République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique
Niveau : Master 2

En vue de l'obtention du diplôme de master en informatique option Génie Logiciel

Thème

Conception et réalisation d'une application mobile e-commerce de vide-dressing

Encadrant :
Dr. AISSANI Sofiane - MCA

Auteurs :
ADJABI Namira
SAIDI Sarah

Membres du jury

Dr. ABBACHE Bournane	MCB Univ. Béjaia	Président
Dr. KACIMI Farid	MCB Univ. Béjaia	Examinateur

Septembre 2020

Dédicaces

A nos chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de nos études,

A nos chers frères et soeur pour leurs encouragements permanents et soutien moral,

A toutes nos familles pour leur appui tout au long de notre parcours universitaire,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour nous.

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui on voudrait témoigner toute notre gratitude.

On voudrait tout d'abord adresser toute notre reconnaissance à notre encadrant, Monsieur AISSANI Sofiane, pour sa patience, sa disponibilité et surtout ses judicieux conseils dans les moments de doutes, faisant témoins de son expérience et qui ont contribué à alimenter nos réflexions.

On désire aussi remercier les membres du jury Dr ABBACHE ET Dr KACIMI pour prendre la peine de juger notre mémoire, ainsi que les enseignants de l'université de Béjaïa qui nous ont fourni les outils nécessaires à la réussite de nos études universitaires à travers multiples modules.

On exprimera aussi notre reconnaissance envers les amis et collègues qui nous ont apporté leur soutien moral et intellectuel tout au long de notre démarche.
Enfin, on tient à témoigner beaucoup de gratitude aux personnes qui ont pris la peine de relire et nous aider à rédiger ce mémoire et à toutes leurs critiques constructives.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Dans un monde moderne où la culture de consommation s'est vite installée, il est devenu plus facile d'accumuler des biens dont on se lasse rapidement, ce notamment dans le domaine du textile. L'accumulation de vêtements qu'on ne porte plus mais qui prennent toujours de la place dans nos armoires a introduit l'idée du vide-dressing ; vendre ou acheter les vêtements qui ne servent plus à prix très raisonnable. L'idée d'en faire profit est une chose attrayante, surtout pour le jeune peuple algérien[1], en quête de rémunération et d'achats à petits prix.

Dans le présent mémoire, on explique la mise en œuvre d'une application mobile e-commerce qui aide les citoyens Algériens à vendre et acheter leurs habits, chaussures et bijoux de façon simplifiée et sans taxes sur une plateforme sécurisée.

Dans le premier chapitre, nous introduisons certaines notions utilisées dans le projet. Ce chapitre introduit les notions de User Interface et de User Experience, les heuristiques avec lesquelles elles sont mesurées et l'importance de chaque élément dans l'interface pour une utilisation fiable et fluide. Ce chapitre traite aussi le choix de la méthode de conception utilisée dans l'organisation du projet. Enfin, il présente quelques commandements de sécurité à respecter lors du développement de l'application mobile.

Le deuxième chapitre est un cahier des charges à la fois technique et fonctionnel, décrivant le projet, exposant la charte graphique de l'application, tous les choix liés aux interfaces, et le suivi de la méthode de conception choisie. Il relate le déroulement des tests et les plannings décrits dans les délais.

Le troisième chapitre est une description précise des fonctionnalités de l'application et la manière dont chaque macro répond aux requêtes. On y trouve les diagrammes de cas d'utilisation côté utilisateur, les diagrammes de séquence système pour l'interaction entre le système et l'utilisateur, les diagrammes d'interaction pour le lien entre les bases de données, les fonctions et les interfaces utilisateur et enfin un diagramme de classes pour obtenir un modèle relationnel nécessaire pour l'utilisation des bases de données relationnelles. Enfin, le dernier chapitre décrit les outils et méthodes de réalisation de l'application.

Chapitre 1

Notions liées au projet

1.1 Introduction

Nous introduisons, dans ce chapitre, les rudiments de notre projet. Nous parlons des processus de conception graphiques UI (User Interface) et UX (User eXperience) en les présentant, et en citant leurs principaux fondements. Nous introduisons ensuite les méthodes de conception SCRUM et UCD (ou développement piloté par l'utilisateur), puis l'intérêt de combiner les deux méthodes. Après cela, nous parlons de l'intégration de la sécurité dans les applications mobiles, et nous finissons par une conclusion.

1.2 Processus de conception graphique

Concevoir en tenant compte des besoins et des attentes des utilisateurs implique deux parties :

— • La conception UX (User eXperience), dans laquelle on s'assure que les utilisateurs trouvent de la valeur dans ce qui est fourni, en créant un chemin qui passe logiquement d'une étape à une autre.

— • La conception UI (User Interface), dans laquelle on assure que chaque partie communique visuellement ce chemin.

1.2.1 User EXperience

La conception UX consiste à résoudre les problèmes. Avoir un processus peut aider à contrôler ce qu'on fait. La différence entre une bonne application et une mauvaise application est la qualité de leur expérience utilisateur [9].

Usability honeycomb, de Peter Moreville 1.1, est devenu le fondement des meilleures pratiques pour les professionnels du User Experience ;cela permet d'aider à gérer les efforts à travers plusieurs points de contact avec l'utilisateur, notamment la manière dont ils ont découvert l'application, la séquence d'actions qu'ils prennent lorsqu'ils interagissent avec l'interface de l'application, les pensées qui surviennent au cours de l'utilisation, les impressions qu'ils retirent de l'interaction dans son ensemble.

FIGURE 1.1 – Fondement des pratiques UX

1.2.2 User Interface

La conception d'interface utilisateur (UI design) est la conception d'interfaces utilisateur pour machines et logiciels qui met l'accent sur la maximisation de la convivialité et de l'expérience utilisateur. Son but est de rendre l'interaction de l'utilisateur aussi simple et efficace que possible en termes de réalisation des objectifs de l'utilisateur (conception centrée sur l'utilisateur).

Principes de la conception UI

Les principes de conception rendent le travail du concepteur d'interface utilisateur beaucoup plus facile. Ils suppriment une grande partie de la conjecture et rendent les interfaces plus prévisibles et, par conséquent, plus faciles à utiliser.
Les heuristiques de Jakob Nielsen, reflétées par Apple, Google et Adobe, via les directives d'interface utilisateur publiées et partagées par ces dernières, ont établi en 1990 une liste de dix directives de conception d'interface utilisateur qui sont, à ce jour, encore d'actualité [8] :

1. **Visibilité de l'état du système :** les utilisateurs sont toujours informés des opérations du système avec un statut facile à comprendre et très visible à l'écran pendant un temps raisonnable.
2. **Correspondance entre le système et le monde réel :** refléter le langage et les concepts que les utilisateurs cibles trouveraient dans le monde réel.
3. **Contrôle et liberté des utilisateurs :** des pas en arrière devraient être possibles (actions d'annulation ou de rétablissement)
4. **Cohérence et normes :** les éléments graphiques et la terminologie doivent être conservés sur toutes les plates-formes similaires (exemple : une même icône doit toujours représenter la même chose)
5. **Prévention des erreurs :** les utilisateurs n'aiment pas être appelés sur les erreurs et résoudre les problèmes qui peuvent être au-delà de leurs aptitudes. L'élimination

ou le signalement des sections pouvant entraîner des erreurs peut permettre de les prévenir.

6. **Reconnaissance au lieu de rappel :** en raison des limites de la mémoire à court terme, il faut s'assurer que les utilisateurs peuvent utiliser la reconnaissance au lieu d'avoir à se rappeler d'informations. Il faut donc minimiser la charge cognitive (effort associé à un sujet spécifique) en conservant des informations pertinentes à la tâche dans l'interface affichée.
7. **Flexibilité et efficacité d'utilisation :** avec une utilisation accrue, avoir moins d'interaction permet d'avoir une navigation plus rapide. Cela peut être réalisé en utilisant des abréviations, des touches de fonction, des commandes cachées et des fonctions de macro.
8. **Conception esthétique et minimaliste :** toutes les informations inutiles risque de limiter l'attention de l'utilisateur, ce qui pourrait empêcher la mémoire de l'utilisateur de récupérer les informations pertinentes. Par conséquent, l'affichage doit être réduit aux seuls composants nécessaires aux tâches en cours.
9. **Aider les utilisateurs à reconnaître, diagnostiquer et récupérer des erreurs :** les concepteurs doivent supposer que les utilisateurs ne sont pas en mesure de comprendre la terminologie technique, par conséquent, les messages d'erreur doivent, dans la mesure du possible, toujours être exprimés en langage clair pour garantir que rien ne se perd dans la traduction.
10. **Aide et documentation :** Idéalement, les utilisateurs doivent pouvoir naviguer dans le système sans avoir à recourir à la documentation. Cependant, selon le type de solution, une documentation peut être nécessaire.

Google Inc. produit certainement des designs qui reflètent les heuristiques ci-dessus. Jon Wiley, le concepteur en chef de la recherche Google en 2012, a déclaré un jour :
"Quand je pense à la conception et à la création de superbes UI, je pense généralement à cela en termes de trois choses : l'utilisabilité, l'utilité et l'opportunité."
Les 10 directives d'interface utilisateur de Nielsen et Molich couvrent très bien ces trois éléments clés de l'expérience utilisateur.

Phases de la conception UI :

La conception de l'interface utilisateur comporte plusieurs phases et processus, dont certains sont plus importants que d'autres, selon le projet [10].

1. **Collecte des exigences de fonctionnalité** - assemblage d'une liste des fonctionnalités requises par le système pour atteindre les objectifs du projet et les besoins potentiels des utilisateurs.
2. **Analyse des utilisateurs et des tâches** - analyse des utilisateurs potentiels en étudiant comment ils effectuent les tâches que la conception doit prendre en charge.
3. **Développement de l'architecture de l'information du processus et du flux d'information**

4. **Prototypage de Wireframes**
5. **Inspection de l'utilisabilité via une alternance entre l'évaluation basée sur les heuristiques et les tests utilisateurs**
6. **Tests d'utilisabilité** - l'utilisateur parle de ce qu'il pense de l'application durant le test.
7. **GUI**- conception réelle de l'interface graphique finale.
8. **Maintenance logicielle**- après le déploiement d'une nouvelle interface, une maintenance occasionnelle peut être nécessaire pour corriger, modifier ou mettre à niveau le système en répétant les étapes du cycle de vie de l'interface.

1.3 Fondements des Processus UX/UI

1.3.1 OnBoarding

L'OnBoarding, dans les applications mobiles, est le processus qui permet aux nouveaux utilisateurs de comprendre et d'interagir suffisamment avec une application pour continuer à l'utiliser, au lieu de l'abandonner après la première utilisation. On satisfait ainsi l'heuristique 10 de Nielsen et Molich liée à la documentation.

Pourquoi l'intégration des applications est-elle importante ? Étant donné que plus de 20% des utilisateurs qui téléchargent une application ne l'utilisent qu'une seule fois, il est important de concevoir un processus d'intégration engageant qui les encourage à revenir.

Il existe 6 types d'approches d'OnBoarding [7] :

1. **L'approche des avantages** particulièrement utile pour les applications qui exigent que les utilisateurs s'inscrivent avant de commencer à l'utiliser.
2. **L'approche des fonctionnalités** particulièrement utile pour les applications qui ont des fonctionnalités complexes.
3. **L'approche orientée action** incite les gens à utiliser l'application lors de l'Onboarding.
4. **L'approche de configuration de compte** explique aux utilisateurs pourquoi l'application a besoin de cetaines autorisation d'accès. Cela rassure l'utilisateur quant aux problèmes de sécurité et renforce la confiance sans être intrusif.
5. **L'approche interactive** en utilisant un quizz, par exemple.
6. **L'approche combinée** la plupart des applications utilisent une combinaison d'approches d'Onboarding multiples pour guider les nouveaux utilisateurs.

Il est conseillé de donner l'option de passer l'Onboarding quand c'est possible, d'afficher les étapes de la barre de progression avec un maximum de 5 étapes.

1.3.2 Typographie

La typographie est la composante visuelle du mot écrit. La typographie englobe toutes les considérations sur l'apparence visuelle du mot ; elle englobe le procédé d'impression, les corps, la présentation, en déterminant la dimension du texte, des illustrations (et leur situation dans le texte).

i. Différents termes liés à la typographie

— **Famille de cacartères** - la conception de tous les caractères composés d'une famille de caractères et de toutes les polices englobantes. Helvetica et Myriad Pro, etc. sont des familles de caractères.

— **Police de caractères** - le poids ou l'instance spécifique d'une famille particulière (gras, italique..) Helvetica Bold Italic est une Police de caractères où Helvetica en est la famille.

— **Police** - Fait référence au fichier du logiciel informatique qui contient des informations concernant l'affichage et la sortie d'une police de caractères. Les polices n'existent que sur un ordinateur.

ii. La Police

Avec la consommation numérique d'aujourd'hui, concevoir une bonne interface utilisateur signifie offrir du contenu dans un format utilisable qui encourage la lecture et l'engagement.

Le rôle d'une police peut sembler se limiter à informer et faciliter la lisibilité. Cependant, il y est pour beaucoup plus : la personnalité, la tonalité, les expressions, l'accentuation, l'engagement sont certaines des exigences psychologiques plus lourdes d'une police aujourd'hui.

iii. Quelques principes de typographie

Etant une pratique quelques peu compliquée, et afin de faciliter son implémentation, la typographie dans les applications se soumet à certaines pratiques communes chez les expérimentés, notamment :

a **Choisir la bonne Famille de caractères, selon son utilisation.** Par exemple :

- **Open Sans :** idéal pour les sites web et applications mobiles pour une utilisation prolongée et la lisibilité. Open Sans est agréable pour les yeux et peut être bon pour une consommation régulière. Meilleur jumelage avec : Montserrat, Brandon Grotesque, Playfair Display.
- **Playfair Display :** idéal pour les sites web et applications mobiles stylisés, élégants et sophistiqués. Cette police stylistique est mieux utilisée dans les blogs d'agence, de portefeuilles, de voyages et de mode. Idéal pour les titres et les citations. Meilleur jumelage avec : Open Sans, Lato, Roboto, Georgia, Museo Sans et Proxima Nov.
- **Roboto :** comme il fait partie de la famille Google et sert de police Android par défaut depuis 2012, il fonctionne bien comme police d'application par défaut, y compris l'application de chat, le commerce électronique et plus encore.

b **Choisir les bonnes mesures :** les paragraphes doivent prendre 50-75 caractères par ligne, les titres 15 caractères par ligne.

c **Définir l'espacement entre les lettres idéalement à -1%.**

d **Définir la hauteur de la ligne comme suit :** taille de police * 1,618 (Golden Ratio).

e **Choisir les bons rapports de taille titre à paragraphe :** éventuellement, 3 :1 pour les titres, 4,5 :1 pour les autres éléments.

1.3.3 Couleurs et contraste

La théorie des couleurs est une science à part entière et elle joue un rôle extrêmement important dans l'Interface Utilisateur. Pour créer une harmonie dans les couleurs de l'interface, il faut tout d'abord établir une palette d'environs 6 couleurs à laquelle on se fie tout au long de la conception de l'interface graphique post-maquettes. A chaque couleur sa signification :

- Les couleurs liées à la marque définissent l'application et sont essentiellement utilisées pour les boutons.

- Les couleurs fonctionnelles définissent l'architecture de l'interface (tableaux de navigation, dialogues, liens, texte...)

- Les couleurs liées au Feedback sont utilisées pour indiquer l'état d'action (avertissement, succès, information, danger) ont toutes des couleurs primaires et secondaires.

- Les couleurs de contraste sont utilisées pour mettre l'accent sur les informations importantes. Ces couleurs doivent être plus vives et plus saturées que les autres

couleurs de l'application.

— Les couleurs neutres, simples ; une couleur claire et une couleur foncée, pour jouer avec l'opacité. Utilisées commes parenthèses fermantes et ouvrante de la palette de couleur de l'application.

— La couleur des interactions est la couleur du survol qui résulte de l'utilisation de la couleur de marque mélangée à une couleur neutre (comme du gris ou du noir.)

Une bonne interface utilisateur prend aussi en considération les personnes présentant des cas de daltonisme. Il existe 8 types de daltonisme, n'apercevant qu'une certaine rangée de couleurs, ce qui accentue l'importance des textes explicatifs dans les formulaires et les erreurs. Il existe un Plugin pour les logiciels de conception de GUI (notamment SketchUp, Figma et AdobeXD), appelé Stark 1.2[3], qui donne un aperçu des différentes façons dont chaque type de daltonisme présente l'interface.

FIGURE 1.2 – Logo du plugin Stark

Le contraste, quant à lui, intervient quand deux éléments ont des propriétés différentes que l'œil humain traite comme contraste, créant ainsi le concept de hiérarchie visuelle. Ainsi, c'est au concepteur de contrôler l'attention visuelles de l'utilisateur. Il peut être lié à la couleur (avec des rapports de luminosité des couleurs et les tons froids et chauds), à la taille des objets graphiques, à la typographie, à la forme des objets étant commune ou pas et à la structure et alignement des objets.

1.3.4 Les Boutons

Les boutons doivent prendre la forme qu'ils veulent communiquer (par exemple, les boutons ronds sont conviviaux), la couleur de la marque (la couleur principale de la palette) ainsi que leur localisation standard dans l'interface. Un bouton doit avoir 4 aspects : son aspect normal, son aspect en cas de survol, un aspect de traitement (d'une requête) et un aspect d'accomplissement correct de la requête. Le nom attribué au bouton doit être standard et clair. Le bouton doit être inactif jusqu'à ce qu'il soit prêt à être cliqué, et enfin, un bouton principal et secondaire doivent différer dans la couleur.

1.4 Méthode de conception

1.4.1 Méthodes de conception Agile

Agile [11] est un processus par lequel une équipe peut gérer un projet en le divisant en plusieurs étapes et impliquant une collaboration constante et une amélioration et itération continues à chaque étape afin de répondre plus pleinement aux besoins d'un client. Tout cela commence par le client décrivant comment le produit final sera utilisé et quel problème il résoudra. Agile a quatre valeurs principales :

a Les individus et interactions sont plus importants que les processus et les outils.

b Le logiciel de travail est plus important qu'une documentation complète.

c La collaboration du client est plus importante que la négociation du contrat.

d Répondre au changement est plus important que de suivre un plan.

Méthode SCRUM

Il s'agit d'un cadre de gestion de projet agile qui peut être utilisé pour gérer des projets iteratifs et incrémentiels de tous types [12]. Il est populaire car il est simple, a prouvé sa productivité et est capable d'intégrer diverses pratiques globales.

Le **Product Owner** travaille en étroite collaboration avec son équipe pour identifier et hiérarchiser les fonctionnalités du système en créant un Product Backlog. Une fois les priorités établies, les équipes inter-fonctionnelles estimeront et s'engageront à fournir des « incréments potentiellement expédiables» de logiciels pendant des sprints. Un sprint ne peut être modifié que par l'équipe qui y travaille.

Le **Product Backlog** [13] est sous la responsabilité unique du Product Owner, mais doit également être accessible par l'équipe. Le guide Scrum préconise un maximum de 150 **items** dans un Product Backlog.

1. Les Product Backlogs inclut en plus des fonctionnalités, les corrections de bugs, les exigences non fonctionnelles, etc. Il diffère en fonction des besoins spécifiques. Dans notre cas :
   (a) La correction de bugs se fait durant le facteur de débogage, c'est-à-dire le temps alloué par sprint pour le débogage. Il est mieux de prévenir les bugs à travers certaines techniques comme les tests de régression automatisés[4] (tests pour confirmer qu'un programme récent ou un changement de code n'a pas affecté les fonctionnalités existantes), l'intégration continue (la fusion de toutes les copies de travail des développeurs sur une ligne principale partagée plusieurs fois par jour), etc.

(b) Exigence non-fonctionnelle : le plan de mise-en-œuvre des exigences non-fonctionnelles est détaillé dans l'architecture du système, car il s'agit généralement d'exigences architecturales importantes. D'une manière générale, les exigences fonctionnelles définissent ce qu'un système est censé faire et les exigences non-fonctionnelles définissent comment un système est censé être.

2. Les items (users stories) : chaque item est une exigence du client rédigé selon une terminologie métier et non technique. Il appartient au Product Owner de s'assurer que l'item pourra être compris par l'équipe. Après s'être assuré qu'il n'y a pas de doublons ou d'items qui se chevauchent, le Product Owner pourra attribuer un ID à chaque item et les disposer dans un tableau avant de procéder à un premier classement pour attribuer une valeur d'importance à chaque item. Cette valeur d'importance n'est pas figée et elle évoluera au fur et à mesure des feedbacks utilisateurs. Il est important de noter que seul le Product Owner est en droit d'attribuer ou modifier cette valeur.
3. Estimation initiale : elle représente, théoriquement, la quantité de travail nécessaire au développement de chaque item. C'est l'équipe qui la détermine, elle prend en compte le temps et la complexité de développement. Elle est souvent faite grâce à la méthode du Planning Poker[17]. C'est une technique ludique qui propose à chaque membre de l'équipe d'attribuer une valeur comprise dans la suite de Fibonacci sans avoir connaissance de la réponse des autres membres. Les réponses sont ensuite dévoilées en même temps afin d'encourager les discussions et échanges d'opinion sur la complexité de l'item en question. Cela permet finalement d'aboutir à l'estimation la plus juste possible.
4. Démonstration et notes : la démonstration est une démarche à suivre pour présenter l'item lors de la réunion de fin d'itération. Les notes permettent d'indiquer la dépendance et autres informations pouvant influer sur l'importance et le développement de l'item.

1.4.2 User Centered Design ou le développement piloté par l'utilisateur

UCD [14] est un cadre de processus, il peut être caractérisé comme un processus de résolution de problèmes en plusieurs étapes qui oblige non seulement les concepteurs à analyser et à envisager la façon dont les utilisateurs finaux sont susceptibles de consommer un produit, mais également de valider leurs hypothèses concernant le comportement des utilisateurs dans les tests du monde réel. Ces tests sont effectués avec/sans utilisateurs réels à chaque étape du processus.
Voici les principes qui garantiront qu'une conception est centrée sur l'utilisateur :

- La conception est basée sur une compréhension explicite des utilisateurs, des tâches et des environnements.
- Les utilisateurs finaux sont impliqués tout au long de la conception et du développement.

- La conception est guidée et affinée par une évaluation centrée sur l'utilisateur.
- Le processus est itératif.
- La conception répond à l'ensemble de l'expérience utilisateur.
- L'équipe de conception comprend des compétences et des perspectives multidisciplinaires.

Le but de la conception centrée sur l'utilisateur est de créer des produits très utilisables. Voici les phases générales du processus de conception centrée sur l'utilisateur :

1. **Spécifier le contexte d'utilisation :** identifier les principaux utilisateurs du produit, pourquoi ils utiliseront le produit, quelles sont leurs exigences et dans quel environnement ils vont l'utiliser.
2. **Spécifier les exigences :** une fois le contexte spécifié, il est temps d'identifier les exigences granulaires du produit. Il s'agit d'un processus important qui peut faciliter davantage la conception.
3. **Créer des solutions de conception et de développement :** en fonction des objectifs et des exigences du produit, lancer un processus itératif de conception et de développement de produit.
4. **Évaluer le produit :** les concepteurs de produits effectuent des tests d'utilisabilité pour obtenir les commentaires des utilisateurs sur le produit.

1.4.3 UCD et SCRUM

Avec le développement Agile [15], la principale mesure de progrès est liée au logiciel fonctionnel. Ce qui n'est plus le cas une fois qu'on adopte une philosophie centrée sur l'utilisateur. Contrairement à la conception Agile, UCD n'est pas axé sur le client - il est centré sur l'utilisateur final.

La recherche UCD fournit un mécanisme par lequel les décisions de conception peuvent être validées et testées, ce qui fait que la conception devient moins sujet à débat. Plus important encore, en gardant les utilisateurs finaux d'un produit au cœur de son processus de conception et de développement, le résultat final est beaucoup plus susceptible d'être utile, utilisable et significatif.

Le développement de logiciels Agiles a considérablement amélioré la fiabilité lors de la livraison de logiciels, en augmentant le retour sur investissement et en réduisant le risque lors de la création de logiciels. Cependant, dans un monde d'iPhones et d'applications Google, cela peut ne plus être suffisant.

Les utilisateurs commencent à attendre davantage des logiciels. Ils s'attendent à ce qu'il fonctionne intuitivement et sans tracas, comme s'il avait été conçu spécialement pour eux. Plus souvent qu'avant, les utilisateurs savent ce qu'ils veulent réaliser. Tout logiciel qui les empêche d'atteindre efficacement leurs objectifs sera rapidement remplacé.

Cela est particulièrement vrai avec les applications Internet où une alternative fiable n'est qu'à une recherche sur le Web.

Il ne fait aucun doute que pour continuer à offrir de la valeur aux clients, il faut continuer à appliquer les principes de la philosophie de développement agile. Mais, pour que les nouveaux logiciels réussissent vraiment, on s'efforce d'adopter une approche plus centrée sur l'utilisateur.

Il existe une histoire de conflits entre les développeurs agiles et les concepteurs UX. Pour mieux comprendre la déconnexion, comparons certains des principes du Manifeste Agile qui provoquent un conflit avec la philosophie UCD :
Le Manifeste Agile dit : notre priorité absolue est de satisfaire le client par la livraison précoce et continue de logiciels. Un logiciel fonctionnel est la principale mesure du progrès.
Le principe UCD équivalent pourrait se lire : notre priorité absolue est d'aider à créer une expérience pour les utilisateurs finaux où ils peuvent atteindre leurs objectifs facilement et efficacement avec une perturbation minimale. La satisfaction des besoins des utilisateurs finaux, équilibrée avec la réalisation des objectifs commerciaux, est la principale mesure du succès.

Problèmes courants lors de l'intégration d'UCD et d'Agile

- **Conception sans contraintes :** lorsque la conception d'un système est créée avec des utilisateurs mais sans rétroaction régulière d'une équipe de développement, il existe un risque important qu'une conception soit proposée et approuvée lorsque personne ne sait combien de temps cela prendra ou si cela coûtera cher à mettre en œuvre. Lorsque le devis arrive, il est bien supérieur au montant attendu.
- **Valider avec une utilisation réelle :** pour pouvoir effectuer et répondre à un niveau de validation réel, il est nécessaire d'avoir l'équipe de conception intégrée à l'équipe de développement pour exécuter les tests et modifier sa conception en fonction des résultats.
- **Pas le temps d'itérer :** lors d'une approche centrée sur l'utilisateur pour le développement Agile, il est essentiel que nous prenions le temps d'itérer afin de pouvoir répondre aux commentaires des utilisateurs sans pousser les fonctionnalités promises aux clients. Il existe quelques stratégies pour gérer cela, par exemple en prenant un pourcentage de l'estimation initiale et en l'ajoutant au calendrier.

1.5 Sécurité des applications mobiles

La sécurité des applications n'est ni une fonctionnalité ni un avantage - c'est une nécessité absolue. La sécurité devrait être une priorité à partir du moment où l'on commence à écrire la première ligne de code.[16]

Avec plusieurs types d'informations en jeu, une application mobile doit être conçue, à tout prix, pour protéger ses utilisateurs et clients. La sécurité doit être assurée et intégrée à chaque étape et élément de l'application via 10 façons :

Sécuriser le code

Les bugs et les vulnérabilités d'un code sont le point de départ de la plupart des attaquants pour pénétrer une application, tout ce dont ils ont besoin est une copie publique de l'application. Des recherches montrent que les codes malveillants affectent à tout moment plus de 11,6 millions d'appareils mobiles. [16]

Pour remédier au mieux à cela, il faut minifier le code afin de ne pas être victime d'ingénierie inverse. Il est aussi nécessaire d'exécuter plusieurs tests suivis de correction des bugs au fur et à mesure. Enfin, Il faut renforcer le code en utilisant une signature du code.

Chiffrer toutes les données

Chaque unité de données échangée sur l'application doit être chiffrée. Le cryptage est le moyen de brouiller le texte brut jusqu'à ce qu'il ne soit qu'un bloc de texte sans signification pour personne, sauf pour ceux qui ont la clé. Cela signifie que même si des données sont volées, il n'y a rien que les criminels puissent lire et utiliser à mauvais escient.

Être prudents avec les bibliothèques

Lors de l'utilisation de bibliothèques tierces, il faut soigneusement tester le code avant de l'utiliser dans l'application. Aussi utiles qu'elles soient, certaines bibliothèques peuvent être extrêmement peu sûres pour l'application. La bibliothèque GNU C, par exemple, avait une faille de sécurité qui permettait aux attaquants d'exécuter à distance du code malveillant et de planter un système. Et cette vulnérabilité est restée inconnue pendant plus de sept ans.

Utilisation d'APIs autorisées

Les APIs qui ne sont pas autorisées peuvent accorder involontairement à un pirate des privilèges malveillants.

Utilisation d'une authentification de haut niveau

Certaines des plus grandes violations de sécurité se produisent en raison d'une authentification faible. Une grande partie de l'authentification dépend des utilisateurs finaux de l'application, mais les développeurs peuvent encourager les utilisateurs à être plus sensibles à l'authentification, et ce en concevant une authentification n'acceptant que des mots de passe alphanumériques forts qui doivent être renouvelés tous les trois ou six mois. L'authentification multi-facteur gagne aussi en importance, utilisant ainsi plusieurs facteurs d'authentification.

Déployer des technologies de détection de sabotage

Il existe des techniques pour déclencher des alertes lorsque quelqu'un essaie de falsifier le code ou d'injecter du code malveillant. Une détection de sabotage active peut être déployée pour s'assurer que le code ne fonctionnera pas du tout s'il est modifié.

Utiliser le principe du moindre privilège

Le principe du moindre privilège dicte qu'un code doit s'exécuter avec uniquement les autorisations dont il a absolument besoin et pas plus.

Déployer une gestion de session appropriée

Les « sessions » sur mobile durent beaucoup plus longtemps que sur ordinateur. Cela rend la gestion des sessions plus difficile pour le serveur. Il est préférable d'utiliser des jetons au lieu des identifiants d'appareil pour identifier une session. Les jetons peuvent être révoqués à tout moment, ce qui les rend plus sûrs en cas de perte ou de vol d'appareils.

Utiliser les meilleurs outils et techniques de cryptographie

Il ne faut pas coder les clés de chiffrement en hard code car cela permet aux attaquants de les voler facilement. Il est préférable de stocker les clés dans des conteneurs sécurisés et jamais localement sur l'appareil.

Tester à plusieurs reprises

La sécurisation est un processus qui ne s'arrête jamais. De nouvelles menaces émergent et de nouvelles solutions sont nécessaires. Il faut investir dans les tests de

pénétration, la modélisation des menaces et les émulateurs pour tester en continu les vulnérabilités et ainsi les corriger avec chaque mise à jour.

1.6 Conclusion

Ce chapitre représente une étude de la conception et la création d'application dans le monde du travail moderne et réel et jouera un rôle important quant aux décisions prises lors de l'organisation et la mise en oeuvre du projet, nous permettant ainsi d'entamer le cahier des charges de ce dernier.

Chapitre 2

Cahier des charges

2.1 Introduction

Le cahier des charges est la base de la conception d'un projet ; c'est un document qui permet de comprendre et d'expliquer un projet dans son ensemble, avec toutes les contraintes, les besoins, les objectifs ou encore les intervenants qui y sont liés. Nous allons, dans ce chapitre, présenter notre projet ainsi que son contexte d'utilisation, faire une analyse concurentielle et énoncer les besoins de l'utilisateur, présenter par la suite un organigramme des interfaces de l'application puis sa charte graphique, parler des méthodes de conception et de sa planification, et finir par une conclusion.

2.2 Présentation du projet

Ce projet sert à organiser la conception et la création d'une application mobile e-commerce moderne avec des outils et techniques d'actualité dans le monde du travail international. Nous avons décidé d'entreprendre la conception et la création d'un vide-dressing en ligne pour tous les utilisateurs Algériens en s'adaptant au manque d'utilisation des moyens de paiement en ligne en Algérie.

Aussi, la notion de vide-dressing en ligne est une notion peu utilisée dans le pays mais qui pourrait être particulièrement rentable aux deux fins d'une transaction.

2.3 Contexte d'utilisation

L'une des étapes de la méthode centrée utilisateur est de définir le contexte exact de l'utilisation et de définir les utilisateurs finaux de l'application (User Persona)

Notre projet vise toute catégorie de personnes ayant une connaissance minimum dans l'utilisation de Smartphone. Elle peut attirer particulièrement les personnes adolescentes et adultes voulant se débarrasser de leurs anciens vêtements, chaussures ou bijoux de manière rentable.

L'application sera utilisable sous le système d'exploitation Android car c'est l'OS le plus utilisé en Algérie, et même dans le monde.[2]

2.4 Analyse Concurrentielle

Avant de se livrer à la planification du déroulement de la conception et la création de l'application, il est intéressant d'étudier le marcher et d'analyser quelques

applications qui ont un but similaire au notre et ainsi inspirer notre produit final.

Nous nous sommes attardées sur deux applications différentes :

Dabchy : c'est à la fois une application de vide-dressing pour vendre et acheter des articles de mode neufs ou d'occasion mais aussi un réseau social où les utilisateurs peuvent interagir (aimer, commenter et se suivre) 2.1.

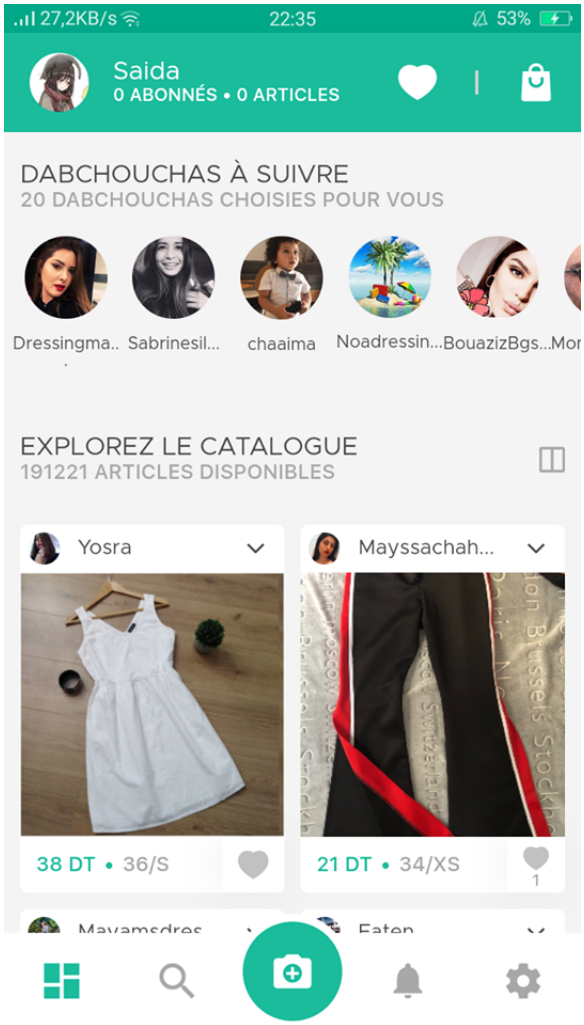


FIGURE 2.1 – Interface principale de Dabchy

Vinted : originaire de Lituanie, Vinted était à la base dédiée uniquement à la vente de vêtements pour femmes, puis s'est élargie et est devenue un marché en ligne communautaire qui permet à ses utilisateurs de vendre, d'acheter, et d'échanger des vêtements et accessoires d'occasion 2.2.

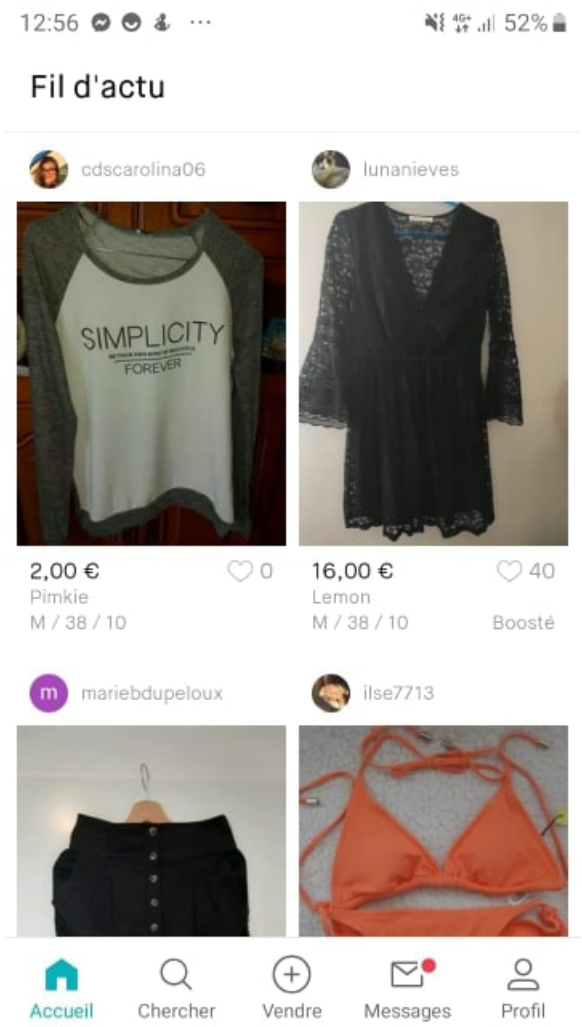


FIGURE 2.2 – Interface Vinted

Le tableau ci-dessous résume notre étude concurrentielle des deux applications mobiles :

	DABCHY	**VINTED**
Site lié à l'application	www.dabchy.com/	www.vinted.fr
Pays où il exerce	Tunisie	11 pays différents dont la France
Type de livraison	Main propre (effectif de l'entreprise)	Livraisons aux points de relais
Type(s) de paiement	En ligne, cash, par carte de crédit.. etc	en ligne ou par poste
Public visé	Femmes et enfants	Tous
Frais de vente	Oui	Non
Notoriété	environs 300 millions de $	+ d'1 milliard de $

TABLE 2.1 – Comparaison de la concurrence

2.5 Les besoins utilisateurs

Pour pouvoir avancer il faut d'abord faire une liste vague des fonctionnalités que l'utilisateur doit attendre de l'application, cette liste sera détaillée dans le BackLog, suivant la méthode SCRUM. Les fonctionnalités seront les suivantes :

— Fonctionnalité d'inscription.
— Fonctionnalité de Login/Connexion.
— Fonctionnalité d'ajout d'article.
— Fonctionnalité de modifier/supprimer un article.
— Fonctionnalité de commande.
— Fonctionnalité de messagerie instantanée.
— Fonctionnalité de recherche d'article.
— Fonctionnalité de notes du vendeur.
— Fonctionnalité de tri des articles.

2.6 Organigramme (Flow Chart)

Pour avoir une bonne vision de l'application à concevoir, et pour bien se départager les tâches, on doit d'abord concevoir, aux côtés du diagramme de cas d'utilisation (voir chapitre 3) , un *Flow Chart* ou un organigramme des interfaces de notre prochaine application 2.3. Ceci se fera avec l'utilisation du logiciel en ligne appelé Whimsical [5], un outil qui permet aux gens de collaborer visuellement sur des idées d'une manière simple, similairement à Dropbox Paper mais avec des organigrammes et des Wireframes au lieu de texte brut :

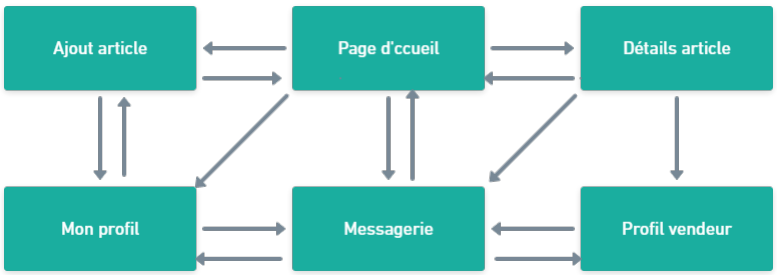


FIGURE 2.3 – Organigramme de l'application

2.7 Charte graphique de l'UI

Il est maintenant possible de mettre en œuvre des interfaces répondant aux heuristiques de Nielsen et Mulich.

Usant du logiciel en ligne Figma, et avec l'aide des multitudes de plugins disponibles et gratuits dans le logiciel, notamment Stark pour une interface répondant aux besoins de personnes atteintes de daltonisme, et Contrast pour la detection automatique des problèmes de contrastes et de couleurs avec étude des ratios, on a décidé de mettre en œuvre les interfaces suivantes, jugées les plus importante en matière d'User Expérience :

2.7.1 Grey Scale UI

Avant de se mettre d'accord sur une palette de couleurs et une typographie adéquate, il est préférable de concevoir des interfaces utilisateurs sur des temps de gris pour visualiser l'emplacement des objets dans l'interface sans être dérangé par d'autres aspects de cette dernière.

FIGURE 2.4 – Interfaces d'inscription sur des temps de gris

FIGURE 2.5 – Interfaces sur des temps de gris

2.7.2 Skeleton Screen

Un écran squelette est une interface utilisateur qui ne contient pas de contenu réel ; au lieu de cela, il affiche les éléments de chargement d'une page sous une forme similaire au contenu réel et ainsi permettre à la page de se charger entièrement sans que l'utilisateur ne s'impatiente.

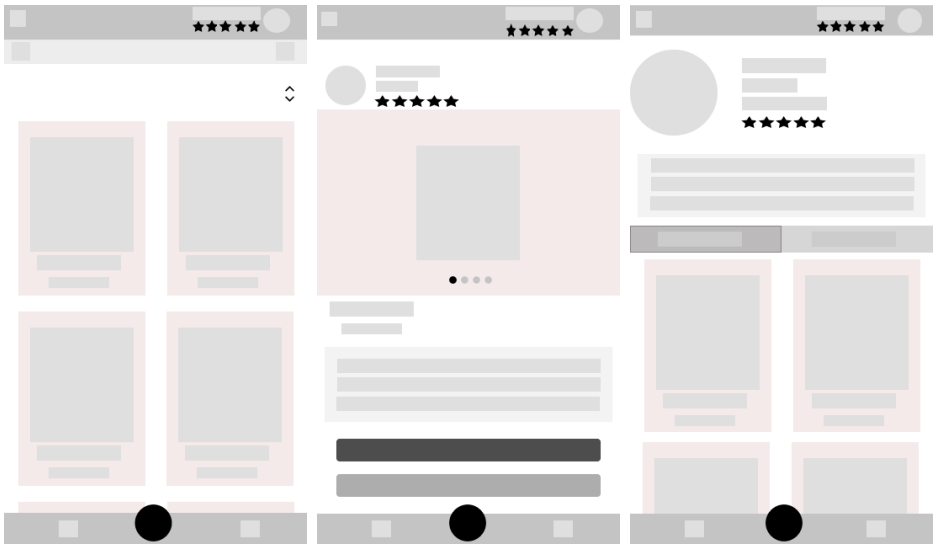


FIGURE 2.6 – Interfaces squelettes

2.7.3 Palettes de couleurs

On a choisi deux palettes servant deux niveaux des interfaces de l'application. La première est une palette contenant les couleurs de la marque, la deuxième sert les fonctionnalités de l'application 2.7 : 71FF02 pour les messages de succès, F95335 pour les messages d'échec ou signalement, FCAF23 pour les notifications importantes (couleur de contraste)

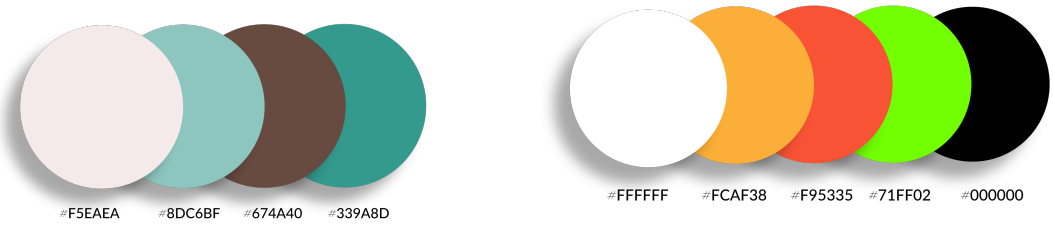


FIGURE 2.7 – Palette principales et Palette secondaire

De la palette principale nous avons créé, grâces aux logiciels Adobe (notamment Adobe Illustrator et Adobe Photoshop), un Logo (figure 2.8) démonstratif du thème de l'application. Il représente deux chapeaux, l'un à tendance féminine et l'autre masculine, accompagné du nom choisi pour l'application 'Gooldies' un mélange entre Good et Old, deux termes anglophone signifiant vieux mais bon faisant référence aux articles d'occasion.

FIGURE 2.8 – Logo de l'application Gooldies

2.7.4 Typographie

Pour mettre en œuvre une application qui inspire la confiance et le sérieux tout en donnant ce côté vêtement et élégance, nous avons décidé d'utiliser la famille de caractères PlayFair Display pour tous les titres des interfaces, accompagnée de la famille Raleway, qui vient encourager le style moderne à travers les sous-titres et les aspects moins importants de l'interface. Enfin, pour rendre lisibles les petits paragraphes, nécessaires à la mise en œuvre de l'interface, tout en restant subtile au changement, la famille Lato nous paraît idéale pour cela.

Nous avons respecté le ratio 3 :1 et 4,5 :1 pour la taille et avons retenu 18/20 pixels pour le contenu en premier plan, 22 pixels pour les titres d'interfaces et 16 pixels pour les paragraphes et autres.

2.7.5 Defect Screen

Répondant à l'heuristique 5 de N&M cette interface existe pour tout utilisateur ayant emprunté un chemin défectueux qui n'affiche rien dans l'application (figure 2.9).

FIGURE 2.9 – Interface de chemin defectueux

2.7.6 OnBoarding

Pour répondre aux besoins de la documentation et aider les nouveaux utilisateurs à comprendre le fonctionnement de notre application, nous avons décidé de mettre en œuvre un bref OnBoarding de type Fonctionnel (voir chapitre 1 sur les types d'OnBoarding). Nous avons utilisé des images vectorielles libre d'utilisation [6].

FIGURE 2.10 – Interfaces d'OnBoarding

2.7.7 Types de boutons

Nous avons conçu trois types de boutons pour les interfaces : un bouton non-cliqué en attente d'action (1), un bouton OK communiquant le bon fonctionnement de la requête et le passage à l'interface suivante, si nécessaire (2), et un bouton cliqué qui procède l'exécution de sa fonctionnalité, urgent l'utilisateur de patienter (3).

FIGURE 2.11 – Types de boutons (1) (2) (3)

2.7.8 Prototype des Interfaces finales

De cette étude de la charte graphique de l'application résulte le prototypage suivant de l'application, telle qu'elle sera conçue.

FIGURE 2.12 – Prototype des interfaces d'inscription finales

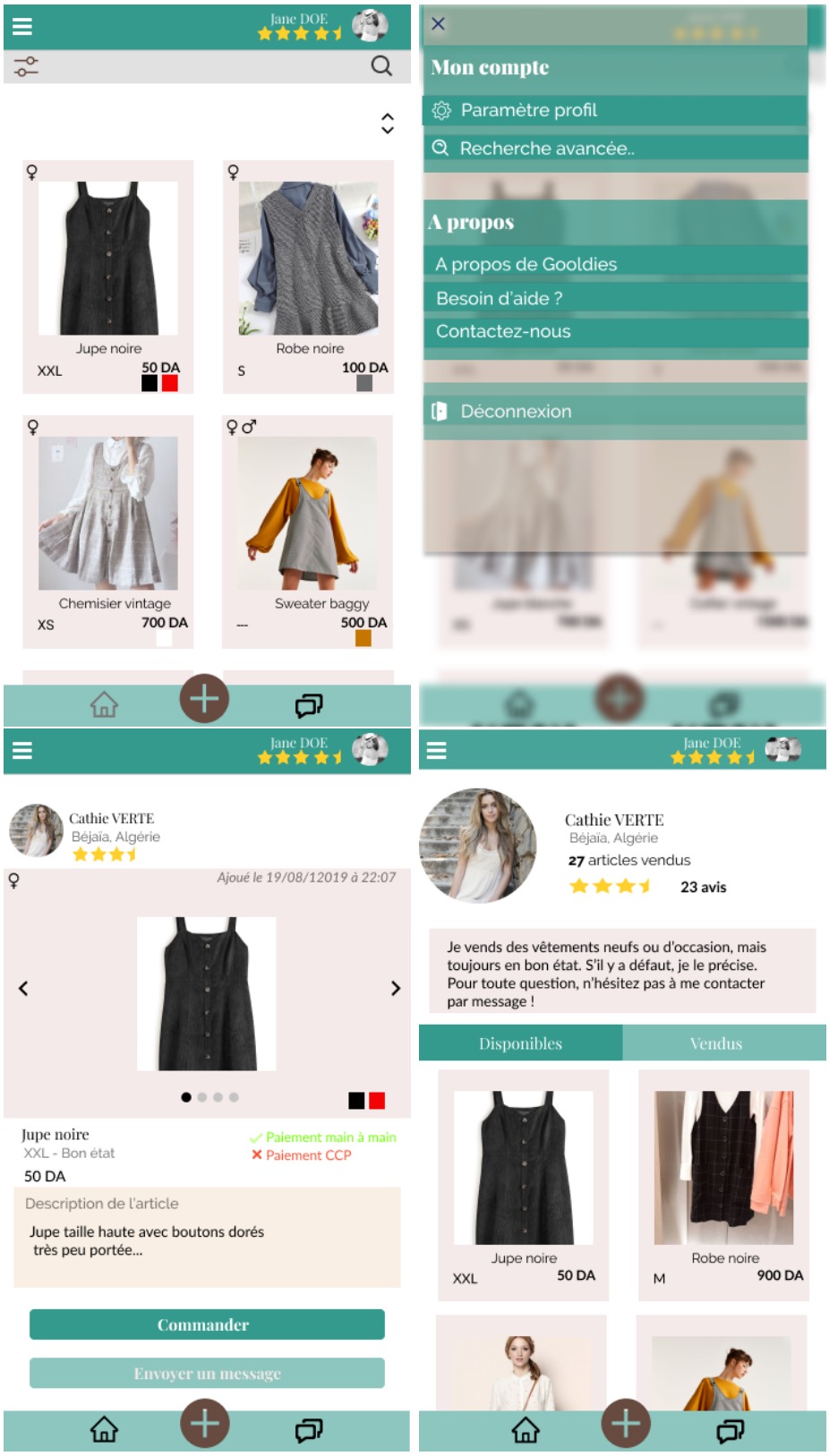


FIGURE 2.13 – Prototype des interfaces finales (page d'accueil, menu, article, profil vendeur)

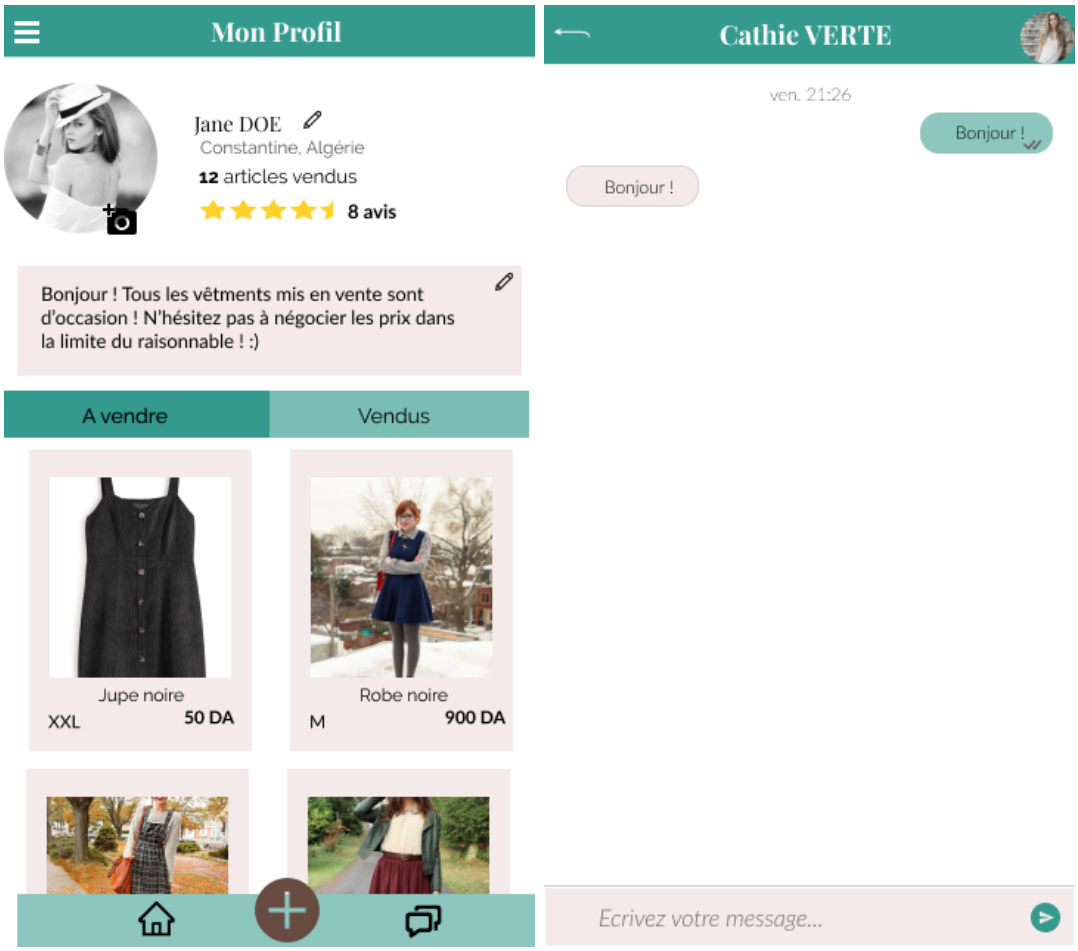


FIGURE 2.14 – Prototype des interfaces finales (profil utilisateur, messagerie)

2.8 Méthode de conception et palnification

Comme le veut la méthode de conception SCRUM, c'est à travers un product Backlog qu'on détaillera les fonctionnalités et besoins de l'applications en tant qu'items pour ainsi passer à la conception détaillée de l'application et planifier le bon fonctionnement de la création de l'application.

2.8.1 Les items

ID item	Nom item	Description item	Priorité de l'item
ID1	Fonctionnalité d'inscription	nécessaire pour l'utilisation de l'application pour maximiser la sécurité de l'application, passe sous 4 formulaires de 3 champs avec l'activation par numéro de téléphone obligatoire	P1
ID2	Fonctionnalité de Login/Connexion	pour toute personne voulant avoir accès à son compte personnel via d'autres appareils après son inscription	P1
ID3	Fonctionnalité d'ajout d'article	un utilisateur doit pouvoir ajouter un article en entrant des champs nécessaire à sa mise en ligne correcte	P2
ID4	Fonctionnalité de modifier/supprimer un article	l'utilisateur qui a déjà mis en vente un article (vendeur) doit avoir la possibilité de le modifier ou le supprimer entièrement de la vente, et ce s'il n'est pas commandé ou bloqué	P3
ID5	Fonctionnalité de commande d'article	un utilisateur doit pouvoir commander un article si celui-ci est disponible (non-commandé ou bloqué) et si le premier répond aux critères de commande (mode de paiement possible, num CCP renseigné etc.)	P2
ID6	Fonctionnalité de messagerie instantanée	un utilisateur peut communiquer avec d'autres utilisateurs via la messagerie instantanée de l'application et ce de façon privée et sécurisée	P3
ID7	Fonctionnalité de notation	un acheteur doit avoir la possibilité de noter son vendeur, mais seulement à la fin d'une transaction réussie, et ce s'il le souhaite	P4
ID8	Fonctionnalité de recherche	un utilisateur peut accéder à la recherche d'un article avec des mots-clés qui pourraient se trouver dans le titre ou la description d'un article, ou dans l'un de ses attributs (couleurs, taille, etc.)	P4
ID9	Fonctionnalité de tri des articles	les articles affichés dns la page d'accueil doivent pouvoir être triés selon le choix de l'utilisateur (catégorie, sexe, taille, couleurs, etc.)	P3

TABLE 2.2 – Liste des items du Backlog

2.8.2 Besoins non-fonctionnels de l'application

— L'application Web devrait être capable de gérer 20 millions d'utilisateurs, avec une incidence sur ses performances, et ce lors de son déploiement.

— La confidentialité des informations, l'exportation de technologies restreintes, les droits de propriété intellectuelle, etc. doivent être vérifiés.

— Les données délicates entrées par les utilisateurs doivent rester strictement privées (numéro du compte CCP, numéro de téléphone, l'adresse exacte)

— L'application doit s'adapter à toutes les versions du système d'exploitation Android qui sont toujours actives (de JellyBean (2013) à Android10 (2019))

— L'application doit être flexible et de ce fait peut être mise-à-jour (base de données, nouvelles fonctionnalités, etc).

— Enfin, l'application doit tourner en réseau pour le bon fonctionnement de la messagerie, et la mise à jour de l'état des articles (commandé, bloqué, modifié, supprimé, etc), et ce après son déploiement.

2.8.3 Estimation Initiale (Retro Planning)

Dans une équipe de deux personnes, avec un Backlog de 9 items, nous avons estimé, théoriquement, une marge de travail et le temps nécessaire pour mettre en œuvre le sprint, avec l'utilisation de la méthode du Planning Poker (comme mentionné dans le chapitre 1, section SCRUM), en consacrant un certain temps pour le développement, un autre pour les tests de regressions automatiques (Chapitre 1 SCRUM) et un autre pour le débogage après chaque item. Nous avons obtenu ce qui suit :

Tâche à accomplir	Responsables	État
Création des Interfaces User		
Création des interfaces GreyScale	Saïdi	Terminée
Réunion et changements	Adjabi, Saïdi	Terminée
Définition de la charte graphique	Adjabi, Saïdi	Terminée
Mise en œuvre des prototypes finaux	Adjabi	Terminée
test UX (heuristiques de Nielsen)	Adjabi	Terminée
Création des interfaces secondaires	Saïdi	Terminée
Mise en œuvre des Logos	Adjabi	Terminée
Réunion et prise de décisions finales	Saïdi, Adjabi	Terminée
Conception		
Conception des diags. de seq. Sys.	Saïdi	Terminée
Réunion et correction	Adjabi, Saïdi	Terminée

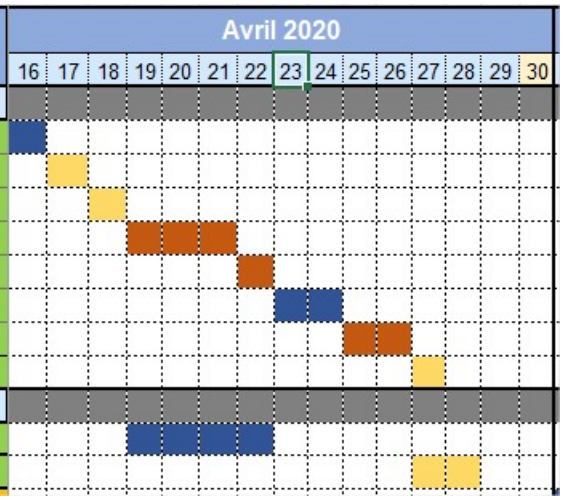
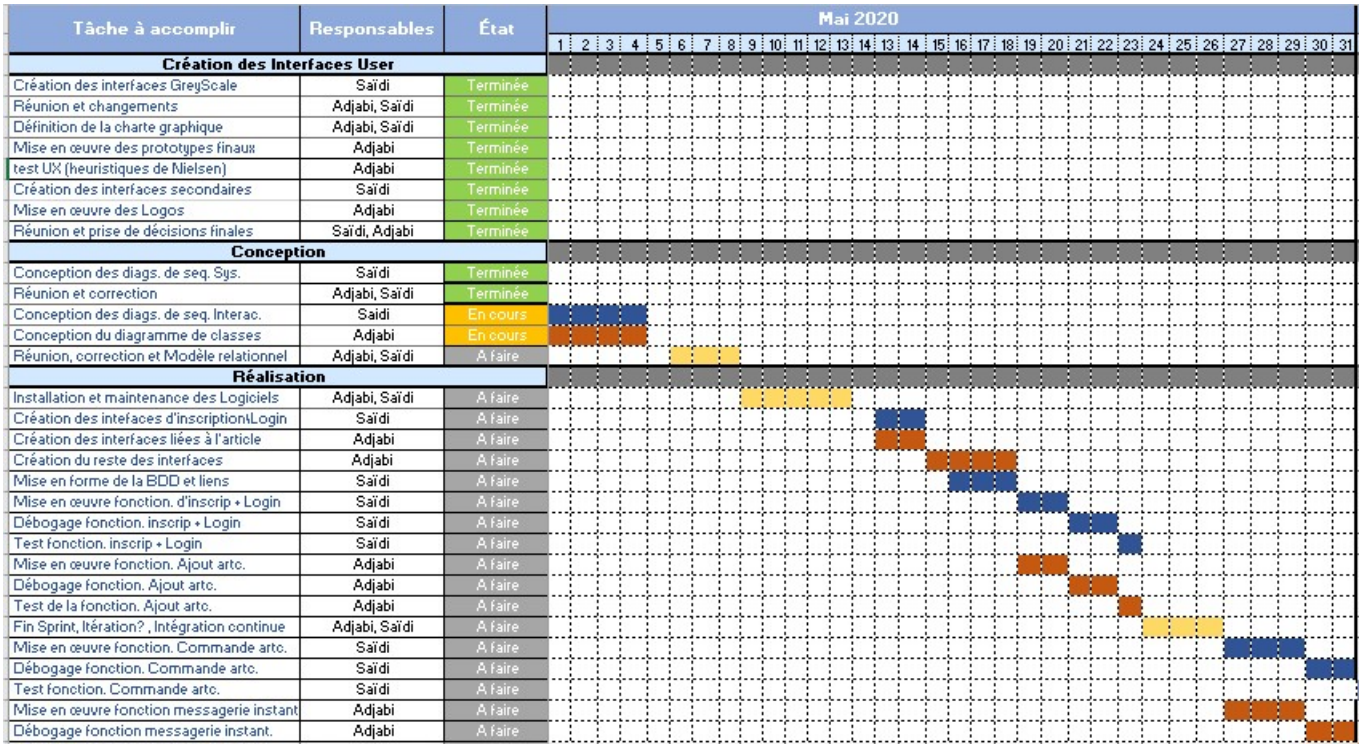
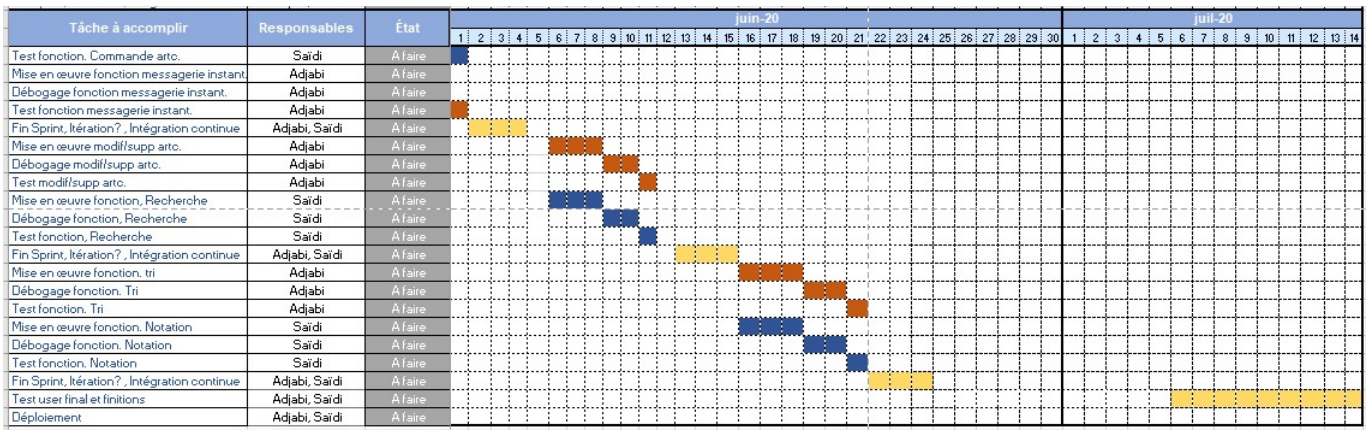


FIGURE 2.15 – Planning du mois d'Avril



Tâche à accomplir	Responsables	État
Création des Interfaces User		
Création des interfaces GreyScale	Saïdi	Terminée
Réunion et changements	Adjabi, Saïdi	Terminée
Définition de la charte graphique	Adjabi, Saïdi	Terminée
Mise en œuvre des prototypes finaux	Adjabi	Terminée
test UX (heuristiques de Nielsen)	Adjabi	Terminée
Création des interfaces secondaires	Saïdi	Terminée
Mise en œuvre des Logos	Adjabi	Terminée
Réunion et prise de décisions finales	Saïdi, Adjabi	Terminée
Conception		
Conception des diags. de seq. Sys.	Saïdi	Terminée
Réunion et correction	Adjabi, Saïdi	Terminée
Conception des diags. de seq. Interac.	Saidi	En cours
Conception du diagramme de classes	Adjabi	En cours
Réunion, correction et Modèle relationnel	Adjabi, Saïdi	A faire
Réalisation		
Installation et maintenance des Logiciels	Adjabi, Saïdi	A faire
Création des intefaces d'inscription\Login	Saïdi	A faire
Création des interfaces liées à l'article	Adjabi	A faire
Création du reste des interfaces	Adjabi	A faire
Mise en forme de la BDD et liens	Saïdi	A faire
Mise en œuvre fonction. d'inscrip + Login	Saïdi	A faire
Débogage fonction. inscrip + Login	Saïdi	A faire
Test fonction. inscrip + Login	Saïdi	A faire
Mise en œuvre fonction. Ajout artc.	Adjabi	A faire
Débogage fonction. Ajout artc.	Adjabi	A faire
Test de la fonction. Ajout artc.	Adjabi	A faire
Fin Sprint, Itération? , Intégration continue	Adjabi, Saïdi	A faire
Mise en œuvre fonction. Commande artc.	Saïdi	A faire
Débogage fonction. Commande artc.	Saïdi	A faire
Test fonction. Commande artc.	Saïdi	A faire
Mise en œuvre fonction messagerie instant	Adjabi	A faire
Débogage fonction messagerie instant.	Adjabi	A faire

FIGURE 2.16 – Planning du mois de Mai



Tâche à accomplir	Responsables	État
Test fonction. Commande artc.	Saïdi	A faire
Mise en œuvre fonction messagerie instant	Adjabi	A faire
Débogage fonction messagerie instant.	Adjabi	A faire
Test fonction messagerie instant.	Adjabi	A faire
Fin Sprint, Itération? , Intégration continue	Adjabi, Saïdi	A faire
Mise en œuvre modif/supp artc.	Adjabi	A faire
Débogage modif/supp artc.	Adjabi	A faire
Test modif/supp artc.	Adjabi	A faire
Mise en œuvre fonction, Recherche	Saïdi	A faire
Débogage fonction, Recherche	Saïdi	A faire
Test fonction, Recherche	Saïdi	A faire
Fin Sprint, Itération? , Intégration continue	Adjabi, Saïdi	A faire
Mise en œuvre fonction. tri	Adjabi	A faire
Débogage fonction. Tri	Adjabi	A faire
Test fonction. Tri	Adjabi	A faire
Mise en œuvre fonction. Notation	Saïdi	A faire
Débogage fonction. Notation	Saïdi	A faire
Test fonction. Notation	Saïdi	A faire
Fin Sprint, Itération? , Intégration continue	Adjabi, Saïdi	A faire
Test user final et finitions	Adjabi, Saïdi	A faire
Déploiement	Adjabi, Saïdi	A faire

FIGURE 2.17 – Planning des mois de Juin et Juillet

Le Retro Planning s'est fait en date du 30/04/2020 sur le logiciel Excel.

A la fin de chaque itération, on mettra en œuvre une réunion pour faire des démonstrations et des critiques constructives. Une fois satisfaits du résultat d'un Sprint, d'autres tests se font avec des utilisateurs finaux de notre entourage. Une intégration continue sera nécessaire pour éviter tout conflit de code.

Ci-dessous (figure 2.18), la légende du planning :

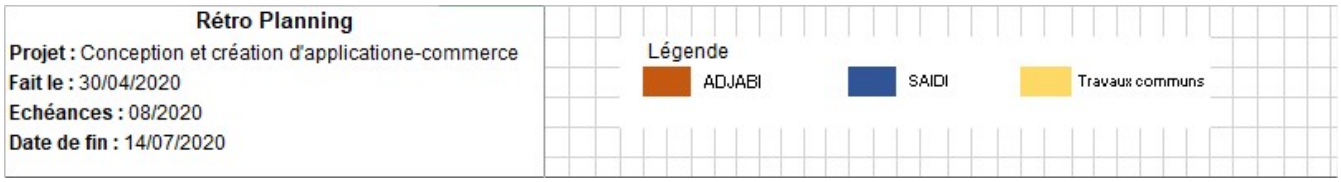


FIGURE 2.18 – Légende du planning via Excel

2.9 Conclusion

La réalisation d'un cahier des charges complet et détaillé est l'une des clés pour un projet bien organisé et clair pour toute l'équipe, et une documentation pour toute personne qui serait curieuse du déroulement du projet. Mais un cahier des charges n'est pas suffisant pour commencer, la réalisation de l'application doit être accompagnée de diagrammes explicatifs de chaque détail des fonctionnalités sécifiques du système. Ce qui nous introduit au chapitre suivant.

Chapitre 3

Conception de l'application

3.1 Introduction

Nous allons, dans ce chapitre, présenter la conception détaillée de notre application, où nous présentons les différents diagrammes des cas d'utilisation, de séquence système, d'interaction, et de classes, ainsi que le modèle relationnel.

3.2 Analyse conceptuelle

3.2.1 Identification des acteurs

Ci-dessous, nous énumérons les acteurs qui interagissent avec le système avec pour chacun une brève explication.

— **Visiteur :** un individu accédant à l'application sans disposer d'un compte.

— **Utilisateur :** tout individu s'étant inscrit à l'application. Un utilisateur peut profiter de toutes les fonctionnalités de l'application.

— **Vendeur :** c'est un utilisateur qui a ajouté un ou plusieurs articles à vendre sur son profil.

3.2.2 Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation permettent d'exprimer les besoins des utilisateurs d'un système informatique, et ce en illustrant les fonctionnalités et services rendus par ce dernier, sans pour autant préciser leurs modes de réalisation et ses interactions possibles avec les acteurs. Ces diagrammes résultent de la mise-en-oeuvre de la liste des besoins de l'utilisateur et sont considérés comme des diagrammes côté user.

Le cas d'utilisation "Consulter articles"

Ci-dessous (tableau 3.1), nous décrivons le cas d'utilisation "Consulter articles".

Cas d'utilisation	Consulter articles
Résumé	Permet à l'utilisateur de consulter les articles disponibles.
Acteur	Utilisateur
Pré-condition	S'authentifier.
Scénario nominal	**[Début]** 1. L'utilisateur demande de consulter les articles. 2. Le système affiche l'interface demandée. 3. L'utilisateur demande d'afficher les détails d'un article. 4. Le système affiche les détails de l'article sélectionné. **[Fin]**
Alternatives	Aucune.

TABLE 3.1 – Description du cas d'utilisation "Consulter détails article".

Diagramme du cas d'utilisation "Consulter articles"

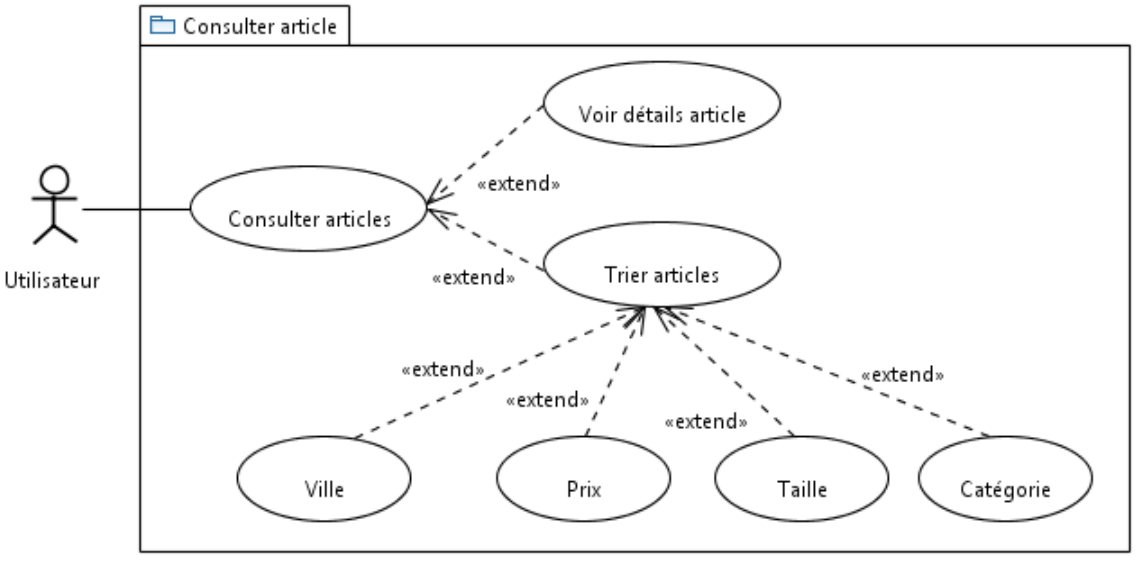


FIGURE 3.1 – Diagramme du cas d'utilisation "Consulter articles".

Le cas d'utilisation "Rechercher articles"

Nous décrivons, ci-dessous (tableau 3.2), le cas d'utilisation "Rechercher articles".

Cas d'utilisation	Rechercher articles
Résumé	Permet à l'utilisateur de rechercher des articles grâce à des mots-clés.
Acteur	Utilisateur
Pré-condition	S'authentifier
Scénario nominal	**[Début]** 1. L'utilisateur saisit des mots-clés dans la barre de recherche puis valide. 2. Le système recherche les articles correspondants aux mots-clés. 3. Le système affiche le résultat. **[Fin]**
Alternative	S'il n'existe pas de résultat, alors le système affiche un message de non-existence de résultat pour la recherche.

TABLE 3.2 – Description du cas d'utilisation "Rechercher articles".

Le cas d'utilisation "S'inscrire"

Ci dessous (tableau 3.3), nous décrivons le cas d'utilisation "S'inscrire".

Cas d'utilisation	S'inscrire
Résumé	Permet au visiteur de s'inscrire.
Acteur	Visiteur
Pré-condition	Aucune.
Scénario nominal	**[Début]** 1. Le visiteur demande le formulaire d'inscription. 2. Le système affiche le formulaire d'inscription (demande de coordonnées, email, mot de passe). 3. Le visiteur saisit les informations demandées puis valide. 4. Le système vérifie la conformité des informations fournies. 5. Le système affiche le formulaire du numéro de téléphone. 6. Le visiteur saisit son numéro de téléphone puis valide. 7. Le système vérifie la conformité de la donnée fourie. 8. Le système envoie un code d'activation au numéro fourni puis affiche l'interface "saisie code". 9. Le visiteur saisit le code reçu, puis valide. 10. Le système vérifie la conformité du code fourni puis enregistre le compte dans la base de données. 11. Le système affiche l'interface "accueil". **[Fin]**
Alternative	1. Si les informations fournies sont incomplètes ou incorrectes(incompatibilité avec le type de champ, champ vide), le système réaffiche le formulaire d'inscription avec un message d'erreur. 2. Si le numéro de téléphone fourni est incorrect(incompatibilité avec le type de champ, champ vide), le système réaffiche le formulaire "numéro de téléphone" avec un message d'erreur. 3. Si le code entré est erroné, le système réaffiche l'interface "saisie code" avec un message d'erreur.

TABLE 3.3 – Description du cas d'utilisation "S'inscrire".

Diagramme du cas d'utilisation "S'inscrire"

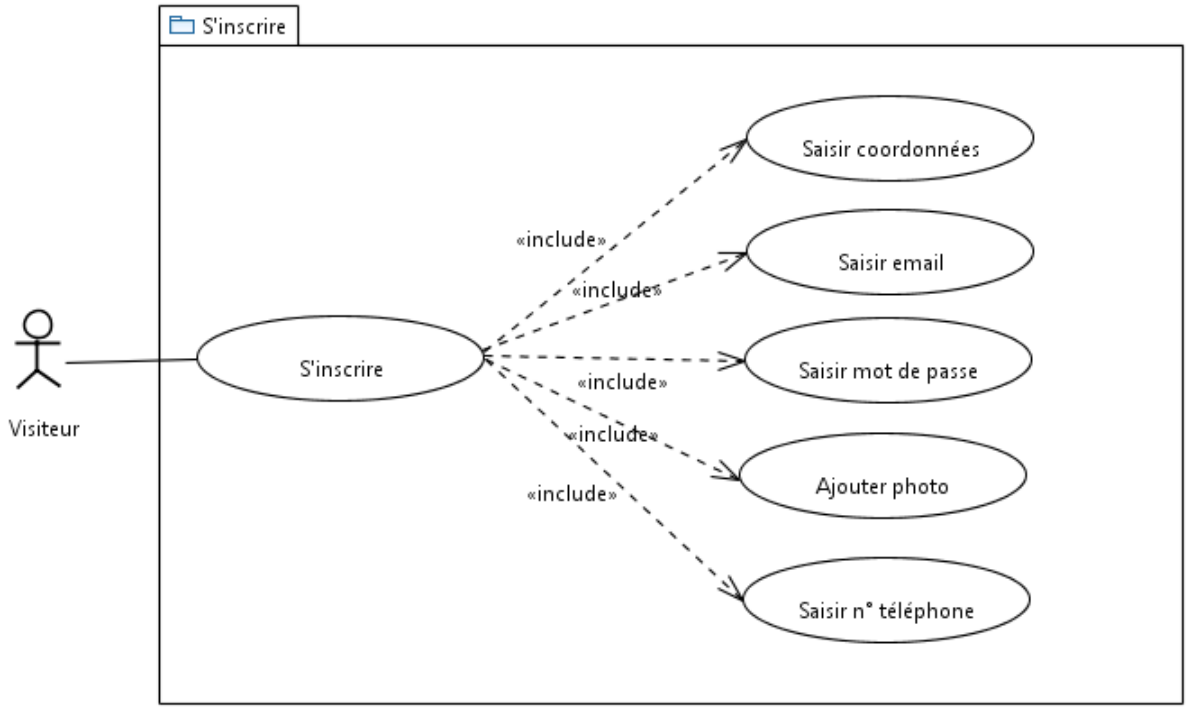


FIGURE 3.2 – Diagramme du cas d'utilisation "S'inscrire".

Le cas d'utilisation "S'authentifier"

Cas d'utilisation	S'authentifier
Résumé	Permet à l'utilisateur de s'authentifier.
Acteur	Utilisateur
Pré-condition	Disposer d'un compte.
Scénario nominal	**[Début]** 1. L'utlisateur demande le formulaire d'authentification. 2. Le système affiche le formulaire d'authentification (demande d'email et de mot de passe). 3. L'utilisateur saisit les informations demandées puis valide. 4. Le système vérifie la conformité des informations fournies. 5. Le système affiche l'interface "accueil". **[Fin]**
Alternative	Si les informations fournies sont incomplètes ou incorrectes (champ vide, incompatibilité avec le type de champ), le système réaffiche le formulaire d'authentification avec un message d'erreur.

TABLE 3.4 – Description du cas d'utilisation "S'authentifier".

Diagramme du cas d'utilisation "S'authentifier"

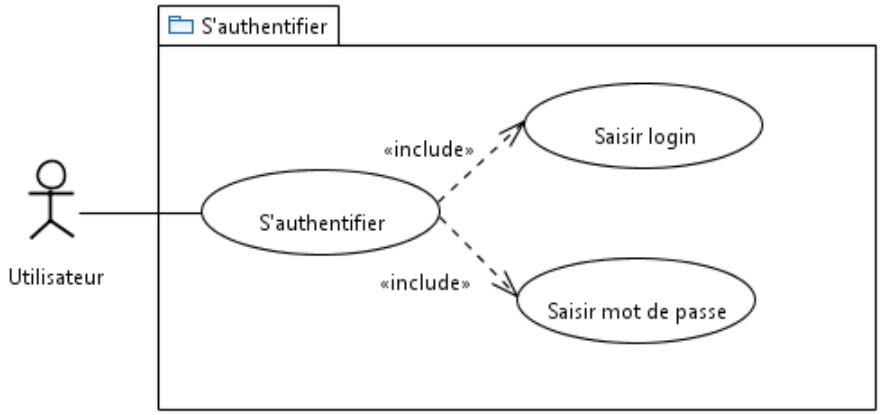


FIGURE 3.3 – Diagramme du cas d'utilisation "S'authentifier".

Le cas d'utilisation "Commander"

Nous décrivons, ci-dessous (tableau 3.5), le cas d'utilisation "Commander".

Cas d'utilisation	Commander
Résumé	Permet à l'utilisateur de commander un article.
Acteur	Utilisateur
Pré-condition	S'authentifier.
Scénario nominal	**[Début]** 1. L'utilisateur demande de consulter les détails d'un article. 2. Le système affiche l'interface "détails article". 3. L'utilisateur choisit le mode de paiement puis commande l'article. 4. Le système demande la confirmation de la requête. 5. L'utilisateur confirme la requête. 6. Le système envoie une notification au vendeur et demande de confirmer la commande. 7. Le vendeur confirme la commande. 8. Le système réserve l'article à l'utilisateur et vérifie le paiement. 9. L'utilisateur effectue le paiement vers un compte intermédiaire. 10. Le système notifie le vendeur pour qu'il envoie le colis. 11. Le vendeur envoie le colis et envoie le récépissé de dépôt d'envoi. 12. Le système enregistre la commande et notifie l'utilisateur de l'envoi du colis. 13. Le système enlève l'article de la vente. **[Fin]**
Alternative	1. Si à 3. l'utilisateur choisit le paiement en main-à-main, alors le système redirige l'utilisateur vers la messagerie afin qu'il communique avec le vendeur directement. 2. Si après 3 jours de réservation, l'utilisateur n'effectue pas de paiement, alors le système annulera la commande. 3. Si après 3 jours le vendeur n'envoie pas le colis, alors le système annulera la commande et remboursera l'utilisateur.

Table 3.5 – Description du cas d'utilisation "Commander".

Le cas d'utilisation "Utiliser messagerie"

Nous décrivons, dans le tableau ci-dessous, le cas d'utilisation "Utiliser messagerie".

Cas d'utilisation	Utiliser messagerie
Résumé	Permet à l'utilisateur d'envoyer un message à un autre utilisateur.
Acteur	Utilisateur
Pré-condition	S'authentifier.
Scénario nominal	**[Début]** 1. L'utilisateur demande de consulter le profil d'un vendeur. 2. Le système affiche l'interface "profil vendeur". 3. L'utilisateur demande d'envoyer un message. 4. Le système affiche l'interface "messagerie". 5. L'utilisateur saisit un message et l'envoie. **[Fin]**
Alternative	Aucune.

TABLE 3.6 – Description du cas d'utilisation "Utiliser messagerie".

Diagramme du cas d'utilisation "Utiliser messagerie"

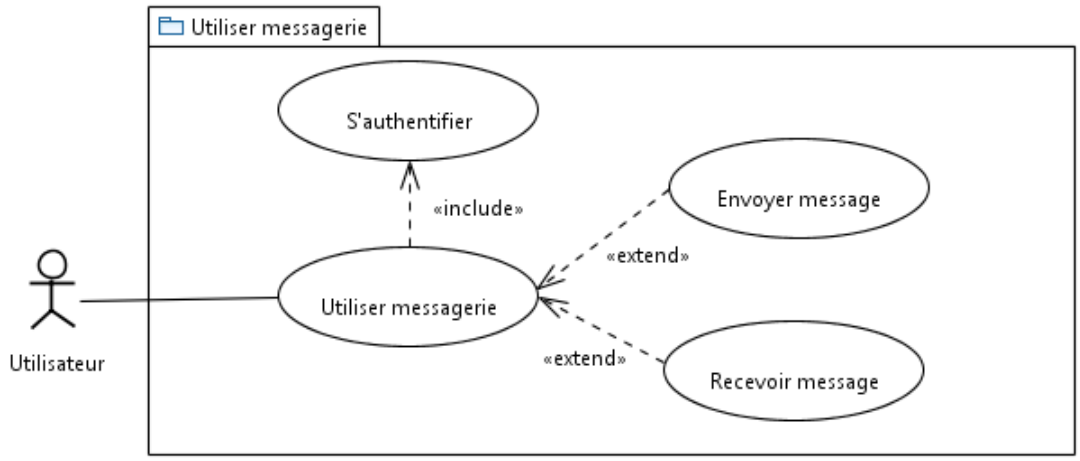


FIGURE 3.4 – Diagramme du cas d'utilisation "Utiliser messagerie".

Le cas d'utilisation "Ajouter article"

Cas d'utilisation	Ajouter article
Résumé	Permet au vendeur de publier un article à mettre en vente.
Acteur	Vendeur
Pré-condition	S'authentifier.
Scénario nominal	**[Début]** 1. Le vendeur demande d'ajouter un article. 2. Le système affiche l'interface d'ajout d'article. 3. Le vendeur saisit les informations de l'article (photo, description, catégorie, état, prix, mode de paiement) puis valide. 4. Le système enregistre l'article dans la base de données et affiche un message de succès. **[Fin]**
Alternative	Si les informations sont incorrectes ou incomplètes (incompatibilité avec le type de champ, champ vide), le système réaffiche l'interface d'ajout d'article avec un message d'erreur.

TABLE 3.7 – Description du cas d'utilisation "Ajouter article".

Diagramme du cas d'utilisation "Ajouter article"

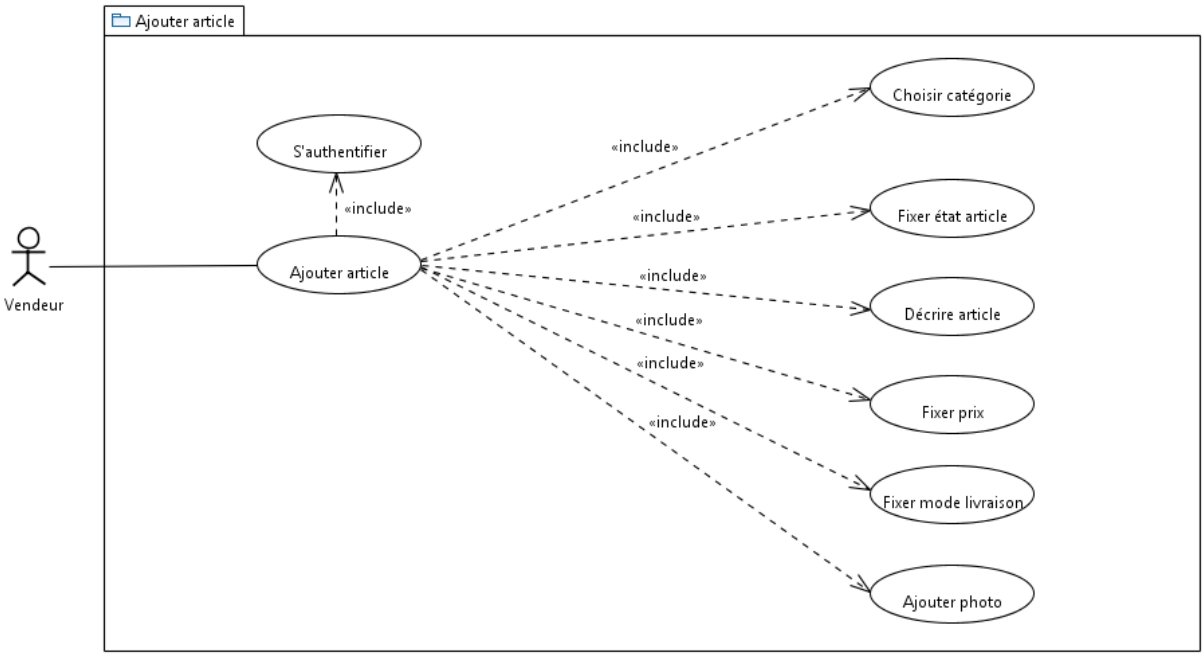


FIGURE 3.5 – Diagramme du cas d'utilisation "Ajouter article"

Diagramme de cas d'utilisation général

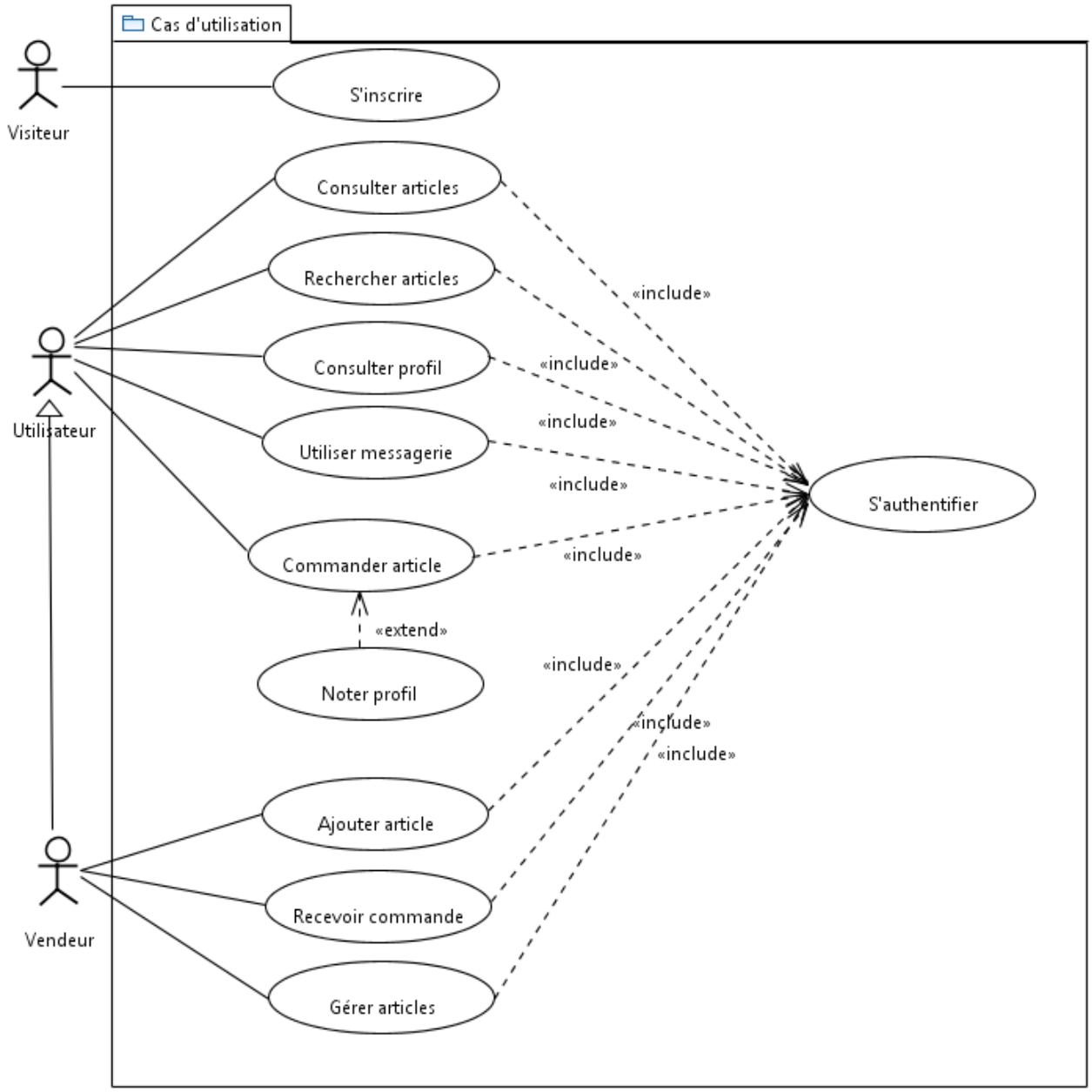


FIGURE 3.6 – Diagramme de cas d'utilisation général

3.2.3 Diagrammes de séquence

Les diagrammes de séquences permettent de montrer les interactions d'objets dans le cadre d'un scénario d'un diagramme de cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système.

Diagrammes de séquence système

Le diagramme de séquence système considère le système comme étant une "boîte noire" vue de l'extérieur (par les acteurs) sans détails quant à la manière de son fonctionnement.

Diagramme de séquence système "S'inscrire"

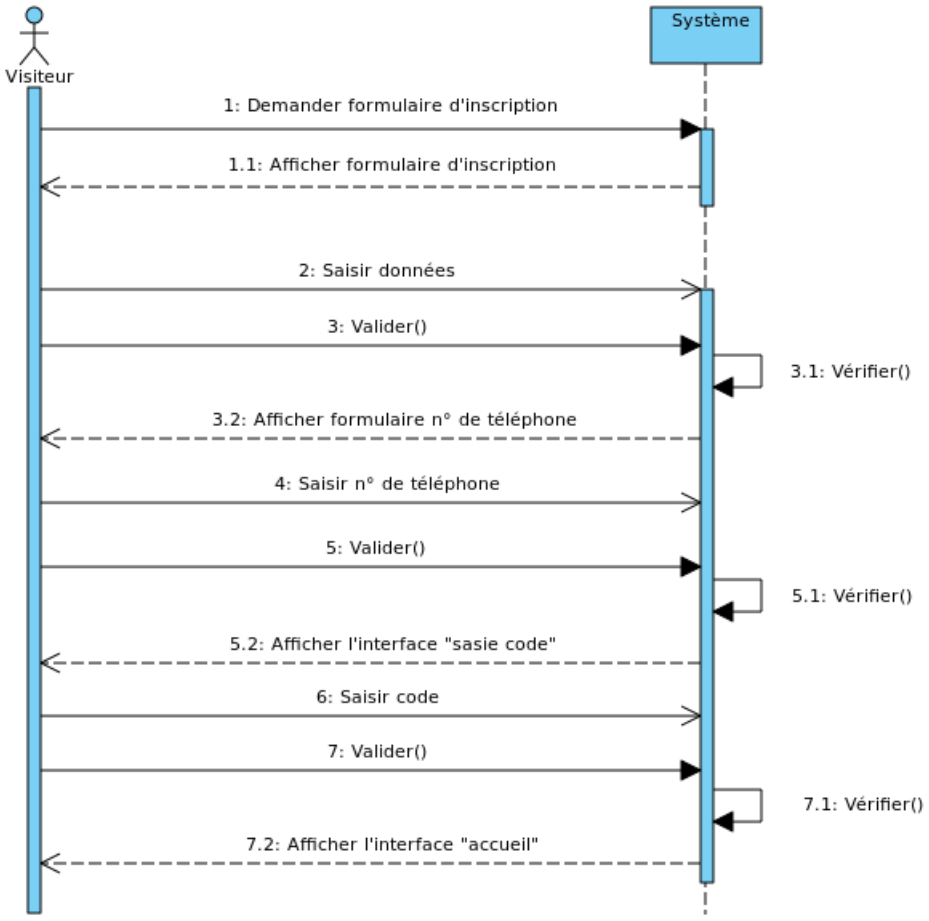


FIGURE 3.7 – Diagramme de séquence système "S'inscrire"

Diagramme de séquence système "S'authentifier"

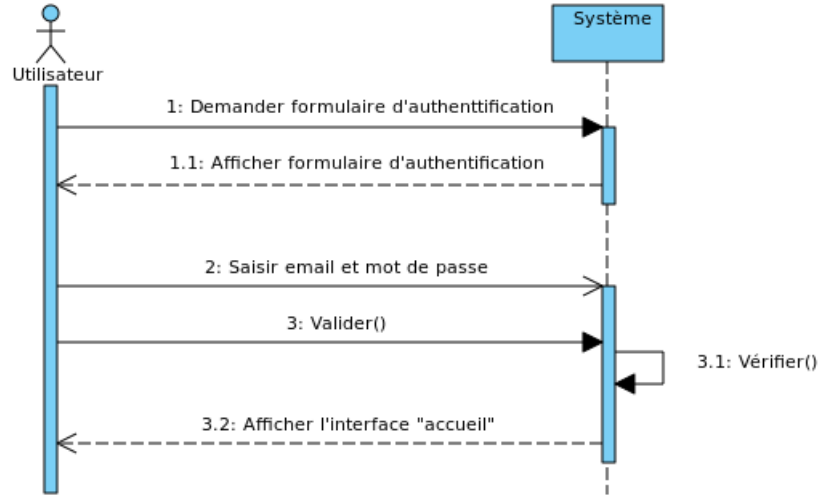


FIGURE 3.8 – Diagramme de séquence système "S'authentifier"

Diagramme de séquence système "Envoyer message"

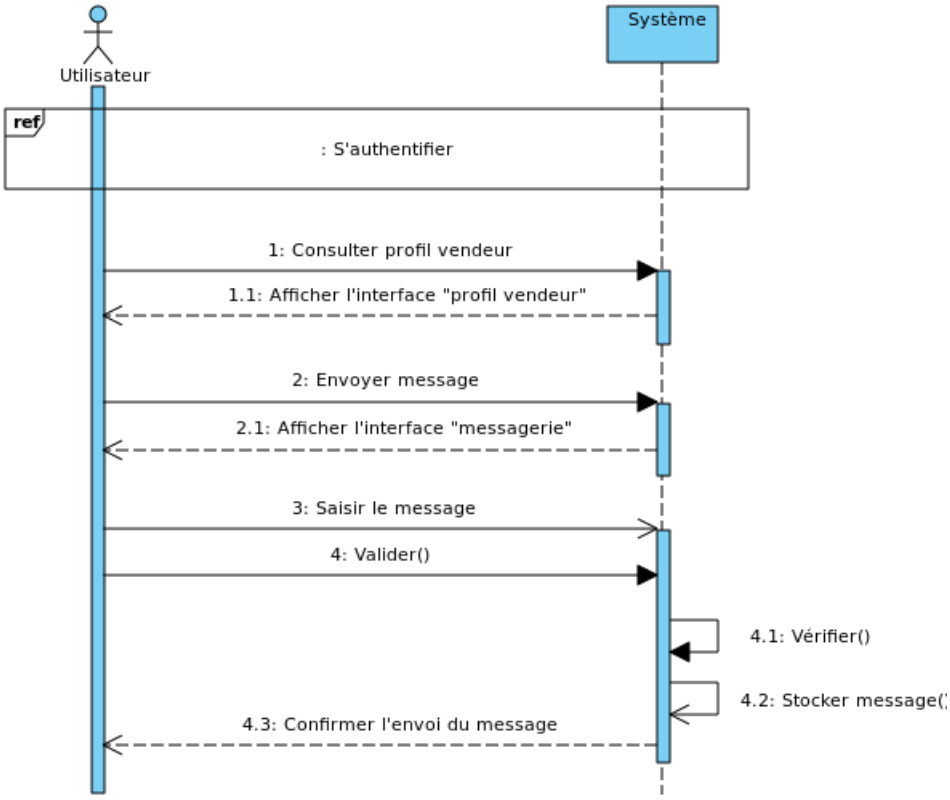


FIGURE 3.9 – Diagramme de séquence système "Envoyer message"

Diagramme de séquence système "Recevoir message"

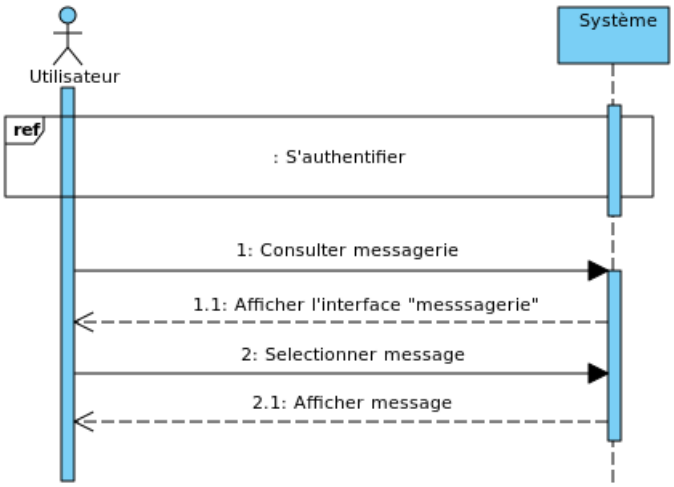


FIGURE 3.10 – Diagramme de séquence système "Recevoir message"

Diagramme de séquence système "Consulter articles"

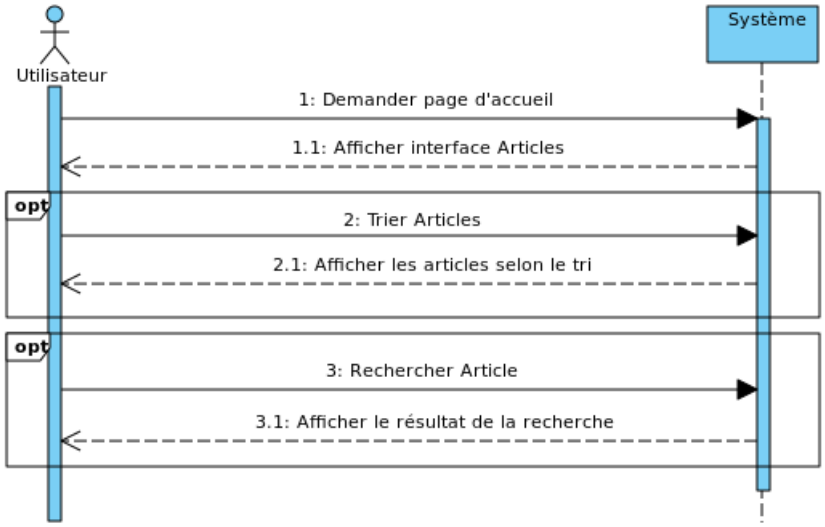


FIGURE 3.11 – Diagramme de séquence système "Consulter articles"

Diagramme de séquence système "Commander"

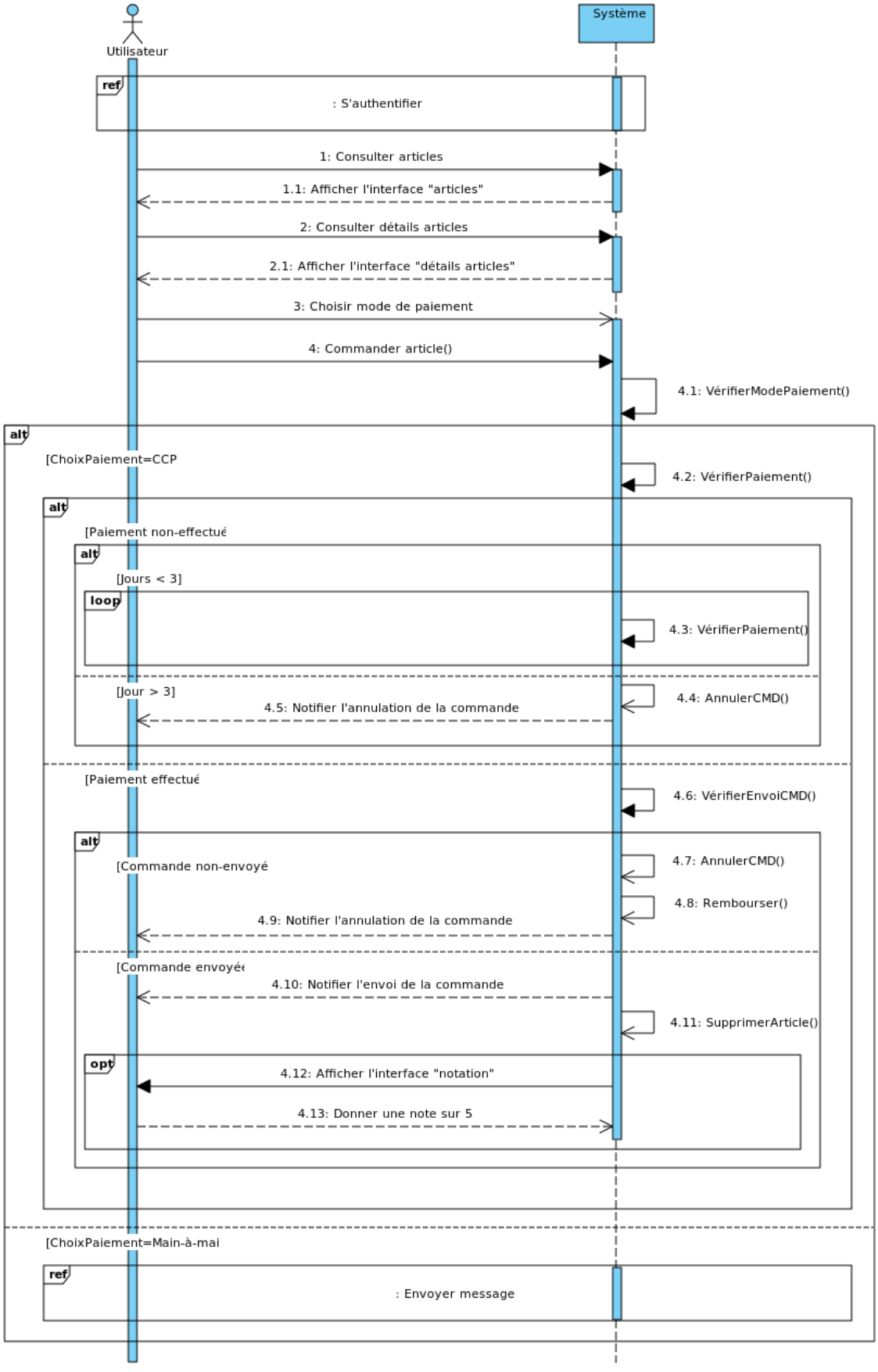


FIGURE 3.12 – Diagramme de séquence système "Commander"

Diagramme de séquence système "Recevoir commande"

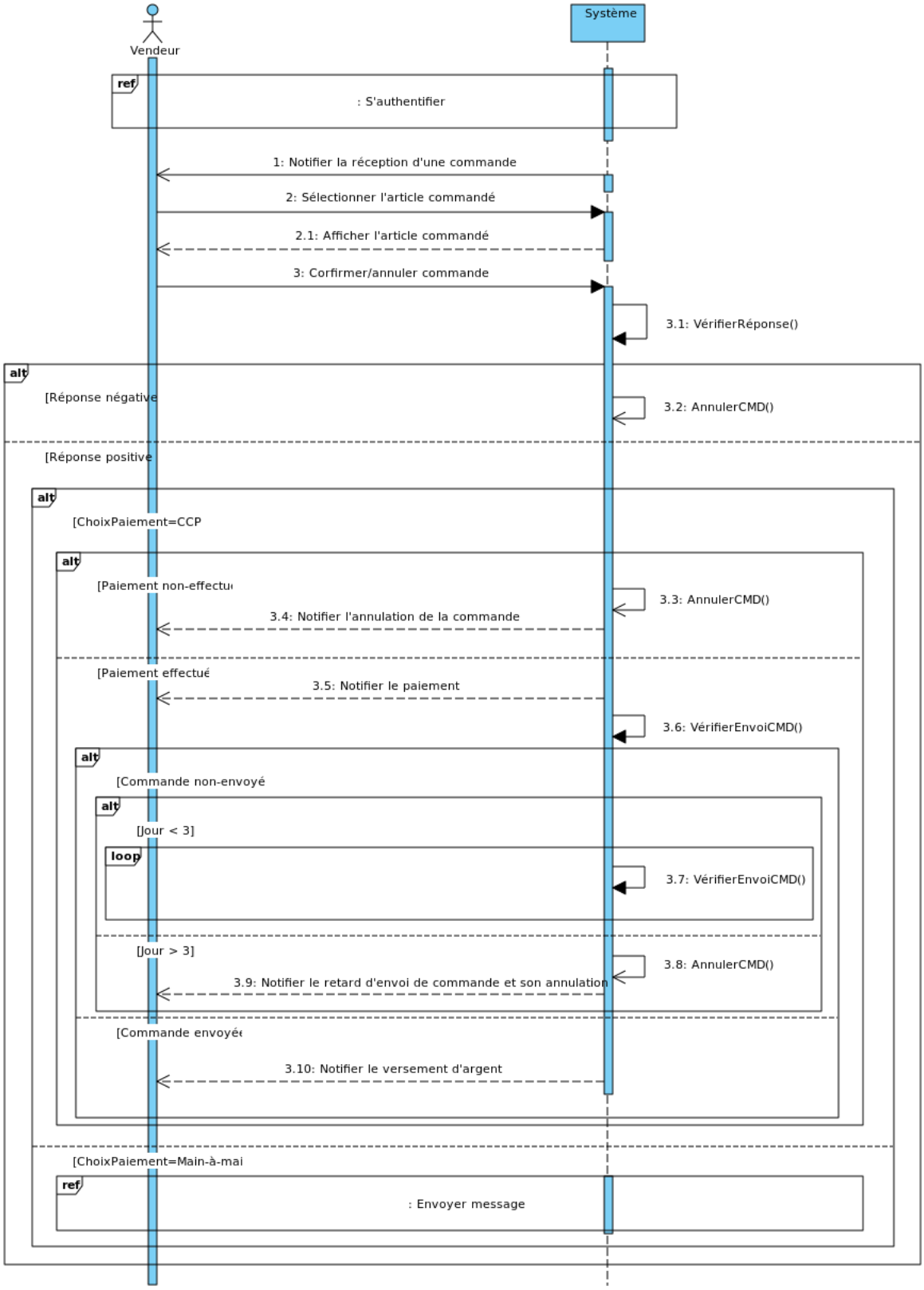


FIGURE 3.13 – Diagramme de séquence système "Recevoir commande"

Diagramme de séquence système "Ajouter article"

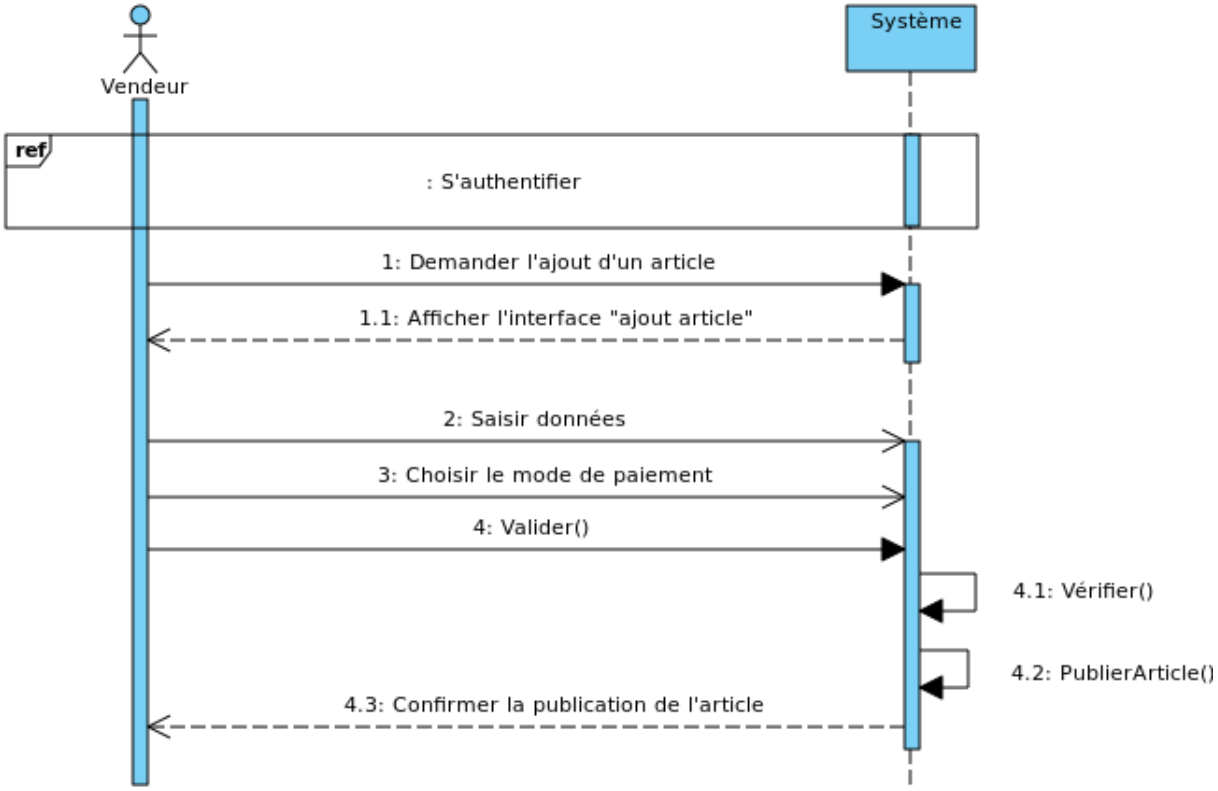


FIGURE 3.14 – Diagramme de séquence système "Ajouter article"

Diagrammes d'interaction

Les diagrammes d'interaction, quant à eux, permettent d'établir un lien entre les diagrammes de cas d'utilisation et le diagramme de classes, et offrent de précieux détails quant au déclanchement réalisé après chaque requête faite par l'acteur. Ces diagrammes illustrent la sollicitation de la base de données, et la communication entre interfaces et fonctionnalités.

Diagramme d'interaction "S'inscrire"

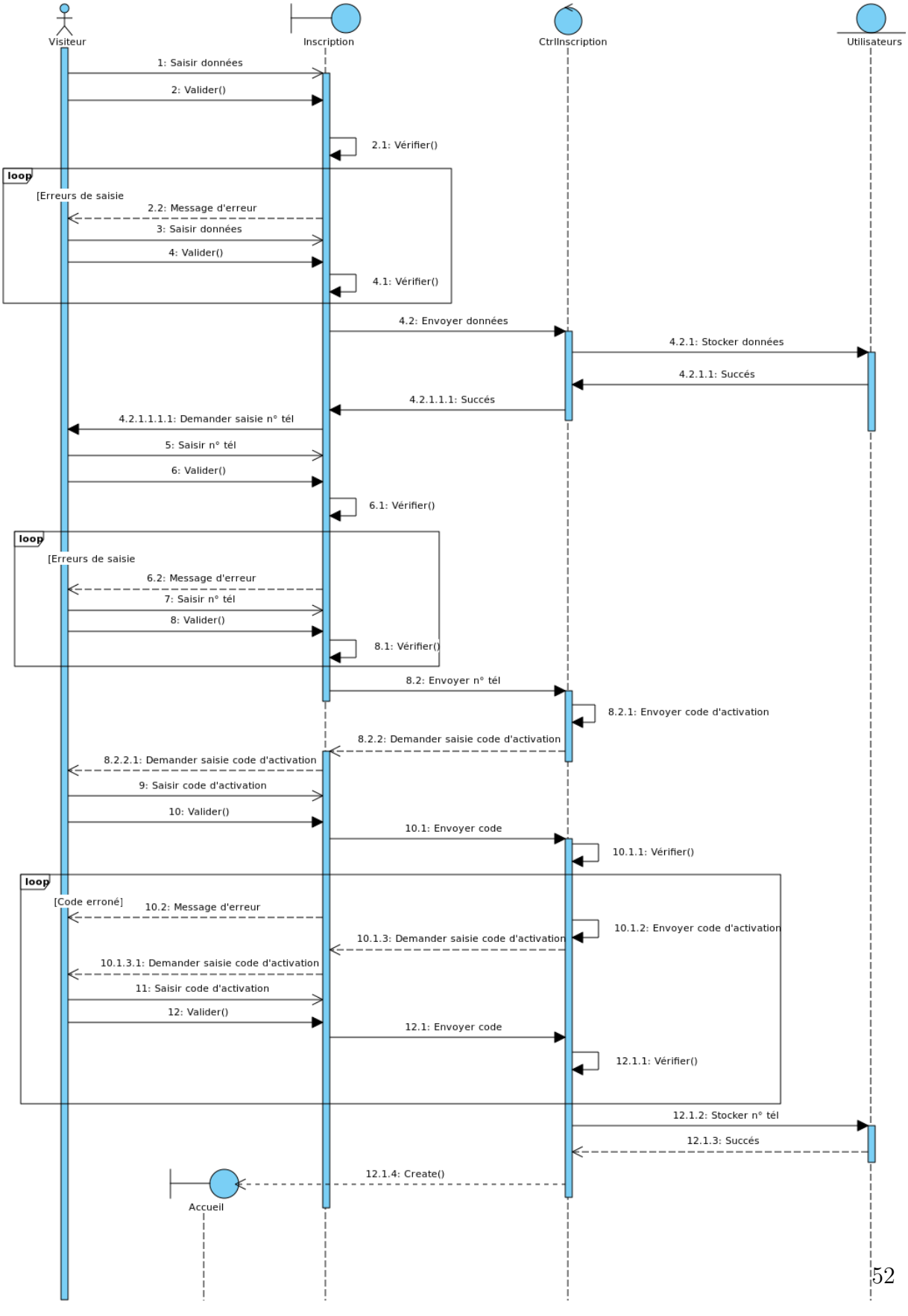


FIGURE 3.15 – Diagramme d'interaction "S'inscrire"

Diagramme d'interaction "S'authentifier"

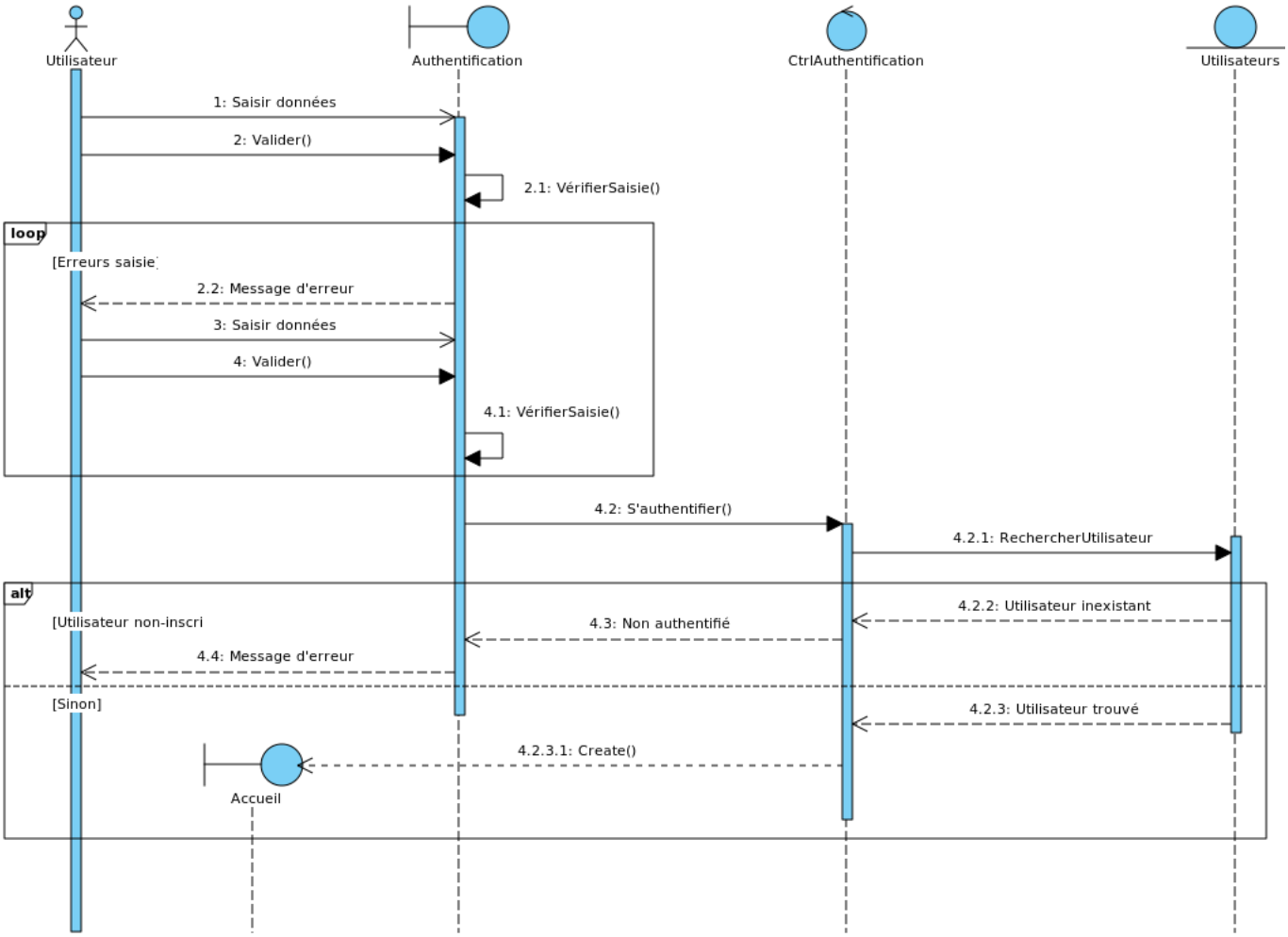


FIGURE 3.16 – Diagramme d'interaction "S'authentifier"

Diagramme d'interaction "Envoyer message"

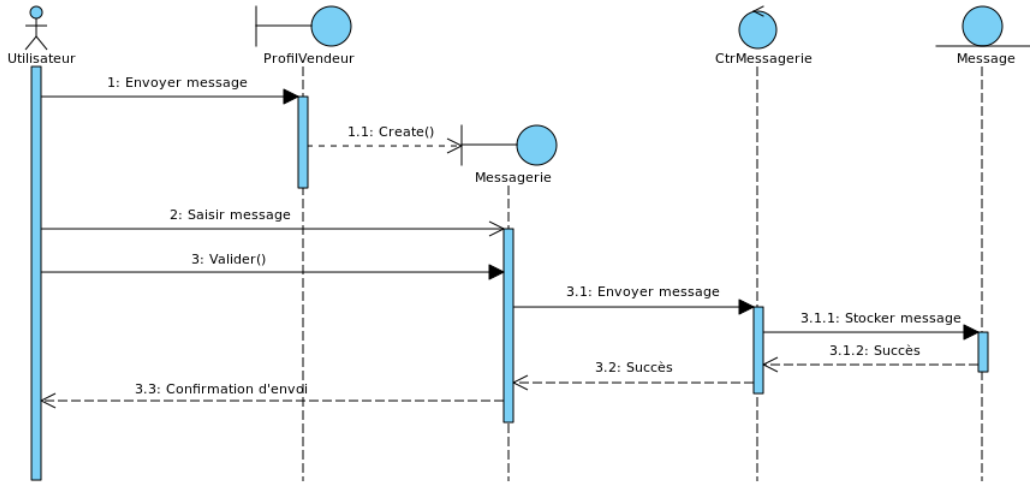


FIGURE 3.17 – Diagramme d'interaction "Envoyer message"

Diagramme d'interaction "Recevoir message"

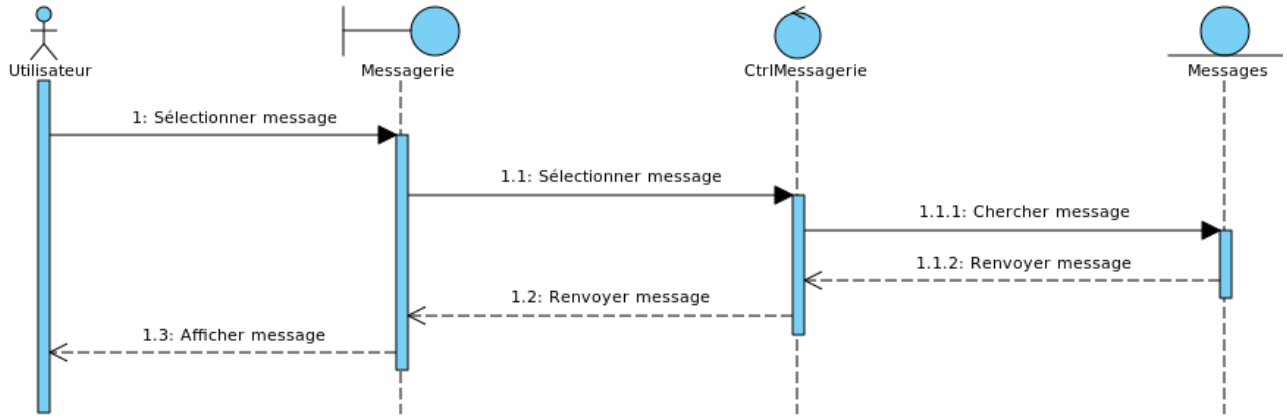


FIGURE 3.18 – Diagramme d'interaction "Recevoir message"

Diagramme d'interaction "Commander"

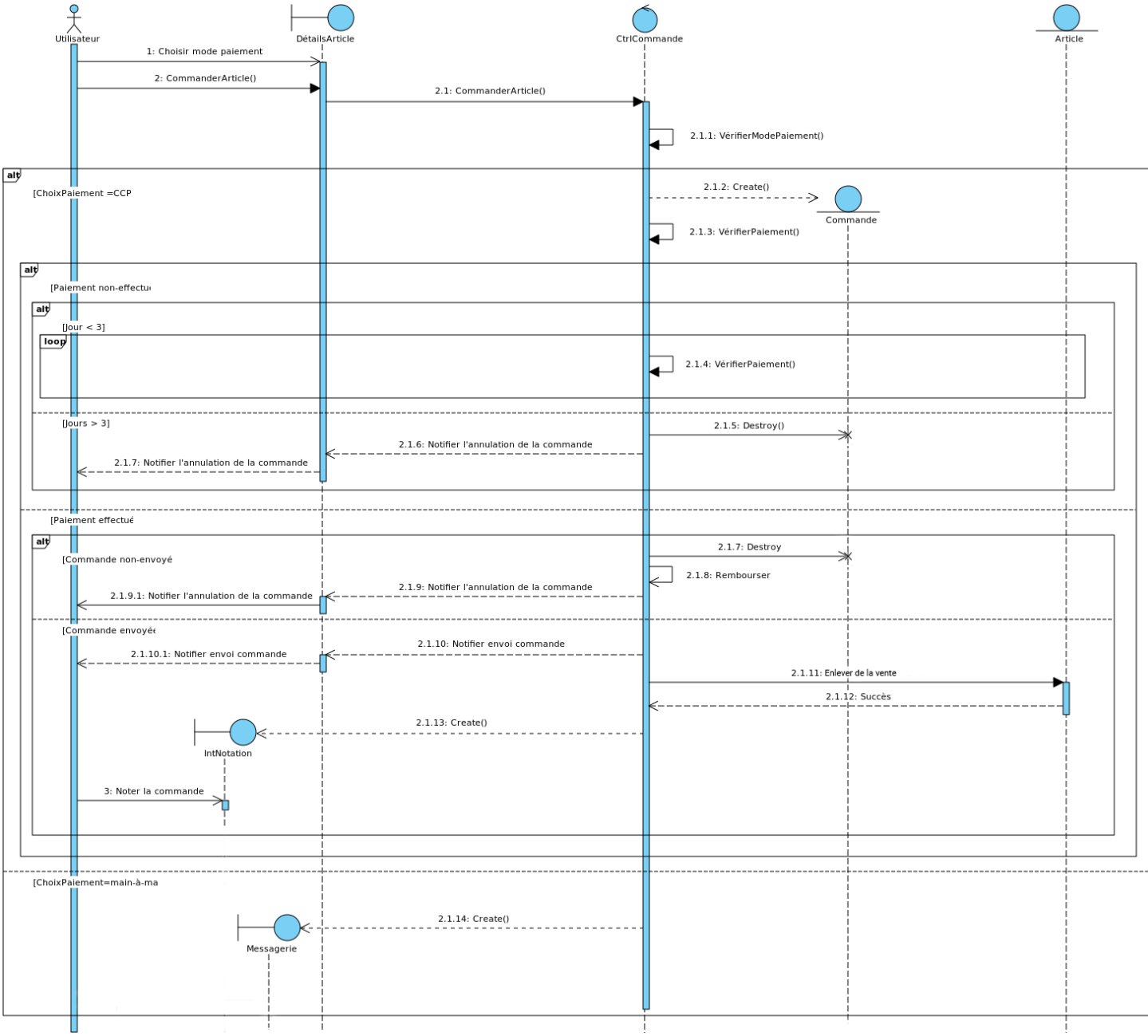


FIGURE 3.19 – Diagramme d'interaction "Commander"

Diagramme d'interaction "Recevoir commande"

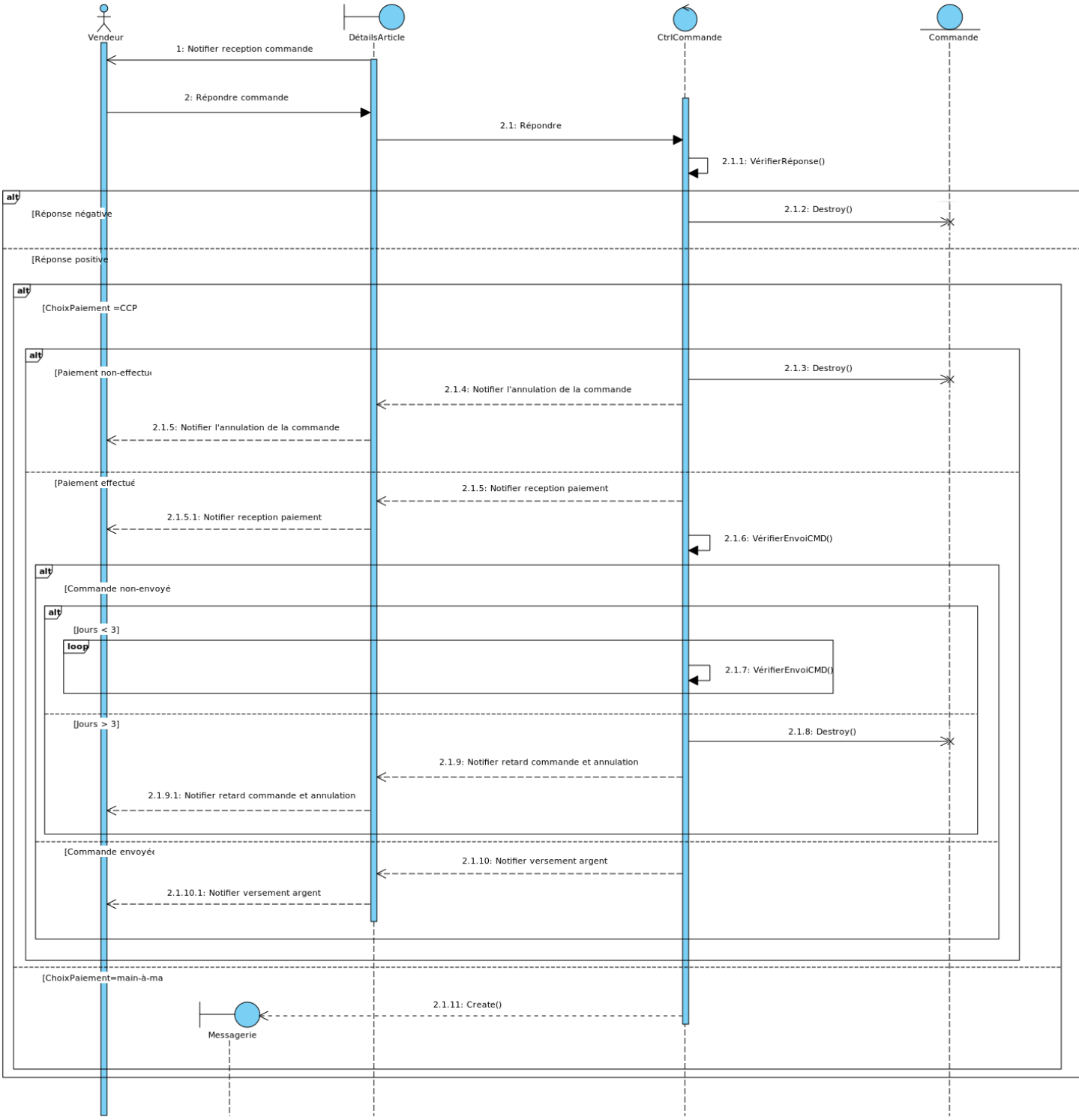


FIGURE 3.20 – Diagramme d'interaction "Recevoir commande"

Diagramme d'interaction "Ajouter article"

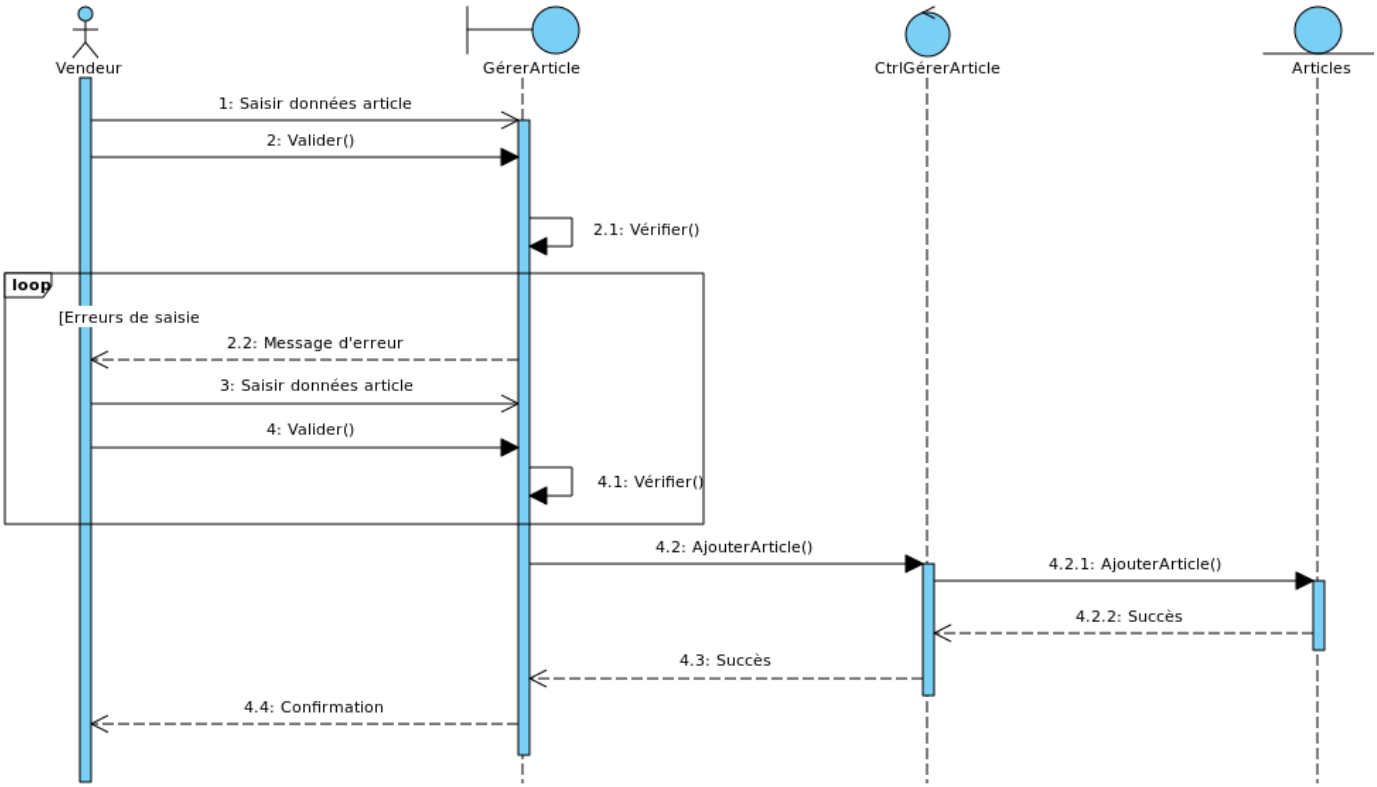


FIGURE 3.21 – Diagramme d'interaction "Ajouter article"

3.2.4 Diagramme de classes

Le diagramme de classes est un diagramme structurel indispensable, particulièrement à la création de base de données relationnelles et qui représente la structure interne du système. Les diagrammes de classes sont donc les plans du système. On peut les utiliser pour modéliser les objets qui constituent le système, pour afficher les relations entre ces objets et pour décrire les services qu'ils fournissent.

Dictionnaire des données

Pour mieux aborder les métadonnées des diagrammes de classes et simplifier la réalisation du modéle relationnel, un dictionnaire des données est idéal pour la collecte des données de référence.Il revêt une importance stratégique intéressante car il est le vocabulaire commun de l'organisation.

Classe	Code de la donné	Désignation	Type	Méthodes
Utilisateur	IdUser	Identifiant de l'utilisateur	int	SauthentifierC() TrierArticles() RechercherArticles() NoterVendeur() RecevoirMessage()
	nom	nom de l'utilisateur	String	
	prenom	Prénom de l'utilisateur	String	
	sexe	Le sexe de l'utilisateur	String	
	adresse	L'adresse de l'utilisateur	String	
	numCCP	Le numéro du compte CCP de l'utilisateur	int	
	email	L'adresse mail de l'utilisateur	String	
	motdepasse	Le mot de passe défini par l'utilisateur	String	
	numTel	Le numéro de téléphone de l'utilisateur	int	
	photoUser	La photo du profil utilisateur	image	
Commander	IdCommande	Identifiant de la commande	int	
	modePaiement	Le mode de paiement choisi par l'utilisateur	String	
	reponseVendeur	Refus/acceptation de la commande par le vendeur	Boolean	
	paiementEffectue	Paiement effectué par l'utilisateur	Boolean	
	commandeEnvoyee	Commande envoyée par le vendeur	Boolean	
	dateCommande	La date de la commande	Date	
Article	IdArticle	Identifiant de l'article	int	
	nom	nom de l'article	String	
	description	Description de l'article	String	
	Catégorie	La catégorie de l'article	String	
	etatArticle	L'état de l'article	String	
	prix	Le prix de l'article	real	
	couleur	La couleur de l'article	String	
	taille	La taille de l'article	String	
	modePaiement	Mode de paiement disponible pour l'article	String	
	photoArticle	Photo de l'article	image	
Message	IdMessage	Identifiant du message	int	
	contenuMessage	Contenu du message	String	
	destinataire	Destinateaire du message	Utilisateur	
	etatMessage	Etat du message (vu/non-vu)	String	
Envoyer	dateEnvoi	Date d'envoi du message	Date	
	heureEnvoi	Heure d'envoi du message	int	

Table 3.8 – Dictionnaire des données

Diagramme de classes

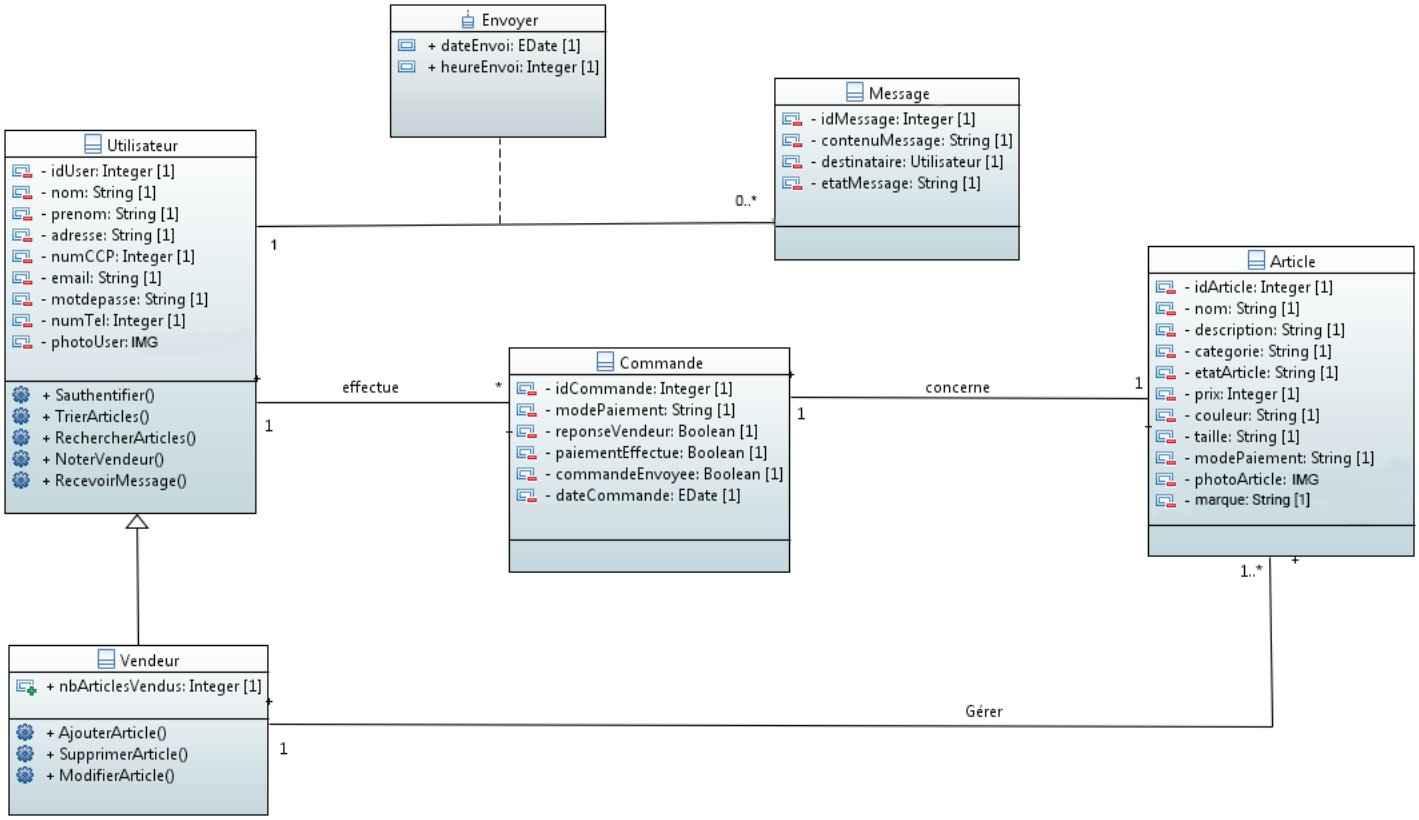


FIGURE 3.22 – Diagramme de classes

3.2.5 Le modèle relationnel

Utilisateur (idUser, nom, prenom adresse, numCCP, email, motdepasse, numTel, photoUser, nbArticlesVendus)

Message(idMessage, contenuMessage, destinataire, etatMessage, #idUser)

Envoyer(#idUser, #idMessage, dateEnvoi, heureEnvoi)

Article(idArticle, nom, description, categorie, etatArticle, prix, couleur, taille, modePaiement, photoArticle, #idUser, #idCommande)

Commande(idCommande, modePaiement, reponseVendeur, paiementEffectue, commandeEnvoyee, dateCommande, #idUser, #idArticle)

Vendeur(nbArticlesVendus)

3.3 Conclusion

Au cours de ce chapitre, nous avons présenté l'étape d'analyse, constituée de l'identification des acteurs et des diagrammes de cas d'utilisation, ce qui nous a permis de réaliser des diagrammes de séquences système associés à chaque cas d'utilisation ainsi que des diagrammes d'interaction qui nous ont aidés à décrire de manière détaillée le fonctionnement du système dans le but de faciliter la réalisation et la maintenance. Nous avons, par la suite, conçu le diagramme de classes pour illustrer globalement la structure des éléments constituant la base de données associée à notre application. Dans le chapitre suivant, nous entamerons l'étape de réalisation.

Chapitre 4

Réalisation

4.1 Introduction

Ce chapitre est consacré à l'étape de réalisation et de mise-en-œuvre de notre application. Nous y présentons les différents outils adoptés tout au long du projet, soit les outils de conception du logo, des différents diagrammes, et des maquettes, ainsi que l'environnement de développement, les langages de programmation, le système de gestion de bases de données avec le langage de manipulation de bases de données.

4.2 Outils de conception UML

Eclipse Papyrus

Eclipse Papyrus est un logiciel de modélisation UML, multiplateformes gratuit et open-source développé par la fondation Eclipse. Il a été utilisé pour la conception des diagrammes de cas d'utilisation. La version utilisée est la dernière 4.7.0

FIGURE 4.1 – Logo Eclipse Papyrus.

Visual Paradigm

Visual Paradigm est un logiciel qui contient plusieurs outils de modélisation UML conçu par OGM, il sera utilisé pour la conception des diagramme de séquence système, d'interaction ainsi que du diagramme de classes. On utilisera la version 16.1 en Free Trial de 14 jours.

FIGURE 4.2 – Logo Visual Paradigm.

4.3 Outils de conception graphique

Adobe Illustrator CC

Adobe Illustrator est un logiciel d'illustration vectorielle qui fait partie de la gamme Adobe qui offre des outils de dessins vectoriels puissants, très adaptés pour la création de logo. Adobe Illustrator CC17.

FIGURE 4.3 – Logo Adobe Illustrator.

Adobe Photoshop CC

Adobe Photoshop est un logiciel de traitement d'images édité par Adobe. Idéal pour le traitement d'images numériques. Adobe Photoshop CC17.

FIGURE 4.4 – Logo Adobe Photoshop.

Figma

Figma est un outil de design d'interface et de prototypage entièrement sur Navigateur. Il a pour avantage d'être collaboratif en temps réel, et aussi de prototyper toutes les interactions entre interface, ce qui permet de montrer facilement le parcours d'un utilisateur de manière réaliste avant que l'application ne soit développée réalisant ainsi un option tree flowchart en temps réel. Dernière mise à jour (09/06/2020)

FIGURE 4.5 – Logo Figma.

Whimsical

Whimsical est un outil de wireframing rapide collaboratif en temps réel. Il a l'avantage, en plus d'être collaboratif en temps réel, d'avoir des designs pour le wireframing préconçus, ce qui rend le processus de création rapide.

FIGURE 4.6 – Logo Whimsical.

4.4 Outils de développement

Visual Studio Code

Visual Studio Code est un éditeur de code multiplateforme édité par Microsoft. Il supporte plusieurs dizaines de langages de programmation et intègre plusieurs outils facilitant la saisie de code et son débogage. En outre, l'outil permet d'installer plusieurs plugins, notamment ceux de **Flutter** et **Dart**. Dernière version Mai 2020 (1.46. 1)

FIGURE 4.7 – Logo Visual Studio Code.

Flutter et DART

Flutter est un framework de Google permettant de concevoir des applications multiplateforme pour Android et iOS, il s'appuie sur le langage de programmation DART, qui est également de Google. Flutter met à disposition ses propres Widgets qui représentent des éléments graphiques, ce qui rend la conception des interfaces graphique considérablement simplifiée. En outre, le build des applications est très rapide, le temps de compilation est largement réduit, et ce grâce à la fonctionnalité Hot Reload de DART. dernière version stable de Flutter 1.17 et 2.8.0 pour Dart (Juin 2020)

FIGURE 4.8 – Logo Flutter et Logo Dart.

Installation et configuration

Pour installer et exécuter Flutter, l'environnement de développement doit répondre à des exigences minimales :

— **Système d'exploitation :** Windows 7 SP1 ou plus récent (64-bit).
— **Espace disque :** 400 MB (sans inclure l'espace pour l'IDE/tools).
— **Outils :**
Windows PowerShell 5.0 ou plus récent (pré-installé sur Windows 10).
Git pour Windows 2.x, l'option d'utilisation de Git pour l'invite de commande Windows.

Si ces conditions sont satisfaites, télécharger le SDK Flutter, et extraire l'archive dans un dossier. Ajouter le chemin vers Flutter dans la variable d'environnement PATH pour pouvoir utiliser les commandes Flutter dans la console Windows. Vérifier par la suite à l'aide de la commande "flutter doctor" qui vérifie l'environnement et affiche un rapport avec le statut de l'installation de Flutter. Ajouter aussi l'emplacement du SDK Android à la variable d'environnement ANDROIDHOME et PATH.

Installer par la suite les extensions Flutter et Dart sur Visual Studio Code (ou Android Studio), et enfin créer un projet Flutter.

XAMPP

Pour implémenter notre base de données, nous avons utilisé l'environnement de création de base de données PHPMyAdmin et le système de gestion de base de donnés MySQL sous la plate-forme XAMPP qui est une distribution Apache. Version PHP 7.2.1 et 4.7.7 PHPMyAdmin

FIGURE 4.9 – Logo XAMPP et outils associés.

GitHub

Git hub est une sorte de cloud pour le code. Il héberge le code source dans plusieurs langages de programmation et garde toute version ou changement que le code subit, en utilisant le git. L'utilisation de GitHub rend aussi la collaboration plus facile et enfin, son service est open-source.

FIGURE 4.10 – Logo GitHub

4.5 Conclusion

Ce dernier chapitre représente la conclusion de notre projet. Dans ce chapitre, nous avons décrit brièvement le processus de réalisation de notre application en spécifiant l'environnement, les outils et les langages de développement associés à notre système.

Conclusion générale

Pour conclure, ce travail, réalisé dans le cadre du projet de fin d'études, en vue de l'obtention du diplôme de master professionnel en Génie Logiciel, est une mise à disposition d'un outil web mobile d'e-commerce pour la population algérienne qui aide à faciliter les échanges urbains et inter-wilaya, et ce grâce à la fonction de commande, la possibilité de payement par CCP et la messagerie instantanée.

La commande fut la fonctionnalité la plus compliquée à concevoir pour arriver à contourner certaines tentatives de fraudes et réaliser un certain degré de sécurité pour l'utilisateur (numéros du comptes CCP, argent, numéros de téléphone, etc.).

Une amélioration que nous verrions appropriée à ce projet serait éventuellement l'adaptation de l'application sur la plateforme iOS, chose facilitée grâce aux outils Flutter. Aussi, une incorporation plus approfondie de l'Intelligence Artificielle, surtout dans les recommandations, serait bénéfique et permettra de donner un contenu plus adapté à chaque utilisateur.

Nous retenons de ce projet l'utilité des méthodes de conception et outils modernes dans la facilitation et accélération des travaux et l'organisation des groupes de façon efficace, pour ainsi donner un produit professionnel et moderne. D'un autre côté, nous avons appris que nous pouvons négliger combien un projet informatique comme celui-ci demande de temps et de ressources pour être accompli.

Bibliographie

[1] *https ://www.cia.gov/library/publications/resources/the-world-factbook/geos/ag.html.*[*31 dec. 2019*]. Vu le 22/05/2020

[2] *https ://econsultancy.com/android-doubles-market-share-to-take-52-5-of-smartphone-market/.* [*15 nov. 2017*]. Vu en 06/2020

[3] `https://www.getstark.co/`

[4] When to use automated regression testing *Dayana Mayfield* [*25 nov. 2019*].

[5] *https ://whimsical.com/a*

[6] `https://undraw.co/license/`

[7] *The Manifest https ://medium.com/@the*$_m$*anifest*/$6-user-onboarding-flows-for-mobile-apps-ef974a330ac1$[*1 mai. 2018*].$Vuen$03/2020

[8] User Interface Design Guidelines : 10 Rules of Thumb (Nielsen Mulich) *Euphemia Wong*
[*juin 2020*].

[9] Past, present, and future of user interface software tools *ACM.* [*2000-03-01. Retrieved 2009-04-02.*].

[10] 6 Tips for Designing an Optimal User Interface for Your Digital Event *Wolf, Lauren* [*23 Mai 2012*].

[11] *https ://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/*

[12] LE guide SCRUM, Le guide définitif de Scrum : les règles du jeu *Développé et maintenu par Ken Schwaber et Jeff Sutherland* [*Juillet 2013*].

[13] *https ://www.unow.fr/blog/le-coin-des-experts/product-backlog-projet-scrum/ Pierre Monclos* [*17 oct 2016*].

[14] *https ://www.w3.org/WAI/redesign/ucd* [*Version : 2004.03.31*]. Vu en 03/2020

[15] Da Silva, Tiago Martin, Angela Maurer, Frank Silveira, Milene. (2011). User-Centered Design and Agile Methods : A Systematic Review. *Proceedings - 2011 Agile Conference, Agile 2011.* 77 - 86. 10.1109/AGILE.2011.24.

[16] *https ://www.peerbits.com/blog/mobile-app-security-coding-practices.html Mohsin Shahjada* Vu en 03/2020

[17] Christoph Schwindt ;Jürgen Zimmermann (2015). *Handbook on Project Management and Scheduling*. Swiss : Spring. pages 910 - 943.

[18] DON NORMAN (2013). *THE DESIGN OF EVERYDAY THINGS*. New York : Basic Books. 331 pages.

Résumé

Aujourd'hui, de nombreuses personnes se plaignent du surplus d'objets qui encombrent leurs armoires sans utilité aucune. En Algérie, il n'existe que peu de plateformes dédiées à tourner profits de ces objets. L'application mobile Gooldies, qui a inspiré ce mémoire, met à la disposition de ses utilisateurs un moyen de se débarasser, de manière économique, des biens dont ils ne se servent plus. Pour cela, n'importe quel utilisateur inscrit à l'application a la possibilité de publier un article et de le mettre en vente, ou en acheter à des prix abordables. L'application comprend, en plus d'une messagerie instantanée, un système de commande sécurisé avec un choix de mode de livraison/payement. L'application a été, en premier lieu, conçue graphiquement grâce à l'outil Figma qui prodigue des interfaces interactives suivant les normes UX/UI. Sa conception UML, a été faite avec le plugin Papyrus d'Eclipse pour les diagrammes de cas d'utilisation et Visual Paradigm pour le reste des diagrammes. En dernier lieu, le codage a été réalisé avec le Framework Flutter de Google accompagné du langage Dart et ce avec l'IDE Visual Studio Code de Microsoft. La méthode de conception agile SCRUM, accompagnée , du Processus centré utilisateur UCD, a été utilisée pour gérer de manière optimale la réalisation de ce projet à travers les planning et le Product Backlog.

Mots-clés : Gooldies, UX, UI, UML, Flutter, Dart, SCRUM, UCD, Product Backlog.

Abstract

Today, many people complain about the surplus of objects which clutter their closets without any use. In Algeria there are only a few platforms dedicated to turning profits from these objects. Gooldies mobile application, which inspired this project, provides its users with a way to economically dispose of goods they no longer use. Therefore, any user registered within the application has the possibility to publish an article and put it on sale, or buy it at affordable prices. The application includes, in addition to instant messaging, a secure ordering system with a choice of delivery / payment mode. The application was, at first, designed graphically thanks to Figma which provides interactive interfaces according to UX / UI standards. Its UML design was done with Eclipse's Papyrus plugin for use case diagrams and Visual Paradigm for the rest of the diagrams. Lastly, the coding was done with the framework Flutter from Google accompanied by Dart language using Visual Studio Code IDE from Microsoft. The agile design method SCRUM, accompanied by the User-Centered Process UCD, was used to optimally manage the realization of this project through the planning and the Product Backlog.

Keywords : Gooldies, UX, UI, UML, Flutter, Dart, SCRUM, UCD, Product Backlog.