interface vitrine de notre plat-forme, il s'agit de la première page qui apparait à l'utilisateur, elle englobe les principales fonctionnalités de notre plat forme et joue le rôle de portail de cette dernière.

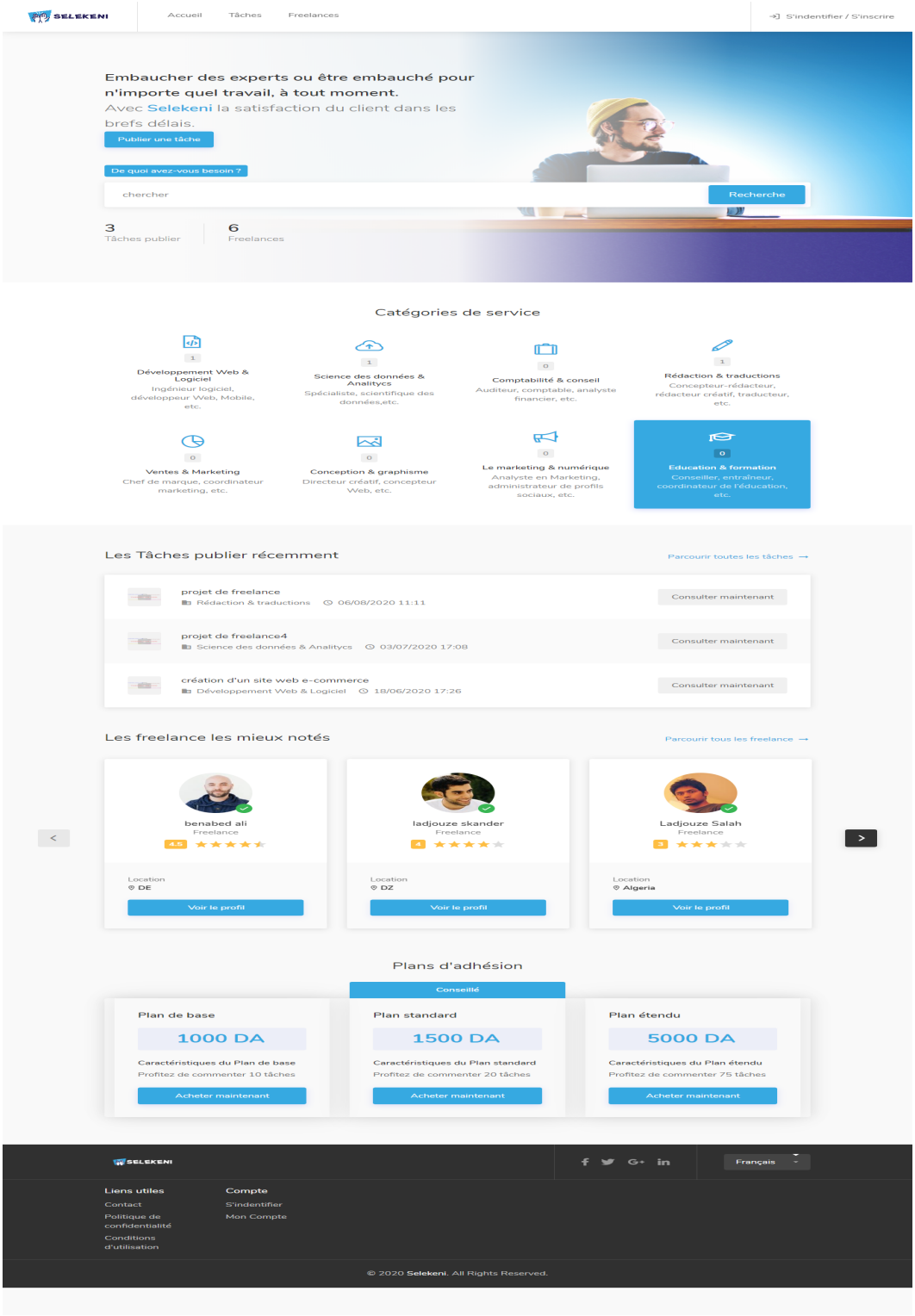

FIGURE 4.1 – Interface D'Accueil.

4.0.1.2. Interface de connexion

La figure ci-dessous représente l'interface pour les utilisateurs déjà inscrit à notre platforme.

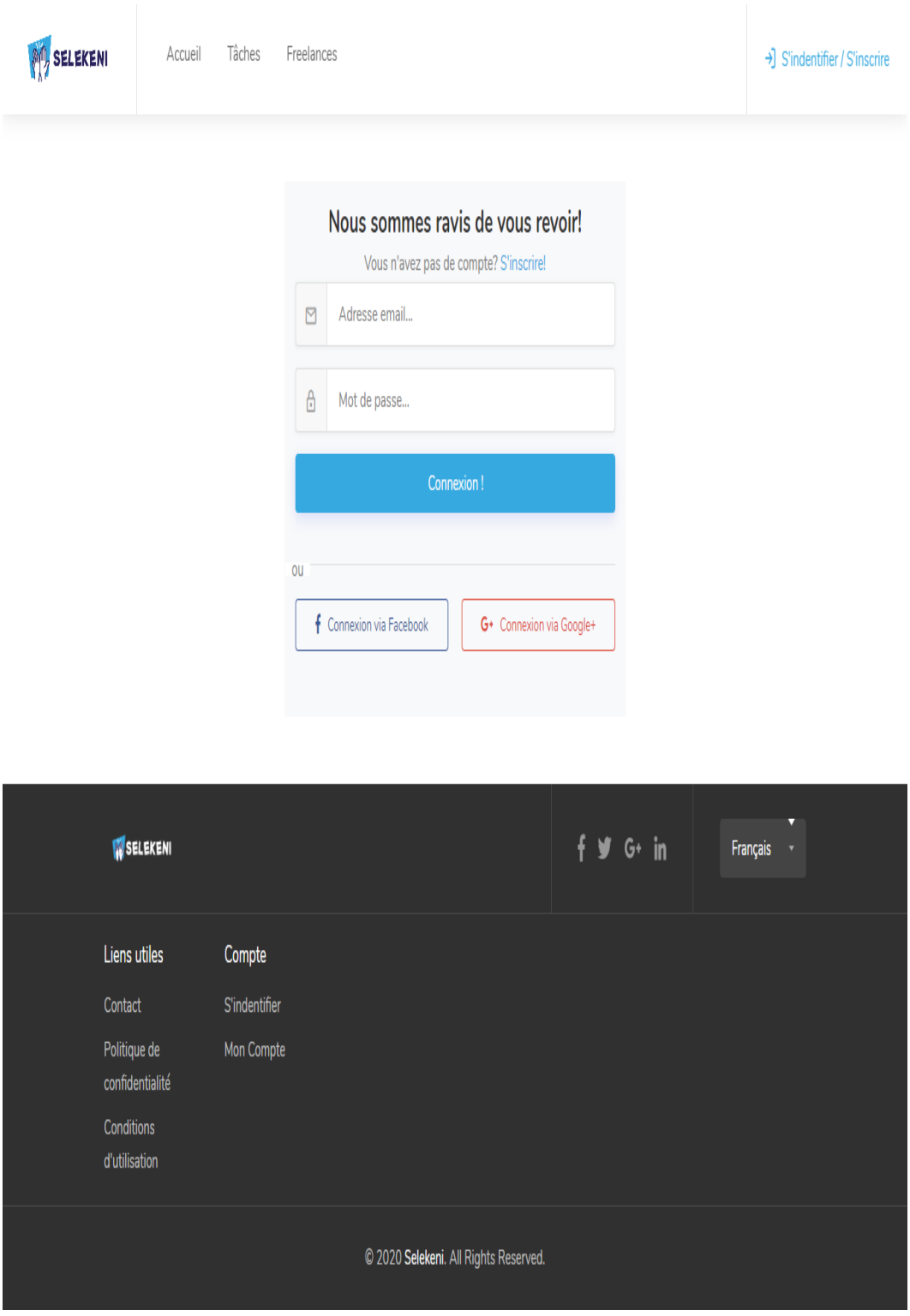

FIGURE 4.2 – Interface de connexion

4.0.1.3. Interface de la liste des freelances

La figure ci-dessous représente l'interface de la liste des freelances inscrits sur notre platforme.

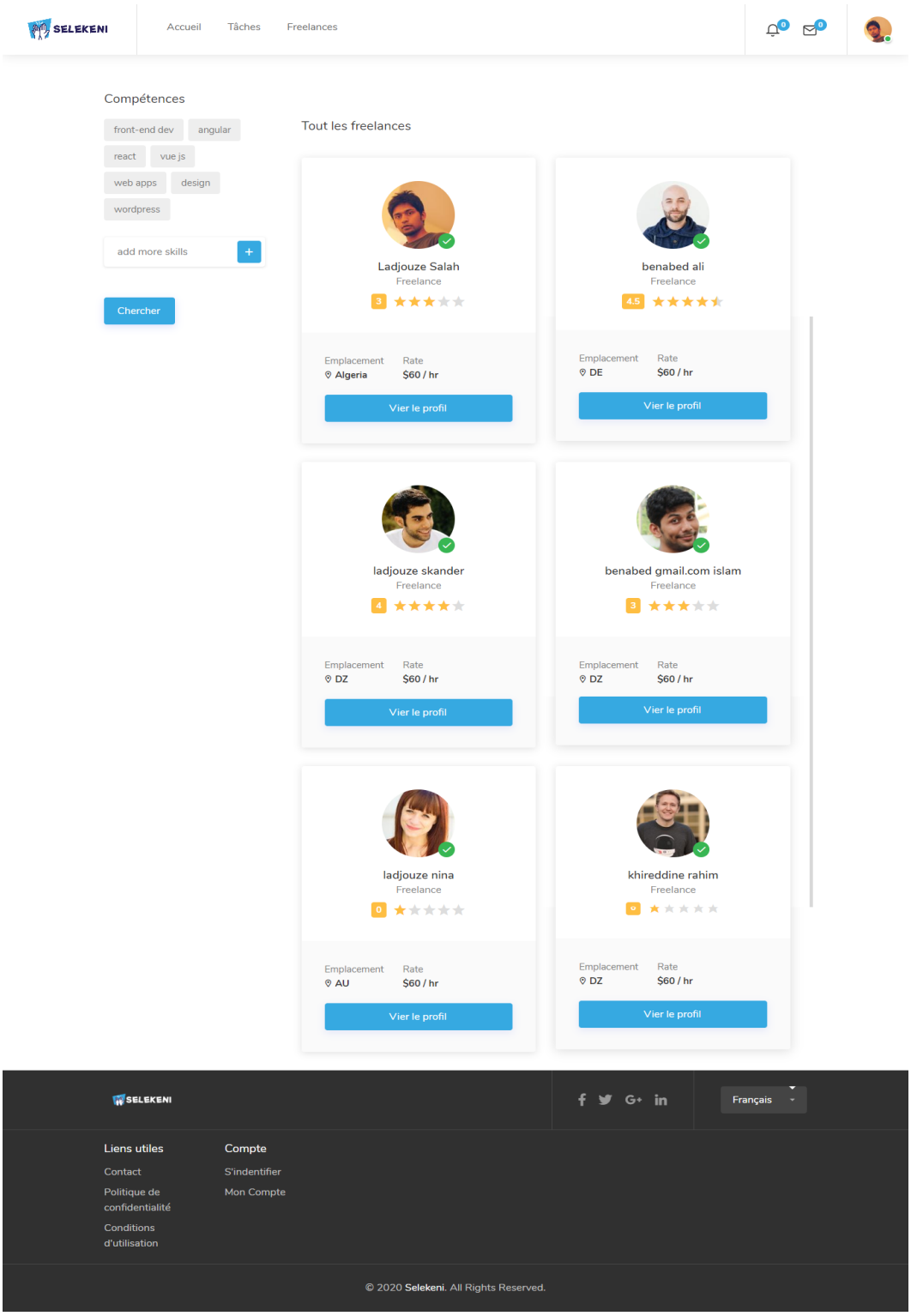

FIGURE 4.3 – Interface de la liste des freelancers

4.0.1.4. Interface d'une tâche

la figure ci-dessus représente l'interface concernant les informations d'une tâche précise.

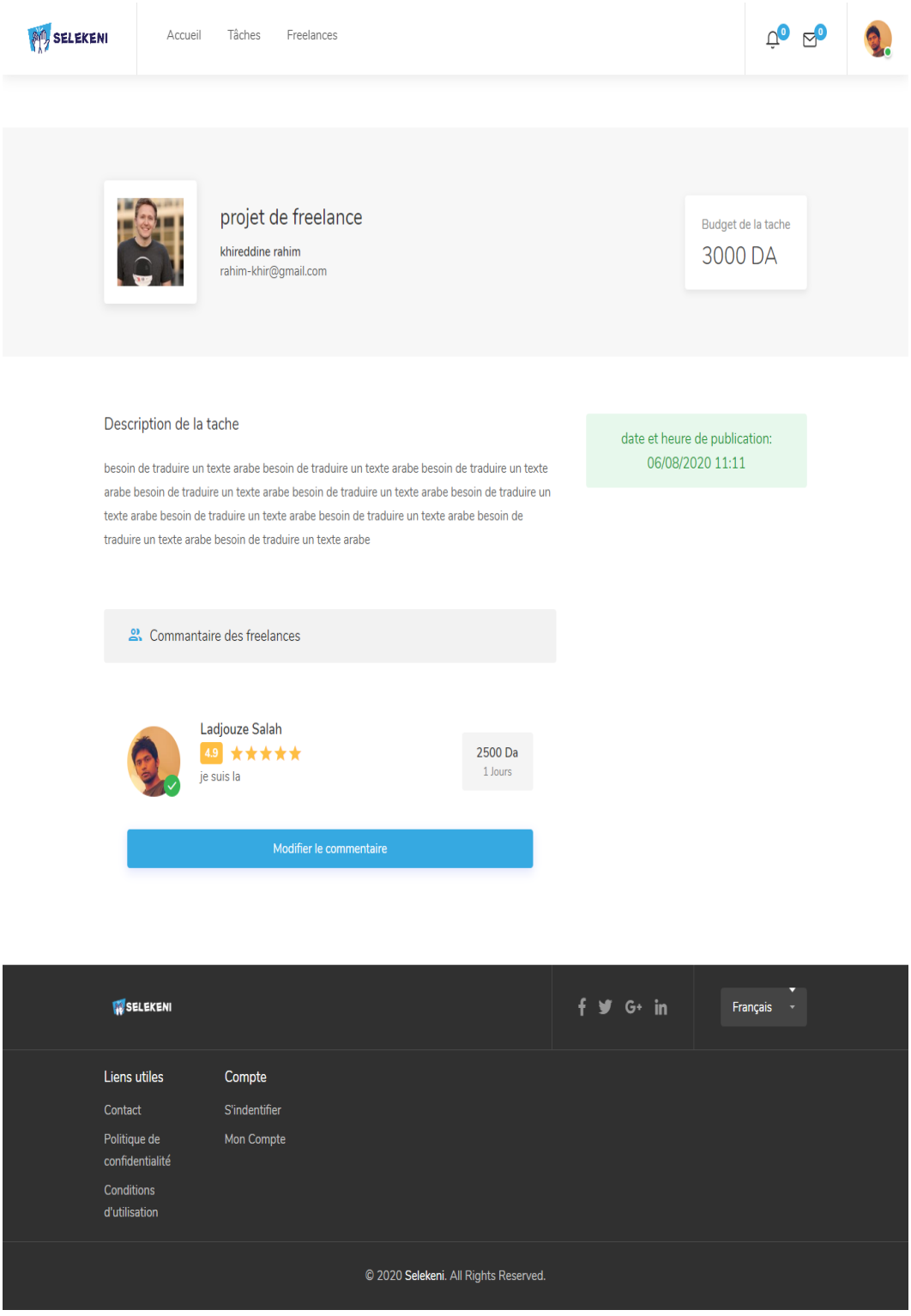

FIGURE 4.4 – Information sur la tâche

4.0.1.5. Interface du profil d'un utilisateur

La figure ci-dessus représente l-interface d'un utilisateur et ses coordonnées.

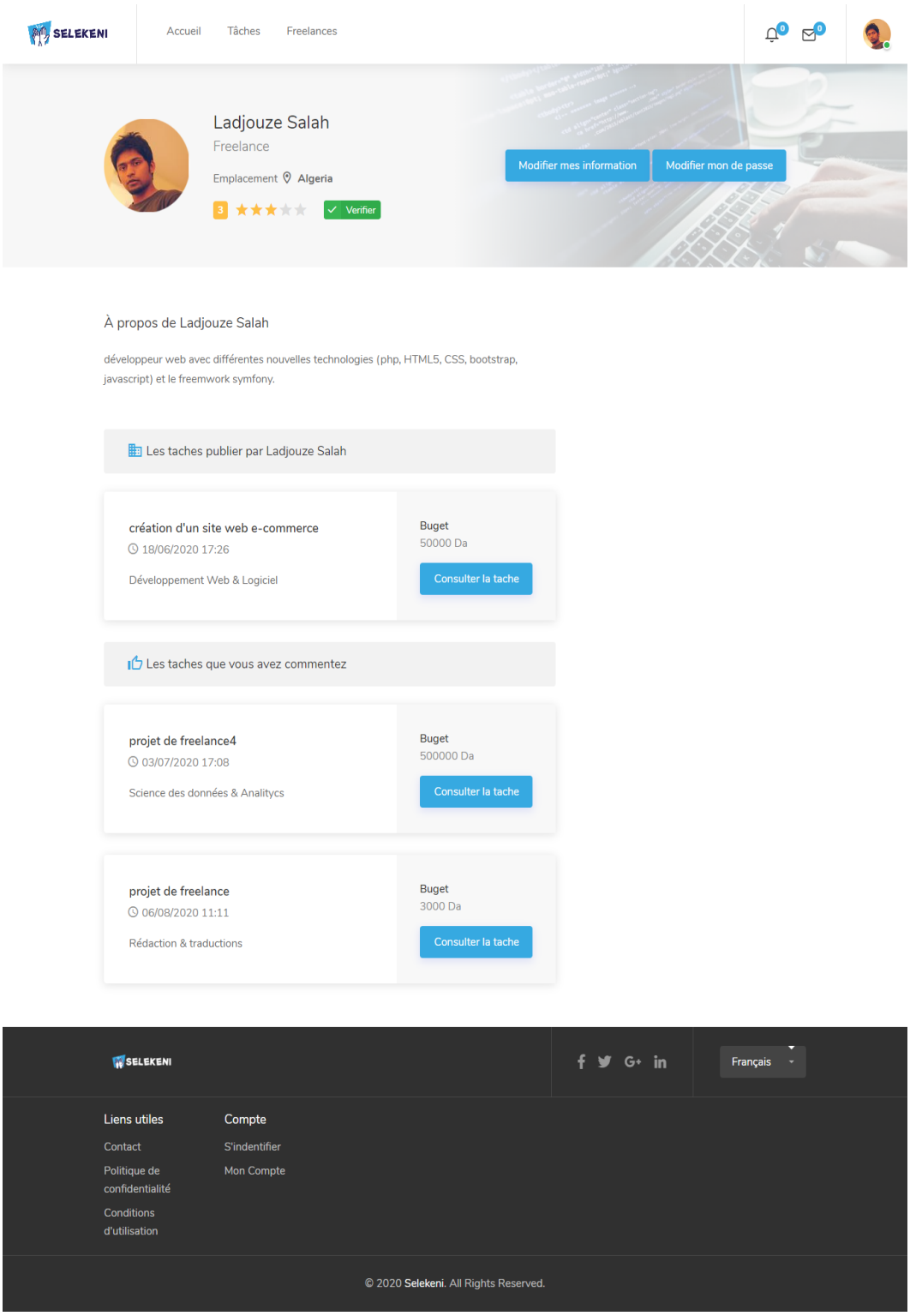

FIGURE 4.5 – Interface d'utilisateur

4.0.1.6. Interface du formulaire de paiement d'un abonnement et de la facturation

La figure ci-dessus représente l'interface qui donne la possibilité à un freelance de payer un abonnement et d'imprimer sa facture.

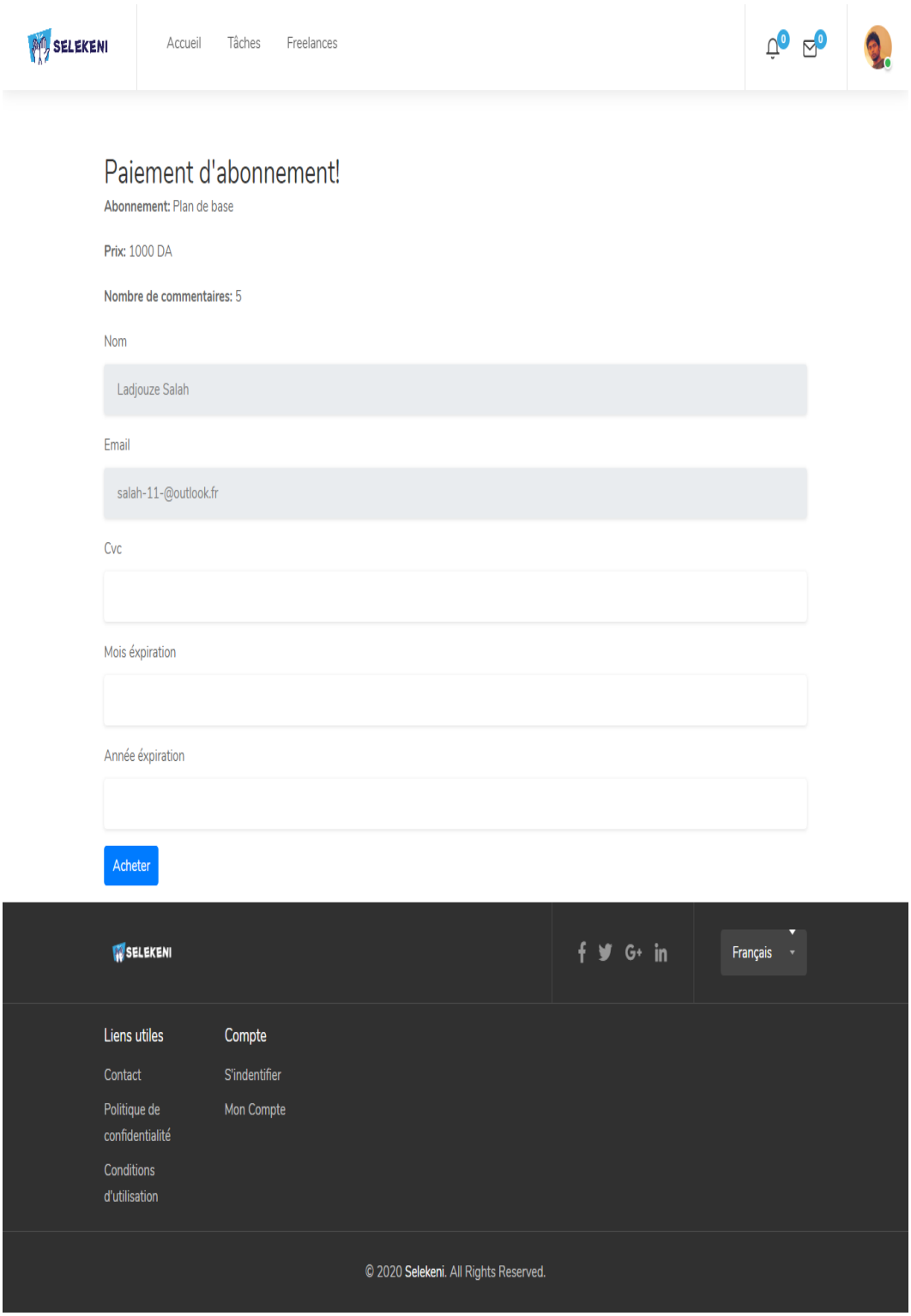

FIGURE 4.6 – Interface paiement d'abonnement

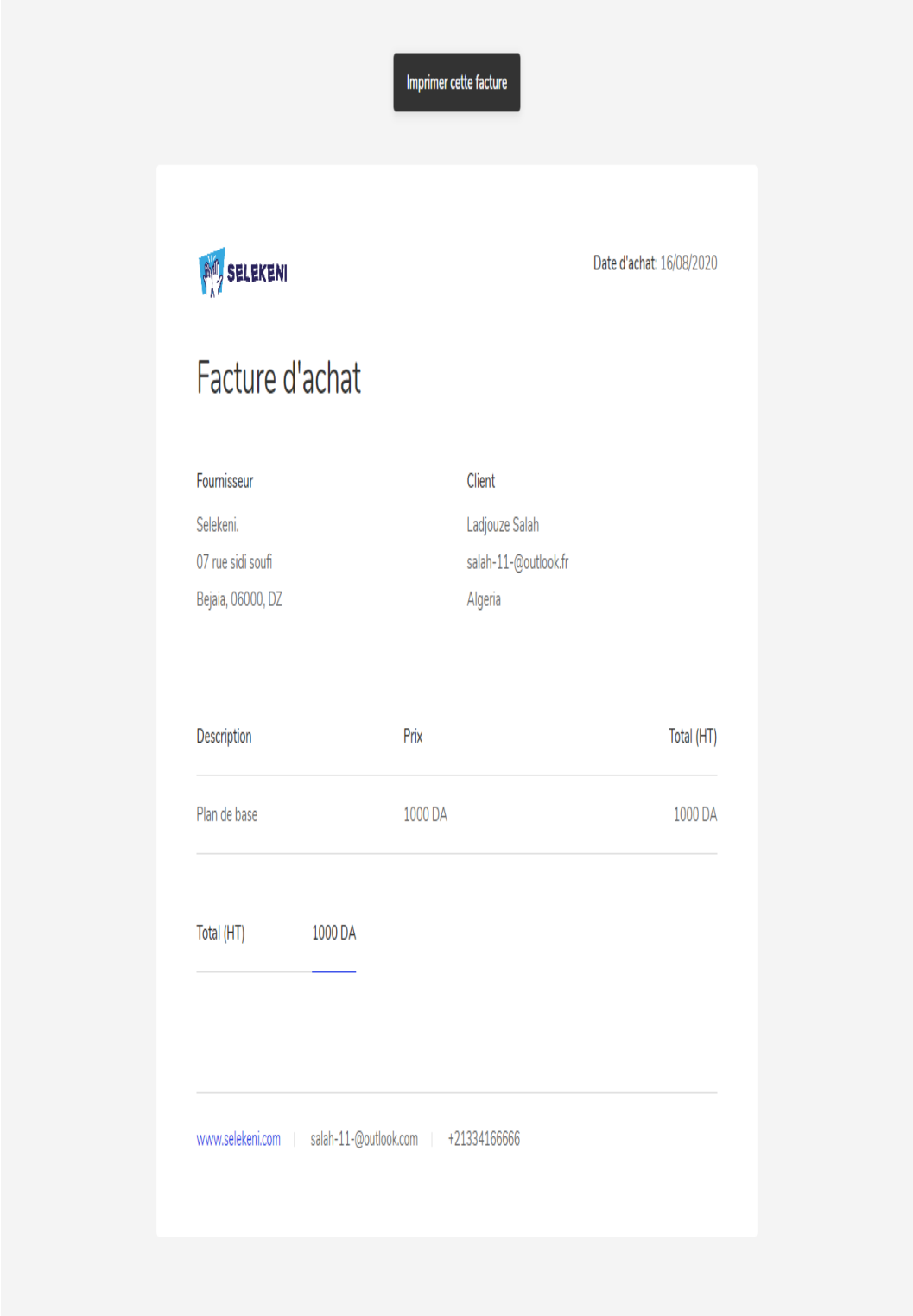

FIGURE 4.7 – Interface d'une facture

4.0.1.7. Interface du formulaire d'ajout d'une tâche

La figure ci-dessus représente l'interface qui donne la possibilité à nos utilisateurs d'ajout une tâche selon leurs besoins.

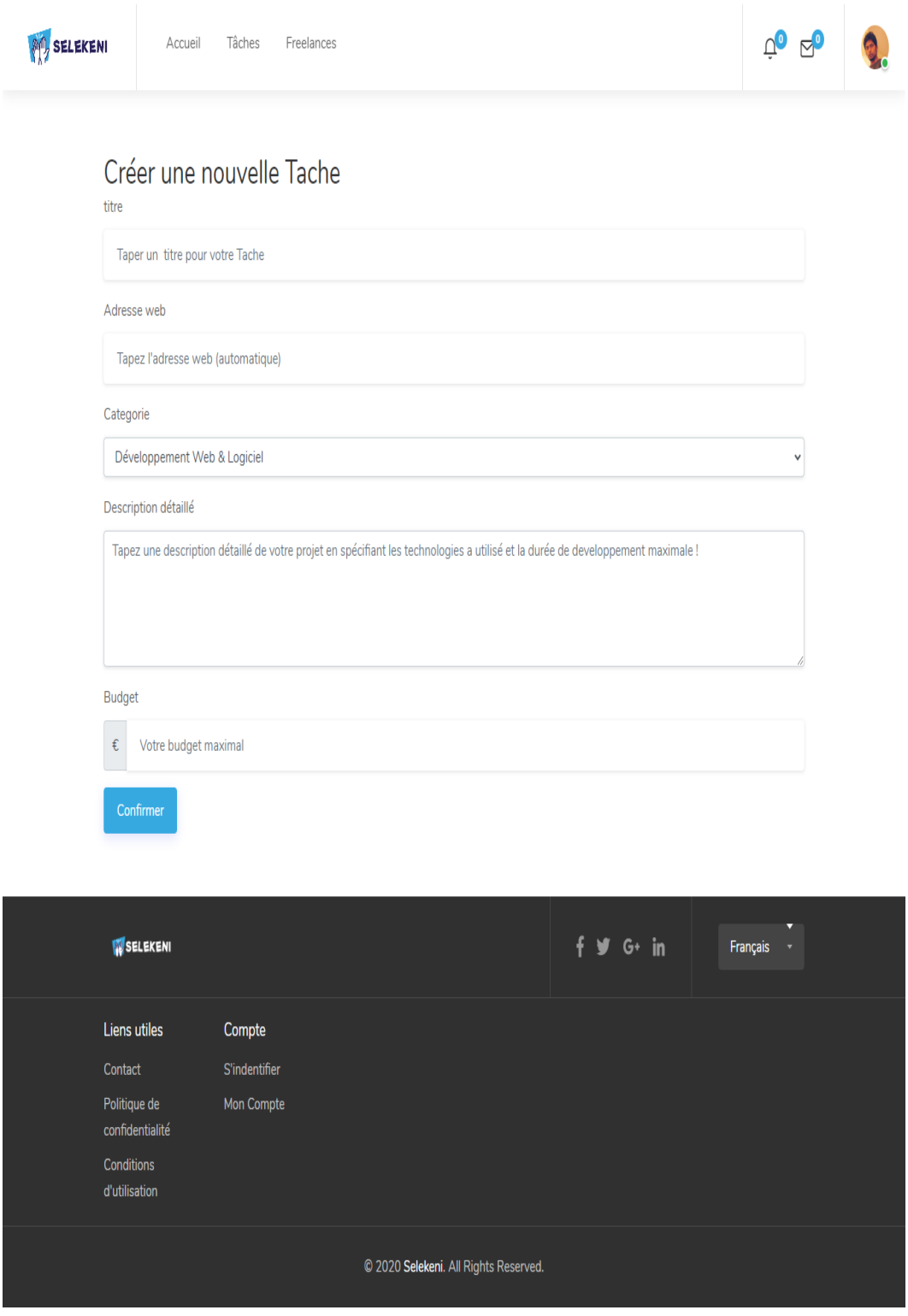

FIGURE 4.8 – Interface d'ajout d'une nouvelle tâche

4.0.1.8. Interface d'adminstration

La figure ci-dessus représente l'interface d'administration qui permet aux administrateurs de gérer les différentes utilisateurs et les tâches publies par ces derniers et leurs commentaires, et de consulter les statistiques de la platforme.

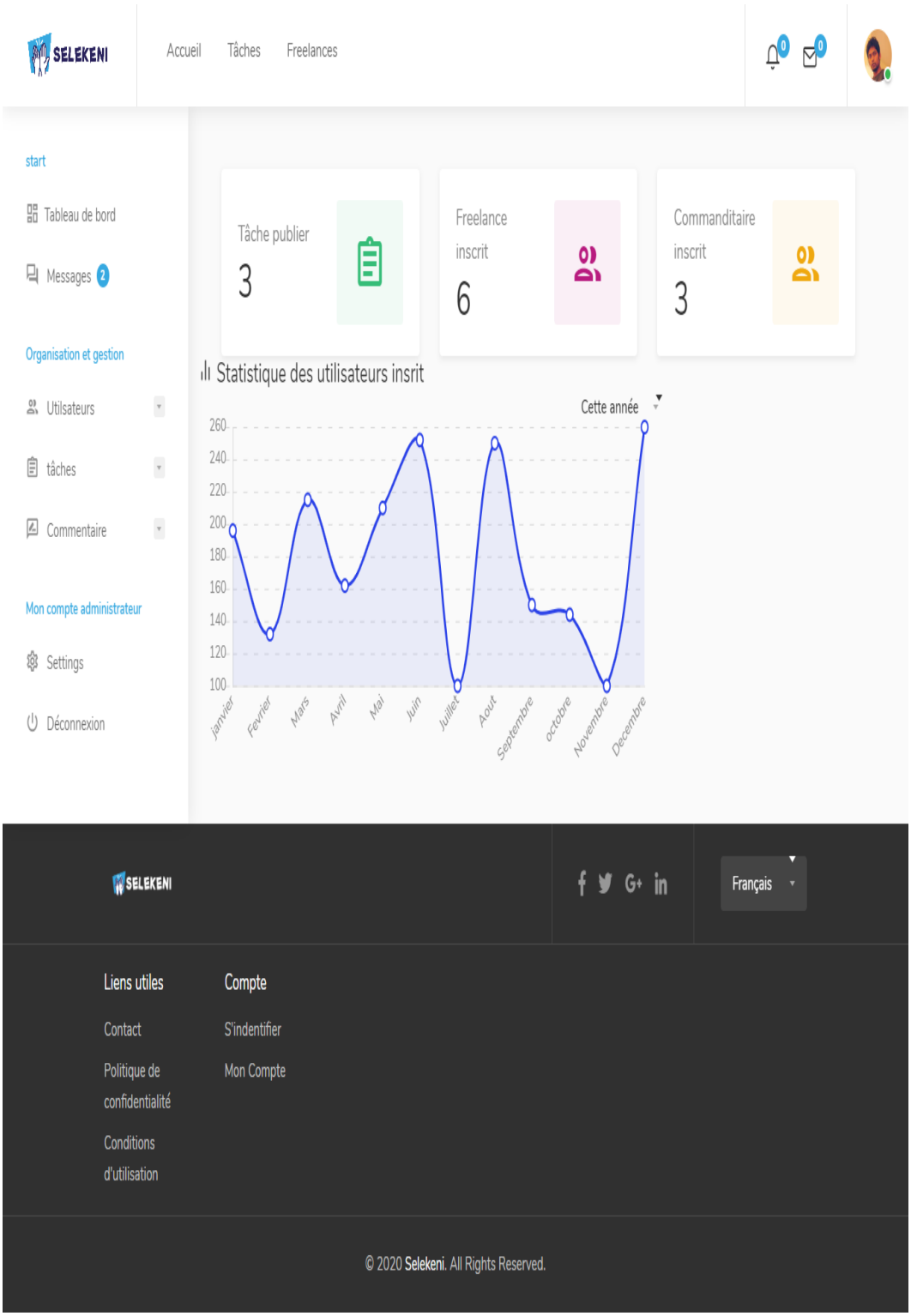

FIGURE 4.9 – Interface d'administrateurs

Conclusion

Dans ce dernier chapitre on a vu les différentes interfaces de notre plateforme Freelance SELEKNI, jugées les plus importantes suivie des tests de validation et d'implémentation notamment l'ajout d'une tâche, la facturation d'abonnement et de connexion.

Conclusion Générale et prespective

Au cours de ce travail, ce mémoire nous a vraiment met dans la situation exacte du travail à distance vu que la pandémie du COVIDE-19 nous a empêcher de ce reprocher entre binômes ou avec notre promoteur, durant toute la période du travail commençant de la théorie jusqu'à la pratique des notions de modélisations abordé dans les différentes diagrammes, la parie de la programmation.

Comme tout projet, qui nécessite une continuité et des améliorations, nous proposons pour de futurs travaux, les perspectives suivantes :

Rendre notre plateforme multilingue pour apporter plus d'accessibilité aux différents clients ; Réserver un nom de domaine pour la plateforme ; Héberger la plateforme pour qu'elle soit accessible sur un serveur en ligne ; Référencer la plateforme pour avoir plus de chance d'être aperçu sur les moteurs de recherche ; Ajouter le payement par carte bancaire en passant par les services intermédiaires tel que PayPal ou autres ; Pousser des données (la messagerie et les notifications) depuis le serveur vers des navigateurs Web en temps réel avec le protocole mercure. Enfin, nous comptons intégrer de nouvelles fonctionnalités dans notre application, comme l'implémentation d'un tableau de bord central qui fournit toutes les statistiques concernant les visites, les dépôts et les catégories les plus populaires ainsi qu'un espace de conversation en direct entre les membres connectés.

Bibliographie

[1] URL : https://www.Openclassroom.com. (visité le 15/04/2020).

[2] URL : https://www.journaldunet.fr (visité le 03/03/2020).

[3] URL : https://www.futura-sciences.com. (visité le 03/03/2020).

[4] URL : https://www.pierre-giraud.com/jquery-apprendre-cours/introduction-ajax/ (visité le 21/04/2020).

[5] URL : https://whatis.techtarget.com/fr/de?nition/Bootstrap (visité le 25/03/2020).

[6] URL : https://www.Symfony.com (visité le 15/04/2020).

[7] URL : https://www.pure-illusion.com. (visité le 01/03/2020).

[8] URL : https://www.doctrine-project.org/ (visité le 10/03/2020).

[9] URL : https://code.visualstudio.com. (visité le 10/03/2020).

[10] URL : http://www.wampserver.com. (visité le 10/03/2020).

[11] URL : https://www.apache.org. (visité le 10/03/2020).

[12] Pascal Roques. *UML 2: Modéliser une application web*. 2008, p. 246.

[13] Joseph Gabay et al. *UML 2 Analyse et conception - Mise en oeuvre guidée avec études de cas*. Dunod. Paris, 2008, p. 242.

[14] URL : https://www.alphalives.com (visité le 15/04/2020).

Liste des abréviations

$AJAX$	AsynchronousJavaScriptandXML.
API	Application Programming Interface.
$B2B$	business to business.
$CRUD$	Create, Read, Update, Delete.
$CSRF$	Cross Site request-forgery.
CSS	Cascading Style Sheets.
DQL	Doctrine et son langage.
GPS	Global Positioning System.
$HTML$	HyperText Markup Language.
$HTTP$	Hypertext Transfer Protocol.
MVC	Model View Controlor
ORM	Object Relational Mapper.
pop	Post Office Protocol.
SI	Système d'information.
$SMTP$	Simple Mail Transfer Protocol.
UP	Unified Process

Glossaire

A

— **AJAX :** Il désigne un type de conception de pages Web permettant l'actualisation de Certaines données d'une page sans procéder au rechargement total de cette page.

— **API :** Une API est une interface de programmation qui permet de se brancher sur une application pour échanger des données.

F

— **Framework :** Ensemble d'outils constituant les fondations d'un logiciel informatique ou d'applications web,et destiné autant à faciliter le travail qu'à augmenter la productivité du programmeurs qu'il l' utilisera.

G

— **Gradle :** Gradle est un outil d'automatisation de génération open-source conçu pour être Su-sammentexible pour créer pratiquement tout type de logiciel.

H

— **DQL :** Signifie Doctrine Query Language et est un dérivé du langage de requête d'objet qui est très similaire au Hibernate Query Language (HQL) ou au Java Persistence Query Language (JPQL)..

M

— **Métalangage :** C'est un langage de description d'un langage informatique.

Résumé

Le freelance est devenue un phénomène mondial au point où un nombre important de sociétés internationales engagent des compétences à travers le monde et n'hésitent pas à faire appel à des plateformes dans ce sens.

Le progrès et l'apport apportés dans ce sens ne cessent de révolutionner notre façon à accomplir les tâches et d'éliminer la nécessité de la présence quotidienne au bureau de travail dans l'entreprise. Le travail élaboré dans ce mémoire est l'implémentation d'une plateforme Freelance nommée SELEKENI pour réaliser et faciliter le travail à distance.

Mots clés : Plateforme, freelance, tâche, SELEKENI

Abstract

Freelance has become a global phenomenon to the point where a significant number of International companies engage skills across the world and do not hesitate to do Call for platforms in this direction.

The progress and contribution made in this direction continues to revolutionize the way we perform tasks and eliminate the need for the daily presence at the work desk in the company the work developed in this thesis is the implementation of a Freelance platform named SELEKENI to perform and facilitate remote work.

Keywords : Platform, freelance, task, SELEKENI