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

Option : Génie Logiciel

Thème

Conception et Réalisation d'un système de livraisons avec tracking GPS

Devant le jury composé de :

Examinateur : Mr SAADI Mustapha

Examinateur : Mr NAFI Mohamed

Encadrant : Dr ACHROUFENE Achour

Réalisé par :

Mr. MAHIOUT Youba

2020/2021

Remerciements

Mes remerciements s'adressent à mon encadrant Monsieur ACHROUFENE Achour, pour avoir accepté de diriger ce travail. Son soutien, sa clairvoyance, ses compétences, ainsi que son infinie disponibilité nous ont été d'une aide inestimable.

Je remercie également M. DJERADA Lyes pour sa disponibilité, sa gentillesse, et sa précieuse directive tout au long de la réalisation de ce travail.

Qu'ils puissent trouver dans ce travail le témoignage de ma sincère gratitude et de mon profond respect.

Je tiens également à remercier sincèrement les membres du jury qui me font l'honneur d'évaluer ce travail.

Je remercie également ma famille et amis pour leur soutien permanent qui m'a été bien utile.

Je remercie tout particulièrement Monsieur MAOUCHI Mohamed Djamil pour m'avoir consacré de son temps, afin de m'aider dans la réalisation du module GPS.

Dans l'impossibilité de citer tous les noms, que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail trouvent ici l'expression de ma sincère gratitude.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

API : Application Programming Interface
CSS : Cascading Style Sheets
DOM : Document Object Model
GPS : Global Positioning System
GSM : Global System for Mobile Communication
HTML : Hyper Text Markup Language
HTTP : Hyper Text Transfer Protocol
IDE : Integrated Development Environment
JIT : Just In Time
JS : Javascript
LED : light emitting diode
NoSQL : Not Only Structured Query Language
OS : Operating System
PDF: Portable Document Format
PWA : Progressive Web App
REST : REpresentational State Transfer
SEO : Search Engine Optimization
SQL : Structured Query Language
UI : User Interface
UML : Unified Modeling Language
UP : Unified Process
URL : Uniform Resource Locator
XP : eXtreme Programing

Introduction générale

Depuis quelques dizaines d'années nous assistons à une révolution des méthodes de travail et des façons de faire principalement liée aux nouvelles technologies. Aujourd'hui, en tant que particulier ou professionnel, nous sommes tous sujets à l'utilisation des nouvelles technologies. Elles ont pris une place indispensable dans nos vies et dans le fonctionnement de l'entreprise notamment.

En effet, avec les anciennes méthodes de gestion, les entreprises de livraisons gèrent les colis et leurs livraisons via le format papier ou sur Excel, et pour le suivi de ces livraisons et colis ils possèdent un service de communication qui assure la relation entre l'entreprise et le client mais ceci peut devenir rapidement débordant quand il s'agit de plusieurs clients.

Afin de bénéficier des avantages des nouvelles technologies, ce projet vise à développer un système qui facilite la gestion des colis de la réception jusqu'à l'expédition vers le client final en offrant à l'entreprise et aux clients un outil qui permet un suivi permanent du colis.

Ce mémoire est structuré en quatre chapitres, le premier chapitre qui s'intitule Contexte du projet et méthodologie de développement, nous permettra d'encadrer le contexte du projet et d'exposer la problématique à résoudre, ensuite nous présenterons les applications Web, mobile et les GPS, ainsi que le processus de développement d'entreprise qui nous facilitera l'élaboration du projet.

Le deuxième chapitre sera consacré à la spécification et l'analyse des besoins, qui consiste à identifier les différents acteurs et décrire les besoins fonctionnels et non fonctionnels.

Le troisième chapitre sera dédié à la conception de notre application, qui aura pour objectif de détailler la description du système d'un point de vue technique.

La réalisation du l'application fera l'objet du dernier chapitre qui sera décomposé en deux sections, dans la première section nous aborderons l'aspect matériel ou nous allons présenter les différents composants qui une fois assemblés forment notre module GPS. Dans la deuxième section nous présentons les outils de développement, les langages de programmation, les librairies utilisés et les logiciels de prototypages, puis nous allons présenter l'application à l'aide de quelques interfaces.

Enfin, nous conclurons ce travail en résumant les connaissances acquises durant ce projet et nous exposerons quelques perspectives.

Chapitre I : Contexte du projet et méthodologie de conception

I.1. Introduction

Dans ce premier chapitre, nous exposerons le contexte du projet et la problématique à résoudre. Ensuite, nous aborderons quelques définitions sur les applications Web, mobile et les GPS. Enfin nous définirons le processus de développement entrepris afin de faciliter l'élaboration du projet.

I.2. Etude de l'existant

Beaucoup de sociétés de livraisons comme (Quick Livraison[1], Wasel[2], FastDelivery Dz[3]) assurent le service de livraison au territoire national. Elles reposent principalement sur quatre services qui assurent leur fonctionnement normal, qui sont : le service gestion comptable, le service ressources humaines, le service de livraisons à domicile et le service de communication.

En effet, la gestion comptable est gérée par un expert-comptable qui est soit employé dans l'entreprise ou indépendant, ce dernier utilise le logiciel Sage [1]. Cette plateforme de gestion assure l'enregistrement de toutes les opérations comptables des sociétés.

Le service de ressources humaines est dédié à la gestion du personnel de ces entreprises. Son travail repose sur la création du dossier papier pour chaque employé ou stagiaire. Ce dossier contient le contrat de travail, les pièces d'identité et le bulletin de paie. Ce service s'occupe également du calcul des heures de présence de chaque employé.

Le service de livraisons est supervisé par le chef d'entrepôt qui est le gérant de l'entrepôt de l'entreprise. Il gère les colis de leur arrivée à l'entrepôt jusqu'à la réception par le client final. Son travail est donc la réception des colis des marchands, la création de dossier papier pour chaque livraison, la création des étiquettes pour l'identification de chaque colis et le regroupement des colis pour avoir des groupes de livraisons prêts pour les livreurs qui sont les chauffeurs de ces entreprises et qui effectuent les livraisons. Ils se rendent alors chaque matin vers le chef d'entrepôt pour récupérer la fiche de la mission du jour afin d'effectuer la livraison.

Enfin, le service de communication qui est géré par les employés de l'entreprise et leur travail est de toujours mettre l'entreprise à l'écoute de ses clients et de donner l'état de leurs colis et ceci par les appels téléphoniques et des messages.

[1] https://quicklivraison.com
[2] https://wasel.dz/home
[3] https://www.fastdelivery.dz

I.3. Problématique et Solutions proposés

Pour gérer l'activité globale de la société, chaque département utilise indépendamment des applications informatiques ou autres méthodes. Ceci génère non seulement une difficulté d'échange de données entre les différents départements, mais également une incohérence de l'information interne. À travers cela on soulève plusieurs problèmes :

- Les informations des colis sont saisies et imprimées à la main.
- Suivi de colis difficile voire impossible.
- Débordement sur le service de communication par rapport aux appels pour le suivi.
- Certaines sources de données sont stockées sous forme de fichiers Excel ou sur papier.

Pour y remédier nous supposons la création d'une société qui va répondre à ces problématiques, nous la nommons « Delivo ».

I.3.1. Delivo

Delivo est une société fictive qui a pour but de créer des solutions logistiques plus pertinentes par rapport aux enjeux actuels de la société, et ce afin de faciliter aux commerçants et leurs clients le suivi de leurs colis en toute transparence et faciliter. La *figure I-1* représente un schéma indiquant le fonctionnement de cette société.

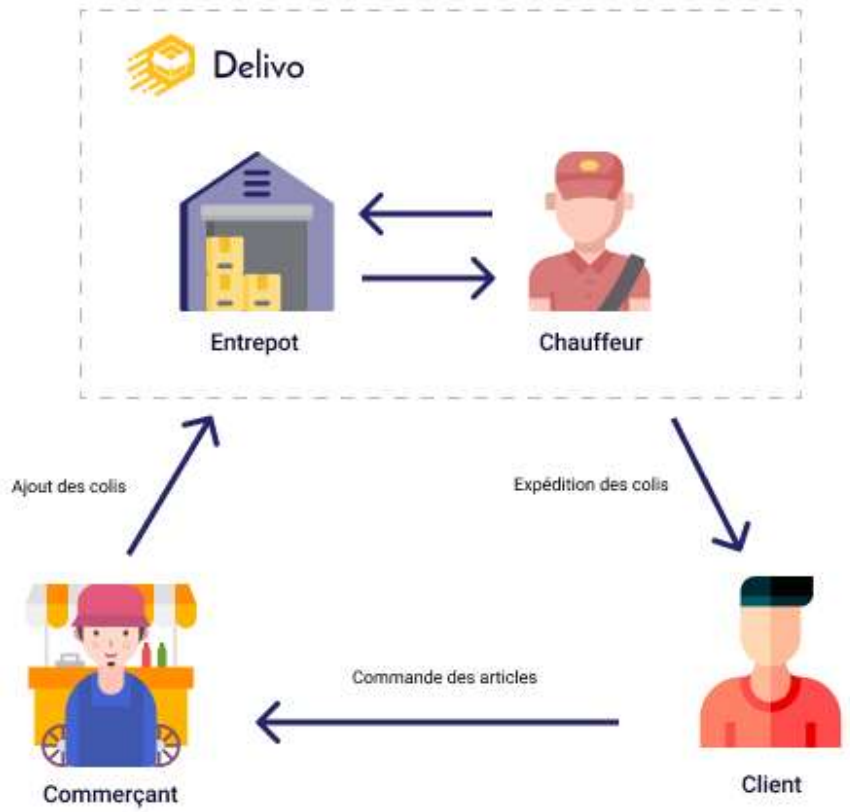

Figure I-1 - Schéma explicative de Delivo

I.3.2. Solutions

La solution que Delivo propose est un système de gestion des colis et des chauffeurs, ainsi qu'un moyen de les retracer en temps réel. Nous allons alors créer une application web qui sera à la fois ergonomique et conviviale, qui facilite la gestion des colis de la réception de ces derniers de la part des commerçants jusqu'à leur réception par le client final, et qui automatise l'interaction entre la société et les commerçants. Ensuite nous créerons une application mobile pour permettre aux livreurs d'avoir accès aux itinéraires de leurs livraisons, ainsi qu'avoir accès aux informations des colis qu'ils transportent en scannant simplement le QR Code qui sera imprimé dessus. Enfin nous allons créer un module GPS à l'aide de quelques composants électroniques afin d'avoir des suivis de colis en temps réel sur l'application web.

I.4. Applications Web

En informatique, une application Web (aussi appelée site Web Dynamique ou WebApp) est un logiciel applicatif manipulable grâce à un navigateur Web. De la même manière que les sites web, une application Web est généralement placée sur un serveur et se manipule en actionnant des widgets à l'aide d'un navigateur.

Une application Web est en général basée sur HTML, JavaScript ou CSS. Étant donné qu'une telle application est chargée depuis un serveur et exécutée via un navigateur, il n'y a donc aucune installation à prendre en charge. Néanmoins l'utilisation d'un signet ou raccourci permet l'accès direct à l'application depuis le bureau ou l'écran d'accueil d'un appareil mobile [2].

I.4.1. Pourquoi une application web

Au cours des dernières années, avec l'amélioration de la sécurité et une technologie de plus en plus flexible, les applications Web se sont développées rapidement. Presque toutes les applications natives peuvent être développées en tant que applications Web et profiter des nombreux avantages offerts par le Web [3] :

- Un accès rapide via divers navigateurs.
- Fonctionne sur tous les systèmes d'exploitation.
- Accessible de partout.
- Travail en simultané dans le « Cloud ».
- Sécurité des contenus.

I.5. Applications Mobiles

Une application mobile est un logiciel applicatif téléchargeable (gratuitement, payante ou mixte avec des contenus ou fonctionnalités payantes) sur un appareil mobile (baladeur, smartphone, tablette tactile) via une plateforme de téléchargement adaptée au système d'exploitation (Google Play, App Store, Windows Store) [4].

I.5.1. Pourquoi une application mobile

D'un côté marketing, lorsque les utilisateurs installent volontairement une application sur leurs smartphones, ils ont déjà montré leur engagement envers le produit. De plus, il y a possibilité de leur envoyer des notifications pour faciliter la communication et attirer leur attention. Les applications mobiles sont répertoriées dans les deux principaux stores. Le moteur de recherche Google référence également des applications mobiles, ce qui représente un avantage supplémentaire pour augmenter sa visibilité. Les applications mobiles sont un moyen de se distinguer de la concurrence, et elles donneront l'image d'une entreprise innovante.

Quant au côté ergonomie, une application mobile permet d'accéder facilement et rapidement à plusieurs interactions autrefois impossibles via des interfaces travaillées pour une simplicité d'utilisation optimale.

I.5.2. Types d'applications mobiles

Il existe principalement trois types d'applications mobiles [5], leur principale différence réside dont la façon avec laquelle elles s'exécutent, ainsi que la méthode de leur écriture. Ces trois types sont :

- **Application native :**

Une application mobile native est une application pour smartphone codée dans un langage de programmation spécifique, tel que Objective C pour iOS ou Java pour les systèmes d'exploitation Android. Les applications mobiles natives offrent des performances rapides et un haut degré de fiabilité. Elles ont également accès aux différents dispositifs d'un téléphone, tels que son appareil photo et son carnet d'adresses. En outre, les utilisateurs peuvent utiliser certaines applications sans connexion Internet. Cependant, ce type d'application est coûteux à développer car il est lié à un type de système d'exploitation, ce qui oblige l'entreprise qui crée l'application à produire des versions dupliquées qui fonctionnent sur d'autres plateformes.

- **Applications cross-platform :**

Ce sont des applications mobiles développées pour fonctionner sur plusieurs plateformes mobiles. Ces applications sont compatibles avec plus d'un système d'exploitation, comme iOS et Android. Avec le développement d'applications mobiles multiplateformes, les développeurs peuvent créer des applications qui peuvent fonctionner sur différentes plateformes avec un seul système de code. Cela signifie que l'entreprise peut publier le produit plus rapidement et avec une meilleure qualité. Puisqu'elle est compatible avec divers systèmes d'exploitation mobiles, l'application peut atteindre un public plus large.

- **Progressive web app :**

Une application web progressive (PWA) est un site web qui se présente et se comporte comme une application mobile. Les PWA sont conçues pour tirer parti des fonctionnalités natives des appareils mobiles, sans que l'utilisateur final ait à se rendre dans une boutique d'applications, à effectuer un achat et à télécharger un logiciel localement. Au lieu de cela,

une PWA peut être localisée par une requête sur un moteur de recherche et accessible immédiatement via un navigateur. Cela élimine le besoin de devoir créer une application mobile à part entière.

I.6. Suivi GPS (Tracking)

Le GPS ou Global Positioning System se définit littéralement par « Système mondial de positionnement ». Sous ce sigle se cache un système sophistiqué de localisation par satellites. Ce système de géolocalisation est aujourd'hui utilisé dans le monde entier [6].

I.6.1. À quoi sert et comment fonctionne un GPS ?

Le GPS est souvent portable et renseigne son possesseur sur la localisation géographique. Celle-ci envoie un signal à un satellite, qui le renvoie au récepteur mais sous la forme latitude longitude cela nous permet d'identifier la position de porteur du GPS sur une carte.

I.6.2. Pourquoi utiliser un GPS ?

D'ordre financier, ce système de suivi offre à l'entreprise d'économiser, notamment dans la consommation de carburant. Cela dit, le GPS installé dans ses véhicules permet à l'entreprise de chiffrer les dépenses en carburant engagé lors d'une livraison. Le but n'est pas seulement de surveiller les comportements de travail du chauffeur, pour le bien de l'entreprise, mais aussi et aussi de pouvoir suivre l'état de livraisons des colis et donner l'opportunité aux clients de suivre en direct leur colis, et de pouvoir améliorer la qualité de du service et d'économiser un maximum les charges.

I.7. Processus de développement

Un processus définit une séquence d'étapes, partiellement ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. L'objectif d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles.

Nous avons choisi de travailler avec le Processus Unifié (UP) combiné avec la méthode agile Extreme Programming (XP) cela n'est pas une obligation mais un choix car elle nous paraît simple avec le nombre réduit dans notre équipe. En effet, le processus unifié est une méthode de développement logiciel orienté objet adaptée à tout type de projet et XP est une méthode agile orientée pour l'aspect de réalisation sans pour autant négliger l'aspect gestion de projet, elle est adaptée aux équipes réduites avec des besoins changeants, et pour la modélisation on a opté pour l'UML qui permet de modéliser les différentes facettes du système durant le développement.

I.7.1. Processus unifié (Unified Process)

Le processus unifié est basé sur l'élargissement et le raffinement d'un système par le biais de multiples itérations, avec une rétroaction et une adaptation cyclique. Le système est

développé de manière incrémentale au fil du temps, itération par itération, et cette approche est donc également connue sous le nom de développement logiciel itératif et incrémental. Les itérations sont réparties sur quatre phases, chacune d'entre elles comprenant une ou plusieurs itérations [7] :

- **Inception :** la première et la plus courte des phases du projet. Elle est utilisée pour préparer les bases du projet, y compris la préparation de l'analyse de rentabilisation, l'établissement de la portée du projet et la définition des limites, la description des principales exigences et la solution d'architecture possible ainsi que les compromis de conception, l'identification des risques et le développement du plan de projet initial.
- **Élaboration :** au cours de cette phase, l'équipe de projet doit saisir la majorité des exigences du système (par exemple, sous la forme de cas d'utilisation), effectuer une analyse des risques identifiés et élaborer un plan de gestion des risques afin de réduire ou d'éliminer leur impact sur le calendrier et le produit final, établir la conception et l'architecture (par exemple, en utilisant des diagrammes de classe de base, des diagrammes de paquetage et des diagrammes de déploiement), créer un plan (calendrier, estimations de coûts et étapes réalisables) pour la phase suivante (construction).
- **Construction :** la phase la plus longue et la plus importante du processus unifié. Pendant cette phase, la conception du système est finalisée et affinée et le système est construit en utilisant la base créée pendant la phase d'élaboration. La phase de construction est divisée en plusieurs itérations, chaque itération devant aboutir à une version exécutable du système. L'itération finale de la phase de construction libère un système entièrement terminé qui sera déployé pendant la phase de transition.
- **Transition :** la phase finale du projet qui fournit le nouveau système à ses utilisateurs finaux. La phase de transition comprend également la migration des données des anciens systèmes et la formation des utilisateurs.

Chaque phase et son itération consistent en un ensemble d'activités prédéfinies. Le processus unifié décrit les activités de travail comme des disciplines :

- **Modélisation du business :** modélisation des objets du domaine et modélisation dynamique des processus du business.
- **Exigences :** analyse des exigences du système considéré. Cela comprend des activités telles que la rédaction de cas d'utilisation et l'identification des exigences non fonctionnelles.
- **Analyse et conception :** couvre les aspects de la conception, y compris l'architecture globale.
- **Mise en œuvre :** programmation et construction du système (sauf le déploiement).
- **Test :** comprend les activités de test telles que la planification des tests, le développement de scénarios de test, les tests alpha et bêta, les tests de régression, les tests d'acceptation et le déploiement.
- **Déploiement :** les activités de déploiement du système développé.

I.7.2. XP (eXtreme Programming)

Extreme Programming (XP) est un cadre de développement logiciel agile qui vise à produire des logiciels de meilleure qualité, et une meilleure qualité de vie pour l'équipe de développement. XP est le plus spécifique des cadres agiles en ce qui concerne les pratiques d'ingénierie appropriées pour le développement de logiciels [8].

Les cinq valeurs d'XP sont la communication, la simplicité, le retour d'information, le courage et le respect et sont décrites plus en détail ci-dessous :

- **Communication :** Le développement de logiciels est par nature un sport d'équipe qui repose sur la communication pour transférer les connaissances d'un membre de l'équipe à tous les autres. XP insiste sur l'importance du type de communication approprié - discussion en face à face à l'aide d'un tableau blanc ou d'un autre mécanisme de dessin.
- **Simplicité :** La simplicité signifie "quelle est la chose la plus simple qui va fonctionner ?". Il s'agit d'éviter le gaspillage et de ne faire que les choses absolument nécessaires, par exemple en gardant la conception du système aussi simple que possible afin qu'il soit plus facile à maintenir, à soutenir et à réviser. La simplicité signifie également qu'il faut répondre uniquement aux exigences que l'on connaît, il faut éviter d'essayer de prédire l'avenir.
- **Retour d'information :** Grâce à un retour constant sur leurs efforts précédents, les équipes peuvent identifier les domaines à améliorer et revoir leurs pratiques. Le retour d'information favorise également une conception simple.
- **Courage :** Kent Beck a défini le courage comme "une action efficace face à la peur" [9]. Cette définition montre une préférence pour l'action basée sur d'autres principes afin que les résultats ne soient pas nuisibles à l'équipe. Il faut du courage pour soulever les problèmes organisationnels qui réduisent l'efficacité de votre équipe. Il faut du courage pour arrêter de faire quelque chose qui ne fonctionne pas et essayer autre chose.
- **Respect :** Les membres de l'équipe doivent se respecter les uns les autres afin de communiquer entre eux, de fournir et d'accepter un retour d'information qui honore leur relation, et de travailler ensemble pour identifier des conceptions et des solutions simples.

I.7.3. UML (Unified Modeling Language)

Le langage UML (Unified Modeling Language [10], ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que leur comportement. L'UML a des applications qui vont au-delà du développement logiciel, notamment pour les flux de processus dans l'industrie.

Il ressemble aux plans utilisés dans d'autres domaines et se compose de différents types de diagrammes. Dans l'ensemble, les diagrammes UML décrivent la limite, la structure et le comportement du système et des objets qui s'y trouvent.

L'UML n'est pas un langage de programmation, mais il existe des outils qui peuvent être utilisés pour générer du code en plusieurs langages à partir de diagrammes UML. L'UML a une relation directe avec l'analyse et la conception orientées objet.

I.7.4. Pratique du processus UP et Extreme Programming sur notre application

Nous avons tout d'abord commencé, au début de ce chapitre, par décrire la vision du projet final à travers l'étude de l'existant, les problématiques et les solutions envisagées ceci représente la phase d'Inception du processus unifié (UP).

Ensuite, à travers les deux chapitres qui suivent, nous allons détailler les fonctionnalités et les acteurs du système et nous allons modéliser les objets à travers des diagrammes de séquences et le diagramme de classe et nous allons créer des prototypes d'interfaces, tout cela représente la phase d'élaboration du processus utilisé.

La phase suivante est la construction, qui est détaillée dans le chapitre final, nous allons utiliser alors la méthode Extreme Programming (XP) qui préconise d'écrire les tests en même temps, ou même avant la fonction à tester (Test Driven Development). Ceci permet de définir précisément l'interface du module à développer. Les tests sont exécutés durant tout le développement, permettant de visualiser si le code fraîchement écrit correspond au besoin. Nous avons alors implémenté les interfaces et réalisé quelques tests unitaires pour tester les classes et leurs attributs, puis nous avons fait des tests d'intégration afin de tester les scénarios prévus et nous avons conclus chaque itération du processus XP avec des tests end-to-end, ce qui nous a permis d'avoir des versions fonctionnelles à la fin de chaque itération.

I.8. Conclusion

Dans ce chapitre nous avons présenté le cadre général du projet, plus précisément le contexte et la problématique en élaborant une solution à cette dernière. Nous avons également défini brièvement les applications web et mobile et les GPS. Nous avons conclu par le processus de développement adapté au projet. Dans le prochain chapitre nous allons passer à la spécification et analyse des besoins.

Chapitre II : Spécification et analyse des besoins

II.1. Introduction

Ce chapitre introduira la phase d'analyse des besoins. Dans cette étape, nous identifierons d'abord les différents acteurs du système ainsi que les besoins fonctionnels qu'ils expriment, puis nous décrirons la spécification des besoins non fonctionnels et leurs contraintes afin de pouvoir modéliser le diagramme de cas d'utilisation.

II.2. Besoins fonctionnels

Les besoins fonctionnels représentent les actions que le système doit exécuter, et qu'il ne devient opérationnel que s'il les satisfait [11]. Le système à concevoir devra répondre aux besoins que nous allons citer.

II.2.1. Identification des acteurs

Un acteur représente les éléments externes qui interagissent avec le système. Cet élément peut être un utilisateur ou un système tiers (autres ordinateurs, autres programmes, bases de données). Tous les éléments extérieurs qui stimulent le système et tous les éléments extérieurs qui sont utilisés par le système sont représentés par des acteurs. Dans notre système, nous pouvons identifier 4 acteurs :

- **Client :** joue le rôle d'une personne qui sera le client final d'un commerçant.
- **Commerçant :** joue le rôle d'une personne qui est dans le domaine du commerce qui souhaite faire livrer ses colis à travers notre système.
- **Chauffeur :** joue le rôle d'une personne faisant partie de l'entreprise et qui livre les colis aux clients finals.
- **Administrateur :** Joue le rôle d'une personne faisant partie de l'entreprise et qui peut être le chef d'entrepôt, il gère les colis, les utilisateurs, les livraisons et les modules GPS afin d'assurer le bon fonctionnement du système.

II.2.2. Besoins

Afin de mettre en évidence les différentes informations accessibles selon le rôle, nous avons décidé de les regrouper par acteur.

- **Client :** Doit avoir la possibilité de consulter son colis à travers un code qu'il recevra dans son site d'achat, ainsi qu'avoir la possibilité de s'inscrire si cet utilisateur est un commerçant.
- **Commerçant :** Doit avoir un accès à son espace membre grâce à un nom d'utilisateur et un mot de passe, afin de voir, ajouter, modifier ou supprimer des colis qui seront validés et livrés par l'entreprise.

- **Chauffeur :** Doit avoir un à son espace membre grâce à un nom d'utilisateur et un mot de passe, pour gérer l'état des colis qu'il va livrer, il confirmera dans le cas où le colis est livré ou retardera dans le cas où il y a un souci.
- **Administrateur :** Doit avoir un accès à un espace admin grâce à un nom d'utilisateur et un mot de passe, ou il pourra gérer tous les colis et leurs validations, les utilisateurs (Clients, Chauffeurs), les livraisons et les GPS. Il pourra ainsi voir les statistiques de l'entreprise afin de toujours améliorer la méthode de travail.

II.3. Besoins non fonctionnels

Tous les systèmes d'information à un certain point dans leur cycle de vie doivent considérer des besoins non-fonctionnels et leurs tests. Pour certains projets ces besoins demanderont un travail très important et pour d'autres un contrôle rapide sera suffisant [12]. La liste qui va suivre représente les besoins non fonctionnels de notre application :

- **Performance :** l'application doit assurer un temps de réponse minime, tout en répondant aux exigences de l'utilisateur.
- **Sécurité :** Les données recueillies par l'application ne doivent être accessibles que par le personnel autorisé.
- **Ergonomie :** Les interfaces doivent être doivent être ergonomiques et simples à utiliser, les informations doivent être lisibles et facilement interprétables. Les actions critiques doivent toujours être confirmées avant leurs exécutions et le système ne doit pas nécessiter une formation longue durée pour être utilisé, il doit offrir une prise en main rapide et facile sans beaucoup d'efforts.
- **Modularité du code :** Le système doit être extensible pour ça le code doit être bien structuré et lisible afin de simplifier la maintenance.

II.4. Identification des cas d'utilisation

Un cas d'utilisation ou cas d'usage représente une fonctionnalité du système. Cette fonctionnalité est définie par une action, un ou plusieurs déroulements possibles et éventuellement une fin. Les différents déroulements aussi appelés scénarii seront modélisés par des diagrammes de séquence, d'activité ou d'état.

II.4.1. Client

Le *tableau II-1* décrit les cas d'utilisations de client :

N°	Cas d'utilisation
1	Consulter un colis
2	S'inscrire

Tableau II-1 - Description des cas d'utilisations du client

II.4.2. Commerçant

Le *tableau II-2* décrit les cas d'utilisations du commerçant :

N°	Cas d'utilisation
1	S'authentifier
2	Lister les colis
3	Consulter un colis
4	Gestion de colis

Tableau II-2 - Description des cas d'utilisations du commerçant

II.4.3. Chauffeur

Le *tableau II-3* décrit les cas d'utilisations du chauffeur :

N°	Cas d'utilisation
1	S'authentifier
2	Lister les colis en attente
3	Consulter un colis en attente
4	Gérer l'état d'un colis

Tableau II-3 - Description des cas d'utilisations du chauffeur

II.4.4. Administrateur

Le *tableau II-4* décrit les cas d'utilisations de l'administrateur :

N°	Cas d'utilisation
1	S'authentifier
2	Consulter les statistiques
3	Gestion des colis (ajouter, modifier, supprimer, consulter, validé les colis)
4	Gestion des utilisateurs (ajouter, modifier, supprimer, consulter)
5	Gestion des livraisons (ajouter, modifier, supprimer, consulter, assigner un groupe de colis à un chauffeur)
6	Gestion des GPS (ajouter, modifier, supprimer, consulter)

Tableau II-4 - Description des cas d'utilisations de l'administrateur

II.5. Diagramme des cas d'utilisation

Les diagrammes de cas d'utilisation servent à décrire un ensemble d'actions (cas d'utilisation) que certains systèmes devraient ou peuvent exécuter en collaboration avec un ou plusieurs utilisateurs externes du système (acteurs). Nous allons ici donner sur la *figure II-1* le diagramme de cas d'utilisation des 4 acteurs du système pour avoir une vue globale de ce dernier.

Note : *Seulement les utilisateurs ayant le code de suivi pourrons consulter des colis.*

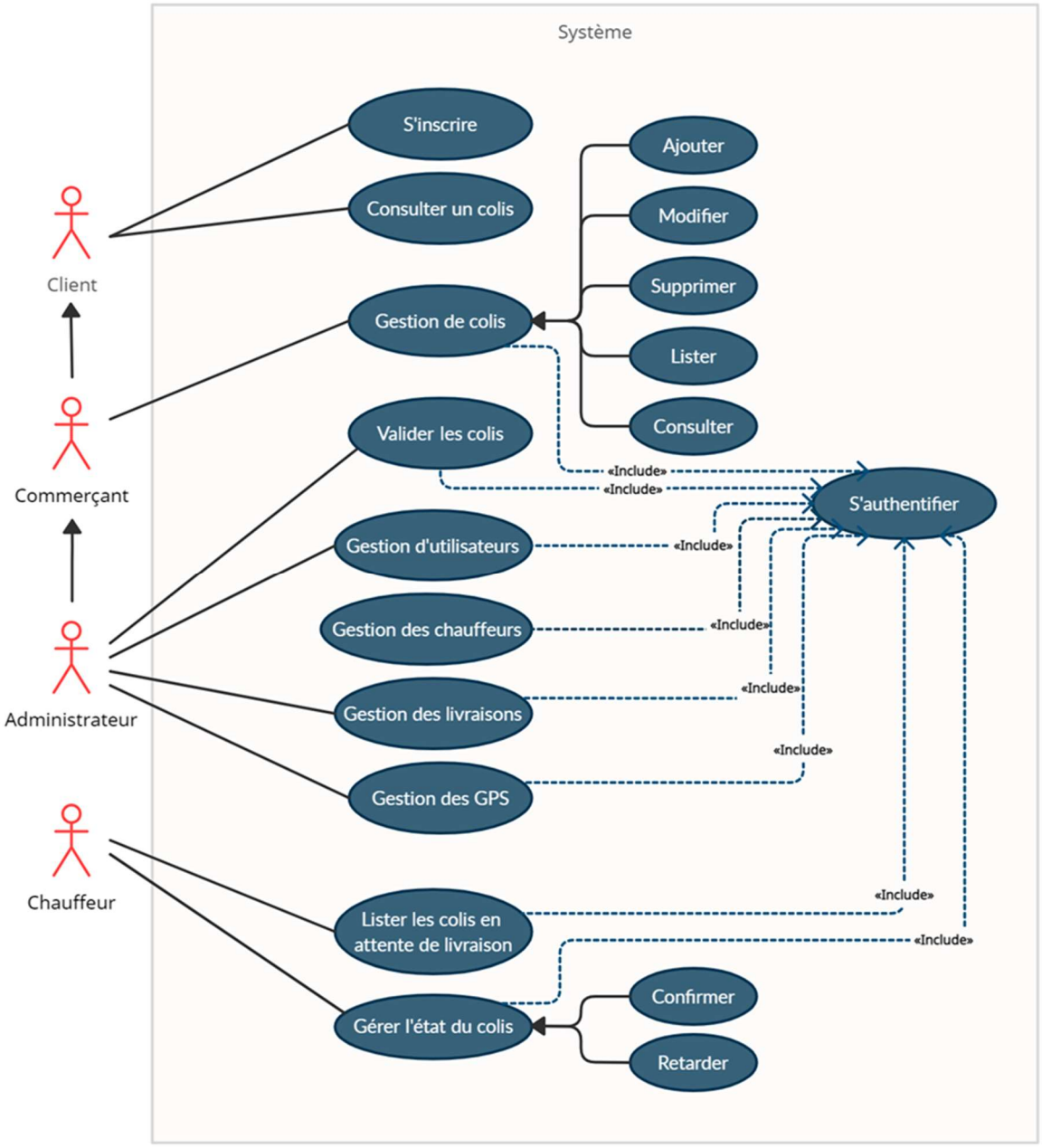

Figure II-1 - Diagramme de cas d'utilisation

II.6. Descriptions des cas d'utilisations

Les cas d'utilisations représentent non seulement les interactions avec les acteurs, mais également les prés et post conditions ainsi que les enchaînements alternatifs. Dans cette section nous décrirons les cas d'utilisations des différents acteurs.

II.6.1. Description du cas d'utilisation « S'authentifier »

Cas d'utilisations « S'authentifier »	
Titre	S'authentifier
Acteur	Commerçant, Chauffeur, Administrateur
Résumé	L'acteur doit s'identifier en saisissant son nom d'utilisateur et mot de passe pour accéder à son espace personnel.
Date de création	01/09/2021
Version	V1.0.0
Descriptions des scénarios	
Préconditions	L'utilisateur doit être créé (Chauffeur) ou validé (Commerçant) par l'administrateur.
Scénario nominal	1. Le système affiche le formulaire pour saisir le code de suivi. 2. L'acteur saisit le code de suivi. 3. Le système vérifie si le code de suivi existe et qu'il est bien correct. 4. Le système affiche l'état de livraison du colis et sa position exacte sur la carte.
Enchaînement alternatif	Le code saisi est incorrect : 1. Le système affiche un message d'erreur pour signaler que le code saisi est incorrect. 2. Le cas d'utilisation reprend de l'étape 1 du scénario nominal.
Postconditions	L'utilisateur est authentifié et accède aux fonctionnalités qui lui sont dédiées.

Tableau II-5 - Description du cas d'utilisation « S'authentifier »

II.6.2. Description du cas d'utilisation « Consulter un colis »

Cas d'utilisations « Consulter un colis »	
Titre	Consulter un colis
Acteur	Client
Résumé	L'acteur doit se rendre sur la page spécifique et saisir le code de sa livraison afin de suivre sa commande en direct.
Date de création	01/09/2021
Version	V1.0.0
Descriptions des scénarios	
Préconditions	/
Scénario nominal	1. Le système affiche le formulaire d'authentification. 2. L'acteur saisit le nom d'utilisateur ainsi que son mot de passe. 3. Le système vérifie si les identifiants saisis sont corrects. 4. Le système affiche le tableau de bord.
Enchaînement alternatif	Les identifiants saisis par l'acteur sont incorrects : 1. Le système affiche un message d'erreur pour signaler que les identifiants sont incorrects. 2. Le cas d'utilisation reprend de l'étape 1 du scénario nominal.
Postconditions	L'utilisateur est authentifié et accède aux fonctionnalités qui lui sont dédiées.

Tableau II-6 - Description du cas d'utilisation « Consulter un colis »

II.6.3. Description du cas d'utilisation « Gestion de colis

Cas d'utilisations « Gestion de colis »	
Titre	Gestion de colis
Acteur	Commerçant
Résumé	L'acteur doit se rendre sur la page des colis afin de pouvoir ajouter, modifier, supprimer des colis.
Date de création	01/09/2021
Version	V1.0.0

Descriptions des scénarios	
Préconditions	Être authentifié.
Scénario nominal	1. Le système affiche un tableau de tous les colis 2. L'acteur consulte un colis ou choisit une action de gestion (Ajout, modification, suppression). 3. Le système vérifie si les modifications ou l'ajout sont corrects. 4. Le système actualise le tableau de tous les colis avec les nouvelles lignes dans le cas d'ajout ou des lignes en moins dans le cas de suppression.
Enchaînement alternatif	Les informations saisis par l'acteur sont incorrectes : 1. Le système affiche un message d'erreur pour signaler que les informations sont incorrectes. 2. Le système demande à l'utilisateur de corriger les erreurs ou de revenir à l'étape précédente.
Postconditions	Le système enregistre les modifications.

Tableau II-7 - Description du cas d'utilisation « Gestion de colis »

II.6.4. Description du cas d'utilisation « Valider les colis »

Cas d'utilisations « Valider les colis »	
Titre	Valider les colis
Acteur	Administrateur
Résumé	L'acteur doit se rendre sur la page des colis pour pouvoir valider ou décliner les colis inscrits par les clients ceci selon si les clients ont rendu les colis ou pas.
Date de création	02/09/2021
Version	V1.0.0
Descriptions des scénarios	
Préconditions	Être authentifié.
Scénario nominal	1. Le système affiche le tableau de tous les colis inscrits. 2. L'acteur consulte les groupes de colis et valide ou met en attente les groupes de colis selon si les colis ont été repris par l'entreprise ou remis par le client. 3. Le système vérifie si l'action à bien été effectuée.

	4. Le système actualise le tableau de tous les colis avec le nouvel état des colis.
Enchaînement alternatif	L'action de validation est interrompue : 1. Le système affiche un message d'erreur pour signaler que l'action n'a pas bien été effectuée. 2. Le système demande à l'utilisateur de réessayer avec un bouton simple qui lui sera affiché.
Postconditions	Le système enregistre le nouvel état des colis, cela indiquera aux clients que leur colis est en expédition par une notification qui leur sera envoyée.

Tableau II-8 - Description du cas d'utilisation « Valider les colis »

II.6.5. Description du cas d'utilisation « Gestion des livraisons »

Cas d'utilisations « Gestion des livraisons »	
Titre	Gestion des livraisons
Acteur	Administrateur
Résumé	L'acteur doit se rendre sur la page des livraisons et pourra créer de nouvelles livraisons en choisissant le groupe de colis à livrer vers une adresse spécifique et assigner la livraison à un chauffeur.
Date de création	03/09/2021
Version	V1.0.0
Descriptions des scénarios	
Préconditions	Être authentifié.
Scénario nominal	1. Le système affiche le tableau des livraisons. 2. L'acteur consulte les livraisons et choisit une action d'ajouter, modifier ou supprimer une livraison. 3. L'acteur remplit le formulaire d'ajout de nouvelle livraison et assigne la livraison à un chauffeur et assigne le GPS pour cette livraison. 4. Le système actualise le tableau des livraisons avec la nouvelle livraison.

Enchaînement alternatif	Les informations saisies sont incorrectes : 1. Le système affiche un message d'erreur pour signaler que les informations sont incorrectes. 2. Le système demande à l'utilisateur de corriger les erreurs afin d'enregistrer à nouveau la livraison.
Postconditions	Le système enregistre la nouvelle livraison et envoie une notification au chauffeur lui indiquant la nouvelle mission.

Tableau II-9 - Description du cas d'utilisation « Gestion des livraisons »

II.6.6. Description du cas d'utilisation « Lister les colis en attente de livraison »

Cas d'utilisations « Lister les colis en attente de livraison »	
Titre	Lister les colis en attente de livraison
Acteur	Chauffeur
Résumé	L'acteur doit se rendre sur la page de missions
Date de création	06/09/2021
Version	V1.0.0
Descriptions des scénarios	
Préconditions	Être authentifié.
Scénario nominal	1. Le système affiche la liste des livraisons en attente de livraison. 2. L'acteur consulte les livraisons afin de voir les détails de chaque livraison qui sera une mission pour lui.
Enchaînement alternatif	La liste des livraisons est vide : 1. Le système affiche un message d'indication que les livraisons sont toutes effectuées et vouloir attendre les prochaines missions.
Postconditions	/

Tableau II-10 - Description du cas d'utilisation « Lister les colis en attente de livraison »

II.6.7. Description du cas d'utilisation « Gérer l'état du colis »

Cas d'utilisations « Gérer l'état du colis »	
Titre	Gérer l'état du colis
Acteur	Chauffeur
Résumé	L'acteur doit se rendre sur la page des livraisons et consulte la livraison en question de livraison, et valide ou retarde cette dernière selon si le colis a été livré ou pas.
Date de création	09/09/2021
Version	V1.0.0
Descriptions des scénarios	
Préconditions	Être authentifié.
Scénario nominal	1. Le système affiche la liste des livraisons. 2. L'acteur choisit la livraison qui est en cours d'être effectuée. 3. L'acteur valide la livraison du colis spécifique et fait signer le destinataire à travers l'interface.
Enchaînement alternatif	Le destinataire ne s'est pas rendu à son colis ou problème de livraison : 1. L'acteur choisit l'action de retarder la livraison cela indique aux administrateurs que la livraison est retardée et sera reportée pour une date ultérieure.
Postconditions	Le système enregistre la confirmation de livraison ou le retardement.

Tableau II-11 - Description du cas d'utilisation « Gérer l'état du colis »

II.7. Identité visuelle et maquettage

Étant donné que nous avons supposé précédemment la création d'une entreprise pour résoudre notre problématique, il nous faut définir ce qui nous a poussé à choisir son nom et détailler les aspects de son identité de marque, ainsi que quelques prototypes et maquettes de la solution qu'elle représente.

II.7.1. Nom et logo de l'application

Nous avons décidé de nommer cette application "Delivo", "deliv" qui réfère à "delivery" qui signifie livraison en anglais, et nous avons ajouté un "o" vers la fin pour créer une identité unique. Le logo est basé sur l'idée d'utiliser un colis avec des petites lignes qui

sortent de ce dernier pour donner un effet de vitesse et exprimer la rapidité du service (*Figure II-2*)

Figure II-2 - Logo Delivo

II.7.2. Palette de couleur

Nous avons décidé d'utiliser deux couleurs principales qui sont le bleu et le jaune. Le premier est une couleur qui symbolise la paix, quant au second il s'agit d'une couleur audacieuse et dynamique, ce choix de couleurs a pour but de donner un aspect de sérénité et de dynamicité aux clients. Nous présentons alors la palette de couleur sur la (*Figure II-3).*

Figure II-3 - Palette de couleur Delivo

II.7.3. Typographie

Le choix de la typographie de notre application repose sur le besoin d'avoir un rendu optimal du texte sur chaque appareil et système d'exploitation. Dans cette démarche, nous avons décidé de choisir la police : Roboto.

II.7.4. Prototypage et maquettage

Nous avons réalisé quelques prototypes et maquettes de notre application afin d'avoir un rendu de ce que nous allons avoir lors de la réalisation.

La maquette de site web correspond à une esquisse, voire un prototype d'un site internet en création. Le processus de création graphique qui donne lieu à la création de maquettes graphiques est appelé « maquettage ». La réalisation des maquettes se concentre principalement sur l'aspect graphique et le design d'un site web et de ses interfaces [13].

Nous présentons quelques prototypes sur les figures *(II-4, II-5, II-6)* qui correspondent à la page d'accueil, page gestion des colis et page missions chauffeur respéctivement.

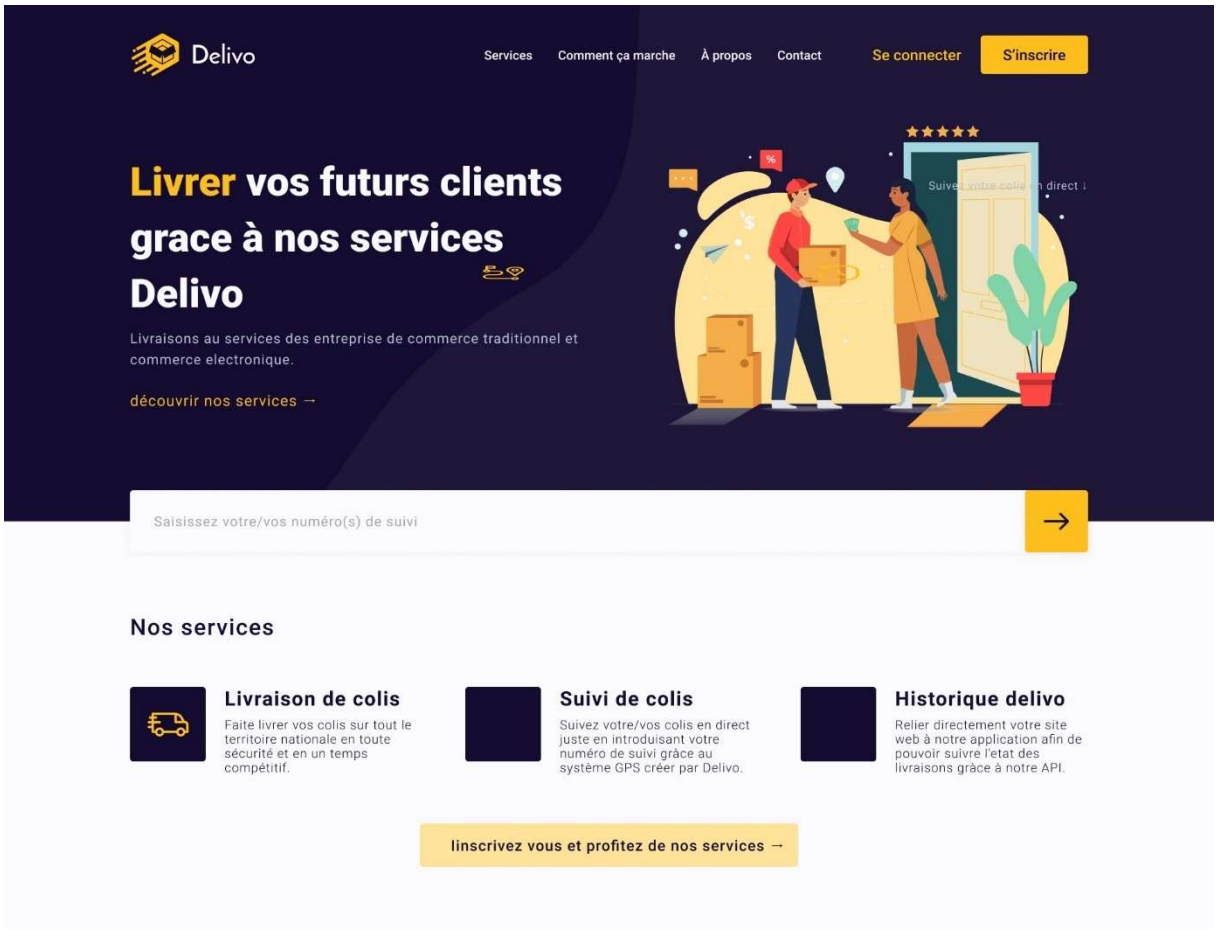

Figure II-4 - page d'accueil

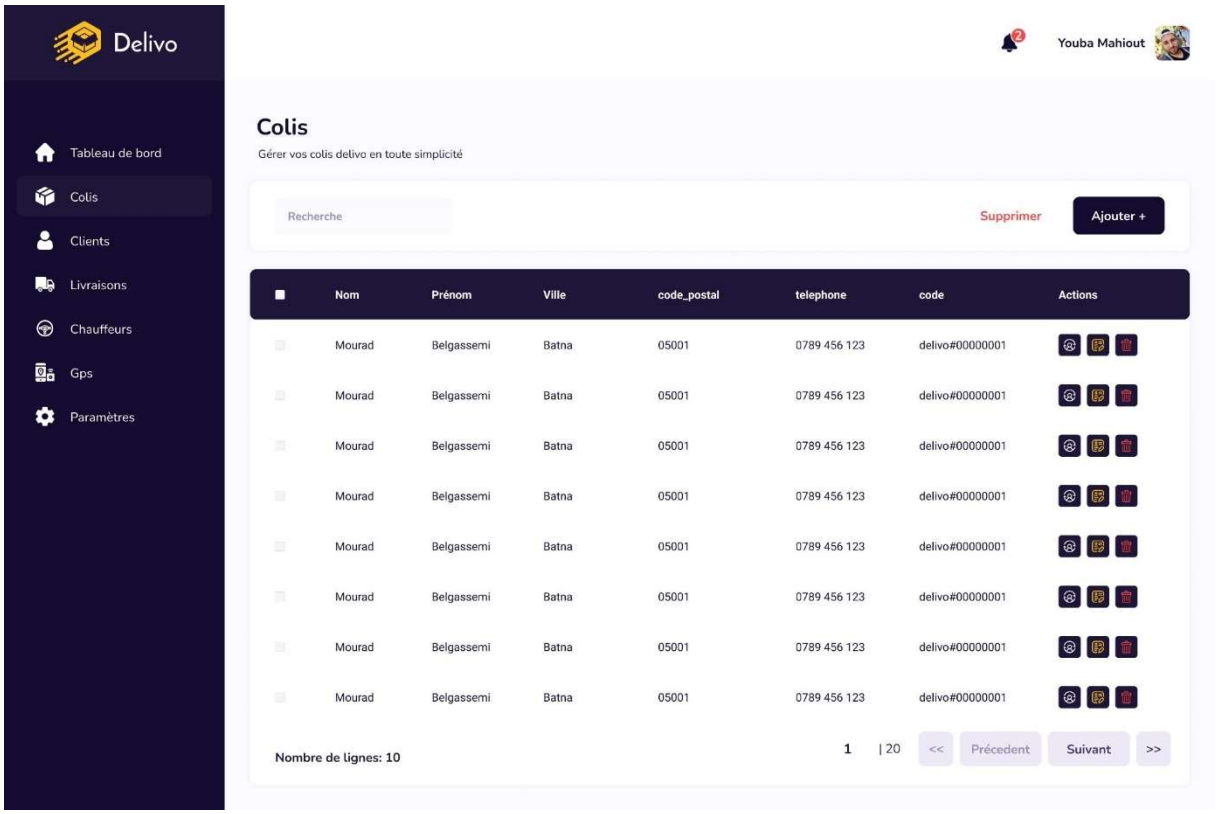

Figure II-5 - Maquette gestion des colis

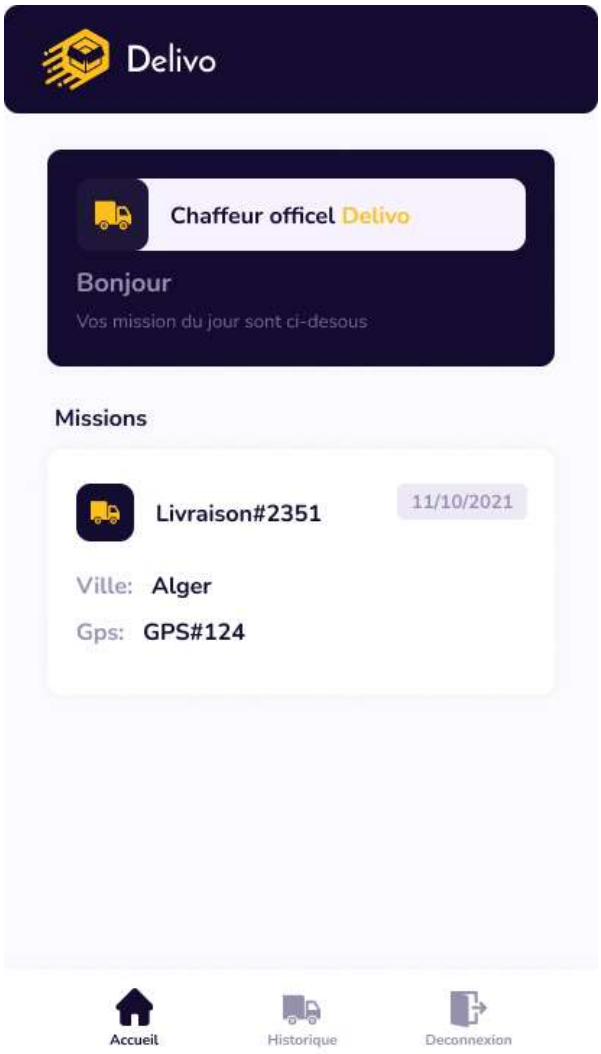

Figure II-6 - Maquette mission du chauffeur

II.8. Conclusion

Ce chapitre nous a permis d'exprimer et d'analyser les besoins permettant de décrire les fonctionnalités du système de manière globale. Nous avons alors identifié les acteurs, les besoins fonctionnels et non fonctionnels du système, et nous avons réalisé un diagramme des cas d'utilisation. Cela nous a permis de faire des descriptions pour chaque cas d'utilisation. Enfin nous avons fini par la réalisation d'une identité visuelle ainsi que les prototypes et squelettes de l'application pour un avoir un rendu de cette dernière. Dans le prochain chapitre nous allons passer à la conception détaillée de notre application.

Chapitre III : Conception

III.1. Introduction

Dans ce chapitre nous allons nous intéresser à la conception détaillée de notre système. Le but est de définir et de mettre en place les choix d'architecture technique, et de compléter la description du système. Nous débuterons avec les différents diagrammes de séquences, ces derniers nous permettront de décrire les interactions au sein de notre système et vont nous permettre de réaliser le diagramme de classes. Nous terminerons par présenter le modèle orienté documents structurés qui à son tour nous permettra de décrire la structure des données stockées et utilisées par notre système.

III.2. Diagrammes de séquence

Les diagrammes de séquence [14] sont la représentation graphique des interactions entre les acteurs et un ensemble d'objets selon un ordre chronologique. Ces objets sont des instances des trois types de classes (dialogues, contrôles et entités).

Dans cette partie nous allons modéliser et présenter pour chaque acteur un diagramme séquence d'un cas d'utilisation, tout en respectant les règles d'interaction suivantes :

- 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.

III.2.1. Diagramme de séquence du cas d'utilisation « Consulter colis »

A partir de l'interface « ConsulationColis », l'utilisateur peut suivre en direct son colis. Pour cela, il doit saisir le code de suivi. Une vérification du format est déclenchée après la saisie, tant que le format est incorrect, l'utilisateur devrait ressaisir le code. Le contrôleur « CtrColis » aura pour rôle de vérifier l'existence du colis, si ce dernier existe il redirigera l'utilisateur vers une nouvelle interface, dans le cas contraire il affichera un message d'erreur.

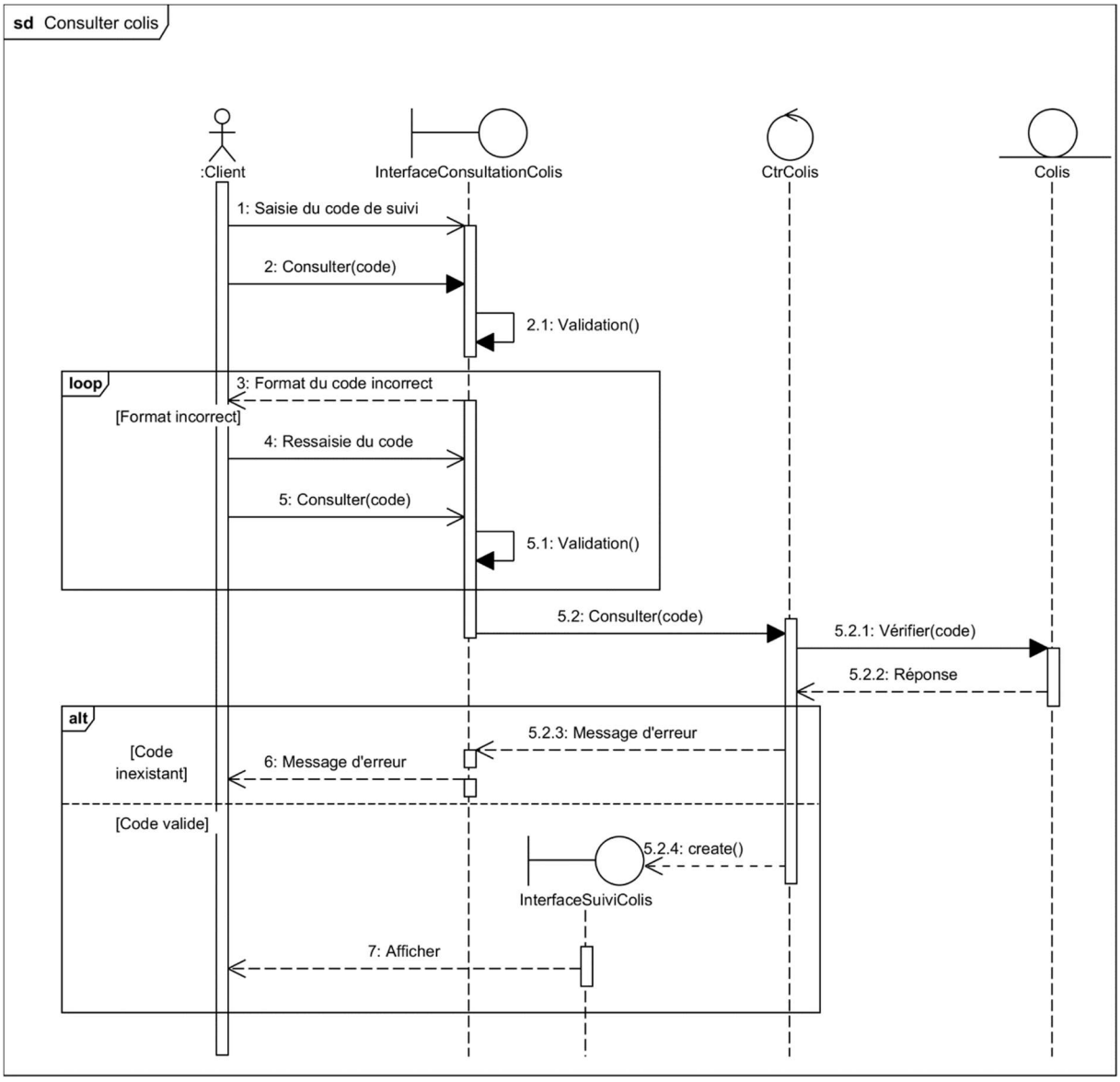

Figure III-1 - Diagramme de séquence « Consulter colis »

III.2.2. Diagramme de séquence du cas d'utilisation « S'authentifier »

Pour s'authentifier, le client accède vers l'interface d'authentification où il saisit son nom d'utilisateur et son mot de passe. Si l'un des champs est vide, le client devra saisir à nouveau ces derniers. Le contrôleur « ctrAuthentification » vérifiera si les informations sont identiques, si la vérification aboutit le client se voit autoriser l'accès et sera redirigé vers le tableau de bord, dans le cas contraire un message d'erreur sera affiché.

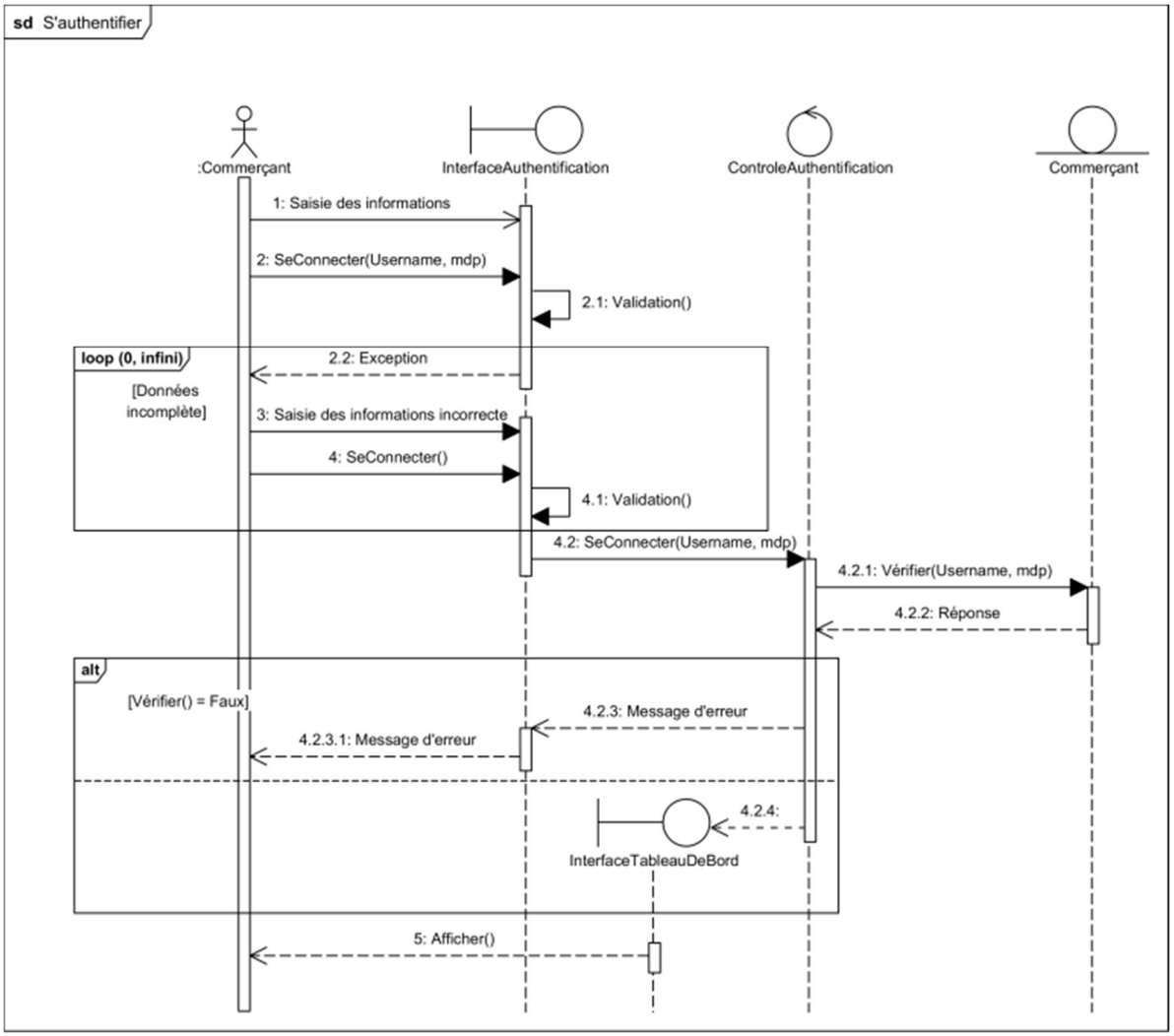

Figure III-2 - Diagramme de séquence « S'authentifier »

III.2.3. Diagramme de séquence du cas d'utilisation « Valider colis »

Pour consulter la liste des colis et les valider, l'administrateur devra s'authentifier, une fois cette étape effectuée il pourra changer le statut des colis à travers l'interface « InterfaceColis ». Si la validation est interrompue, le contrôleur « ctrColis » affichera un message d'erreur, dans le cas contraire la mise à jour du statut sera effectuée et les données de l'interface seront actualisées.

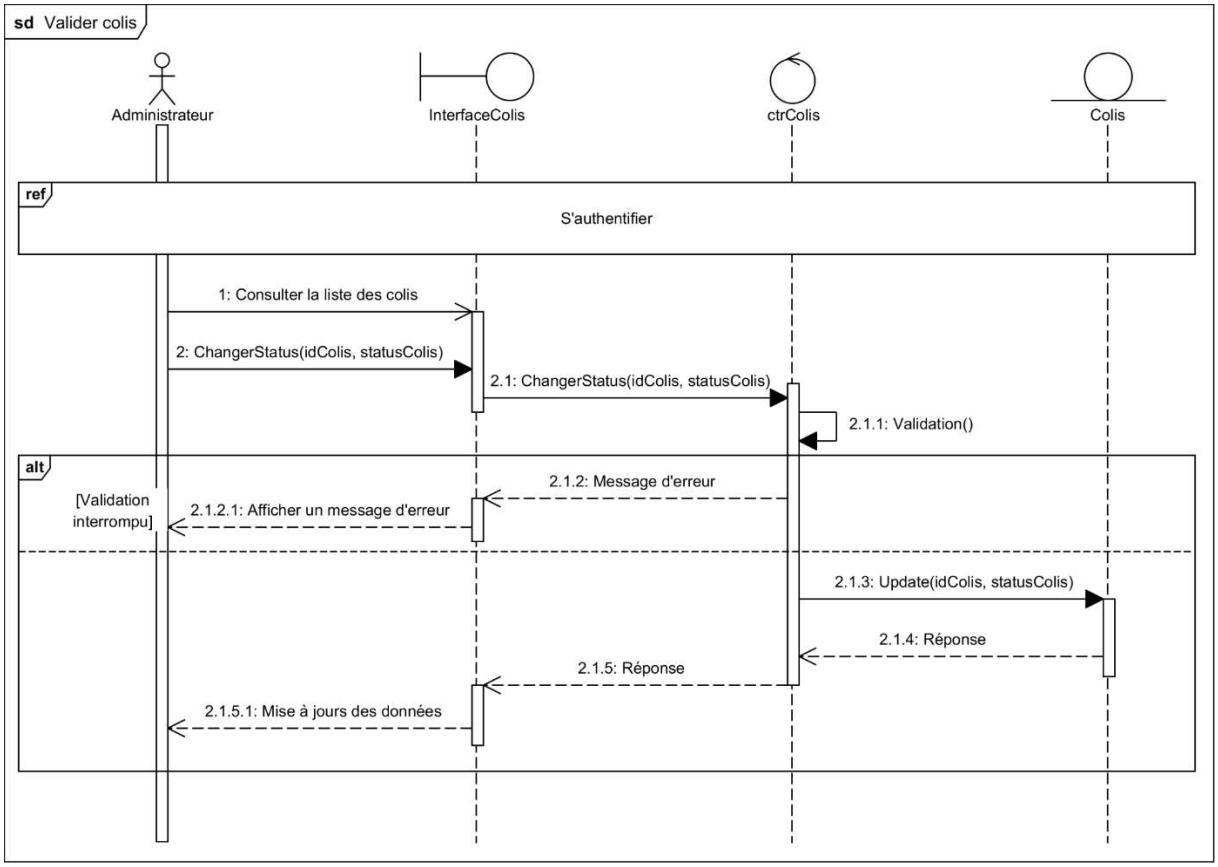

Figure III-3 - Diagramme de séquence « Valider colis »

III.2.4. Diagramme de séquence du cas d'utilisation « Lister les colis en attente »

Pour consulter la liste des colis en attente, le chauffeur devra s'authentifier, une fois cette étape effectuée il pourra consulter les détails à travers l'interface « InterfaceMissions ».

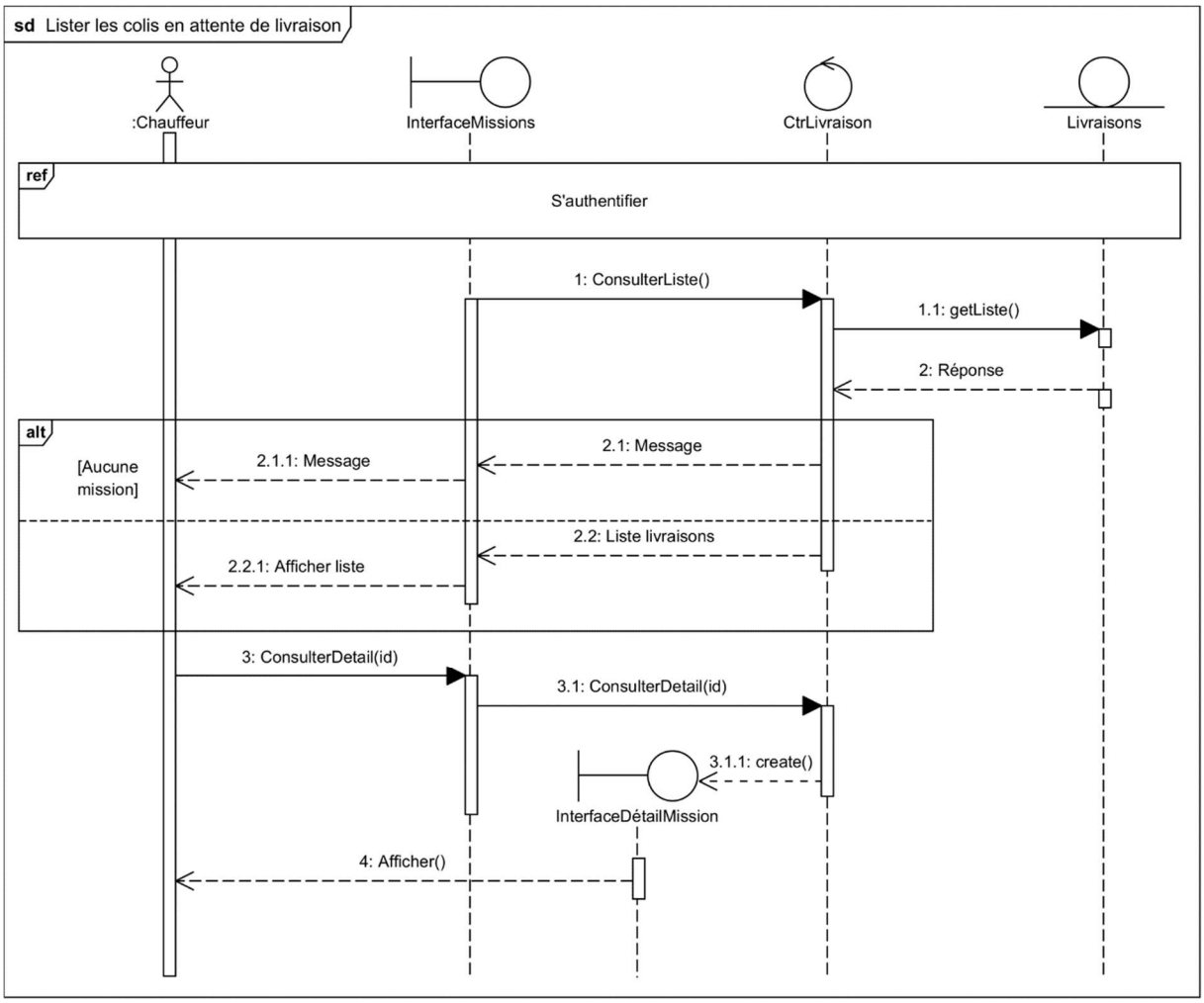

Figure III-4 - Diagramme de séquence « Lister les colis en attente »

III.3. Diagramme de classe

Le diagramme de classe représente les classes intervenant dans le système. Il décrit clairement la structure d'un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets.

La classe est un concept abstrait qui permet de représenter toutes les entités d'un système. Une classe peut donc représenter une voiture, un bouton cliquable, un devis, un utilisateur connecté, une structure de donnée ou tout autre élément devant être modélisé et donnant généralement lieu à la génération d'un code informatique. La classe est définie par son nom, ses attributs et ses opérations [15].

La figure *(Figure III-5)* représente le diagramme de classe que nous avons modélisé pour notre système.

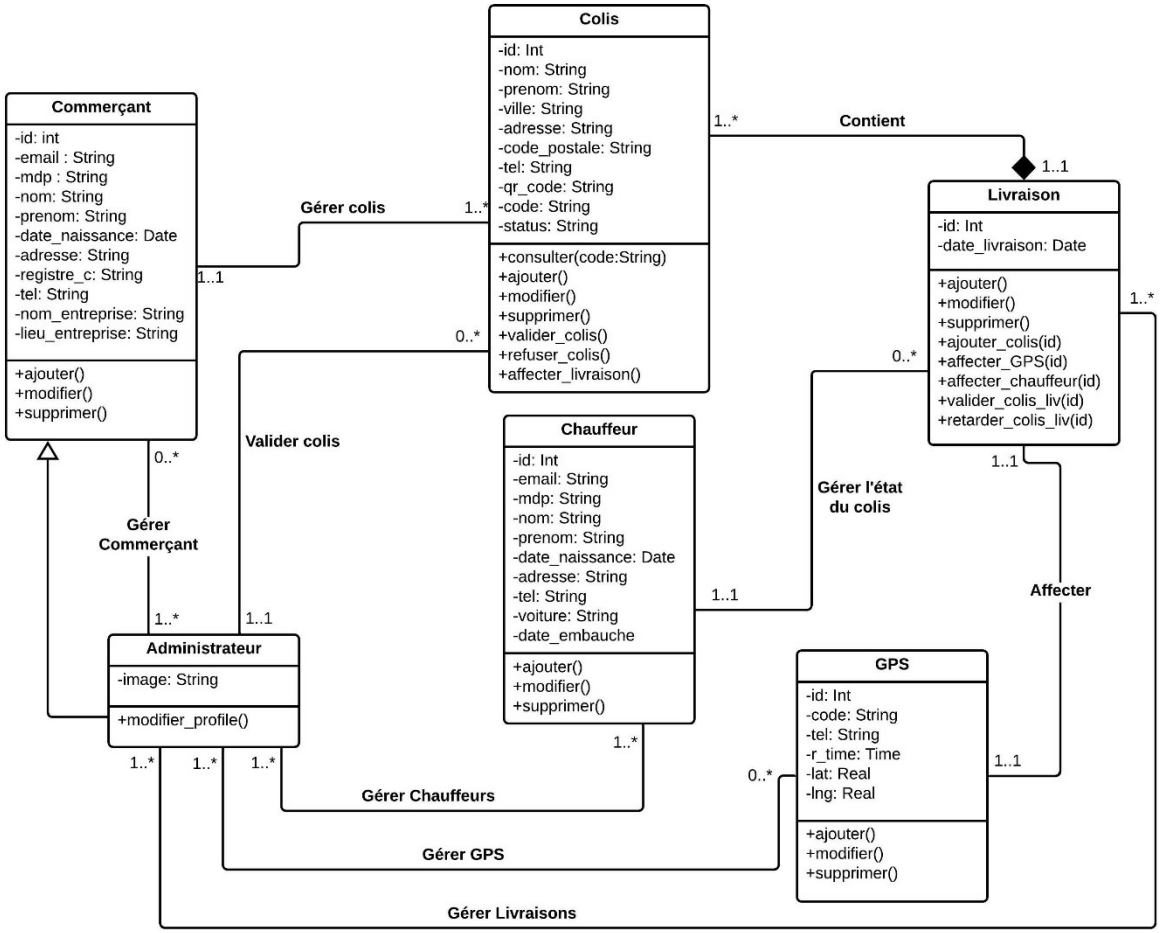

Figure III-5 - Diagramme de classe

III.4. Modèle orienté document

Concernant la structure des données de l'application nous avons choisi de dévier du modèle relationnel, et de nous intéresser au NoSQL, c'est-à-dire les base de données dites non relationnelles et plus précisément au modèle orienté document.

III.4.1. Modèle relationnel et modèle orienté document

Les bases de données SQL sont connues sous le nom de bases de données relationnelles, et ont une structure de données basée sur des tableaux, avec un schéma strict et prédéfini requis. Les bases de données NoSQL, ou bases de données non relationnelles, peuvent être basées sur des documents, des bases de données graphiques, des paires clé-valeur ou des magasins à colonnes larges. Les bases de données NoSQL ne nécessitent aucun schéma prédéfini, ce qui vous permet de travailler plus librement avec des "données non structurées". Les bases de données relationnelles évoluent verticalement, mais généralement plus coûteuses, alors que la nature évolutive horizontale des bases de données NoSQL est plus rentable [16].

III.4.2. Modélisation

Les bases de données NoSQL sont conçues pour s'affranchir des lignes et des colonnes du modèle de base de données relationnelle. Mais c'est une erreur commune de penser que les bases de données NoSQL n'ont aucune sorte de modèle de données. Une description utile de la manière dont les données seront organisées est le début d'un schéma.

Les bases de données de documents stockent les données sous forme de documents. Chaque document stocke des paires de champs et de valeurs, avec une grande variété de types de données et de structures de données utilisées comme valeurs. Les développeurs intègrent généralement la structure des champs et des valeurs du document dans les objets de code de leurs applications. Les requêtes sont utilisées pour récupérer les valeurs des champs, et un certain nombre de langages de requête puissants ont été développés pour exploiter la variété des types de valeurs des champs [17].

Nous utilisons le modèle de documents structurés pour définir les documents présents dans notre structure de données (Figure III-6).

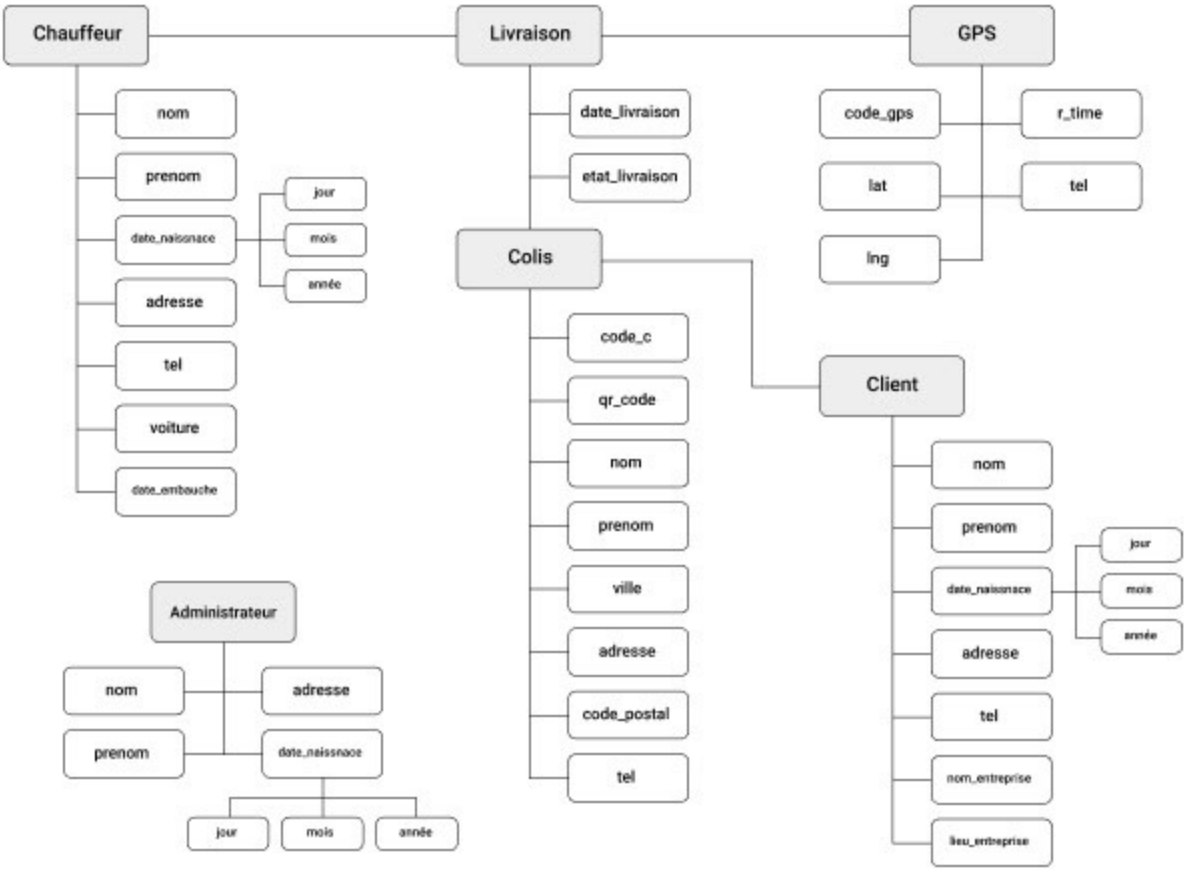

Figure III-6 - Modèle de documents structurés

III.5. Conclusion

Au cours de ce chapitre nous avons en premier lieu modélisé les diagrammes de séquences qui nous ont donné une vue approfondie sur les interactions entre les objets et cela nous a permis de réaliser le diagramme de classe. Nous avons ensuite présenté les modèles orientés documents pour avoir une vue globale de notre base de données. Dans le prochain chapitre nous allons passer à la réalisation de notre application.

Chapitre IV : Réalisation

IV.1. Introduction

Ce chapitre sera divisé en cinq sections, sur la première section nous allons exposer l'environnement de développement que l'on a utilisé pour réaliser l'application. Ensuite la deuxième section sera sur le module GPS au sein de laquelle nous allons détailler l'aspect matériel du projet notamment les composants utilisés et l'assemblage de ces derniers. La section suivante présente l'application web des deux côtés serveur et client, ou nous allons parler des langages, librairies et Framework et bases de données utiliser et nous présenterons quelques interfaces réalisées. Puis sur la prochaine section nous présenterons l'application mobile avec les langages et librairies utilisés et nous présenterons aussi quelques interfaces que nous avons mises en place. Enfin nous finissons par la dernière section qui sera consacrée au test que nous avons pu faire durant tout le processus de développement de notre application.

IV.2. Environnement de développement

Pour la réalisation de l'application nous avons tout d'abord repris les maquettes que nous avons réalisées lors de l'étape de spécification des besoins avec le logiciel « Figma ». Puis nous nous sommes attelés à la partie module GPS du projet et nous avons programmé le comportement de notre module GPS via le langage C grâce à l'IDE « Arduino ».

IV.2.1. Figma

Figma [18] est une application Web d'édition graphique et de conception d'interfaces utilisateur. Celle-ci est utilisée pour réaliser toutes sortes de travaux de conception graphique, de l'élaboration de sites Web à la conception d'interfaces d'applications mobiles, en passant par le prototypage de modèles, la création de messages sur les médias sociaux, etc.

Figma est différent des autres outils d'édition graphique. Principalement parce qu'il fonctionne directement sur navigateur. Cela signifie que l'on peut accéder aux projets et commencer à concevoir à partir de n'importe quel ordinateur ou plate-forme sans avoir à acheter plusieurs licences ou à installer un logiciel.

IV.2.2. Arduino IDE et langage C

L'Arduino IDE [19] est un environnement fournit par les créateurs d'Arduino qui permet d'écrire, de compiler et de transférer un programme vers une carte Arduino. Le langage de programmation supporté par cette carte est le langage C qui est un langage de programmation procédural de bas niveau, utilisé dans tous les domaines, de la création de systèmes d'exploitation (Unix, Windows…) à la création de jeux vidéo (Counter Strike, Tomb Raider…), en passant par la création de logiciels (PhotoShop, Autocad…), la création d'autres

langages de programmation (Java, Python…) et la programmation de microcontrôleurs (Arduino, Raspberry…).

IV.2.3. Git

Git [20] est un système de contrôle de version gratuit et open source, créé à l'origine par Linus Torvalds en 2005. Contrairement aux anciens systèmes de contrôle de version centralisés tels que SVN et CVS, Git est distribué : chaque développeur dispose localement de l'historique complet de son dépôt de code. Cela rend le clonage initial du dépôt plus lent, mais les opérations ultérieures telles que “commit”, “blame”, “diff”, “merge” et "log" sont beaucoup plus rapides.

Git offre également un excellent support pour le branchement, la fusion et la réécriture de l'historique du dépôt, ce qui a conduit à de nombreux outils et flux de travail innovants et puissants. Les demandes de retrait sont l'un de ces outils populaires qui permettent aux équipes de collaborer sur les branches Git et de réviser efficacement le code des autres. Git est le système de contrôle de version le plus largement utilisé dans le monde aujourd'hui et est considéré comme la norme moderne pour le développement de logiciels.

IV.2.4. Visual Studio Code

Visual Studio Code [21] est un éditeur de texte écrit en JavaScript via Electron.JS par Microsoft, qui lui permet de s’exécuter sur différents systèmes d’exploitation, cet éditeur de code supporte la majorité des langages de programmations existant, et il peut aussi exécuter du code grâce à aux extensions crées par la communauté, ce qui lui permet de devenir un IDE complet. En plus de la capacité d’installer des extensions il embarque de base une puissante fonction permettant d’analyser et d’autocompléter le code appelée IntelliSense.

IV.3. Module GPS

A ce stade de la réalisation, nous nous somme fourni une carte Arduino, plus précisément l’arduino Mega, ainsi qu’un module GPS et GSM compatible le GSM SIM 808.

IV.3.1. Arduino Mega

Arduino est une plateforme électronique open-source basée sur du matériel et des logiciels faciles à utiliser. Les cartes Arduino sont capables de lire des entrées - de la lumière sur un capteur, un doigt sur un bouton ou un message Twitter - et de les transformer en une sortie - activer un moteur, allumer une LED, publier quelque chose en ligne.

L’Arduino Mega [22] est le modèle le plus perfectionné et puissant de la célèbre carte électronique. Elle permet d’effectuer un maximum d’actions et délivre un potentiel tel, qu’il est possible de se pencher sur les montages les plus lourds et gourmands en code.

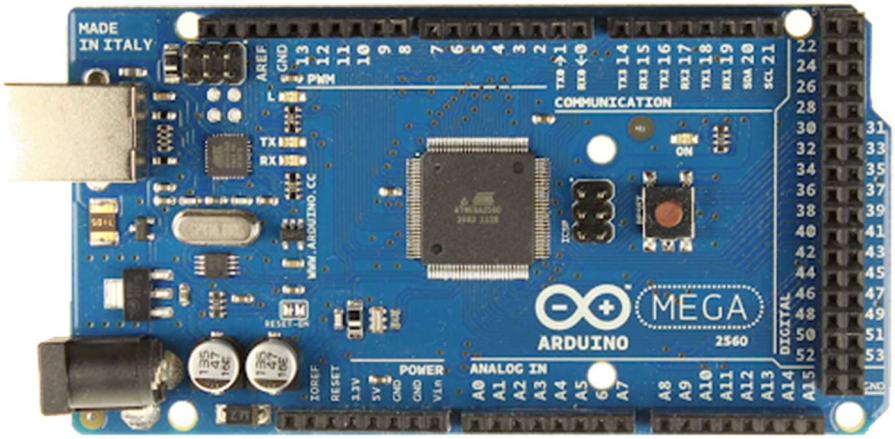

Figure IV-1 - Photo Arduino Mega

IV.3.2. GSM SIM 808

C'est un module équipé d'un transpondeur GPS pour la géolocalisation ainsi qu'un emplacement SIM permettant l'utilisation de d'une puce GSM et donc de données mobiles.

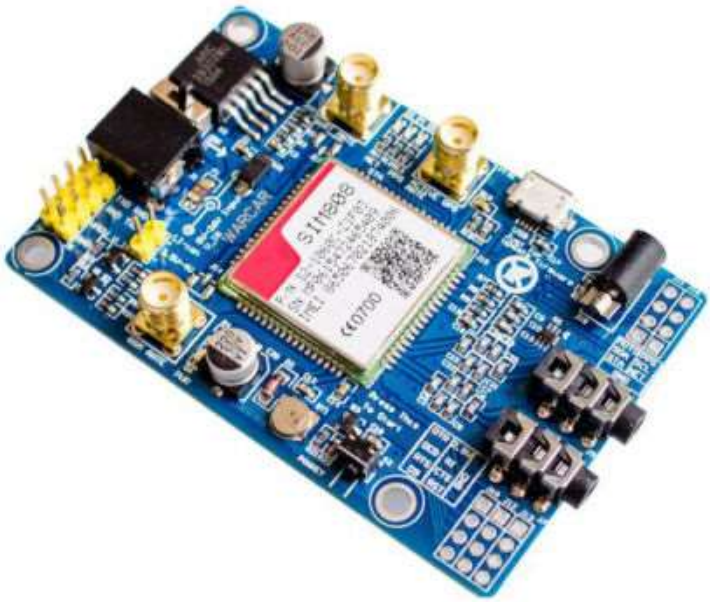

Figure IV-2 - Photo Module GPS

IV.3.3. Réalisation du module

Le montage de notre module de suivi est expliqué dans la *figure IV-3*. Nous avons tout d'abord branché l'antenne GPS et l'antenne GSM dans leurs emplacements respectifs au sein du module GSM SIM808, nous avons par la suite connecté ce dernier à notre microcontrôleur Arduino via 3 câbles, RX pour la réception et TX pour l'émission de données, et GND représente la prise de terre pour plus de sécurité. Finalement nous avons injecté notre code à partir de l'IDE Arduino.

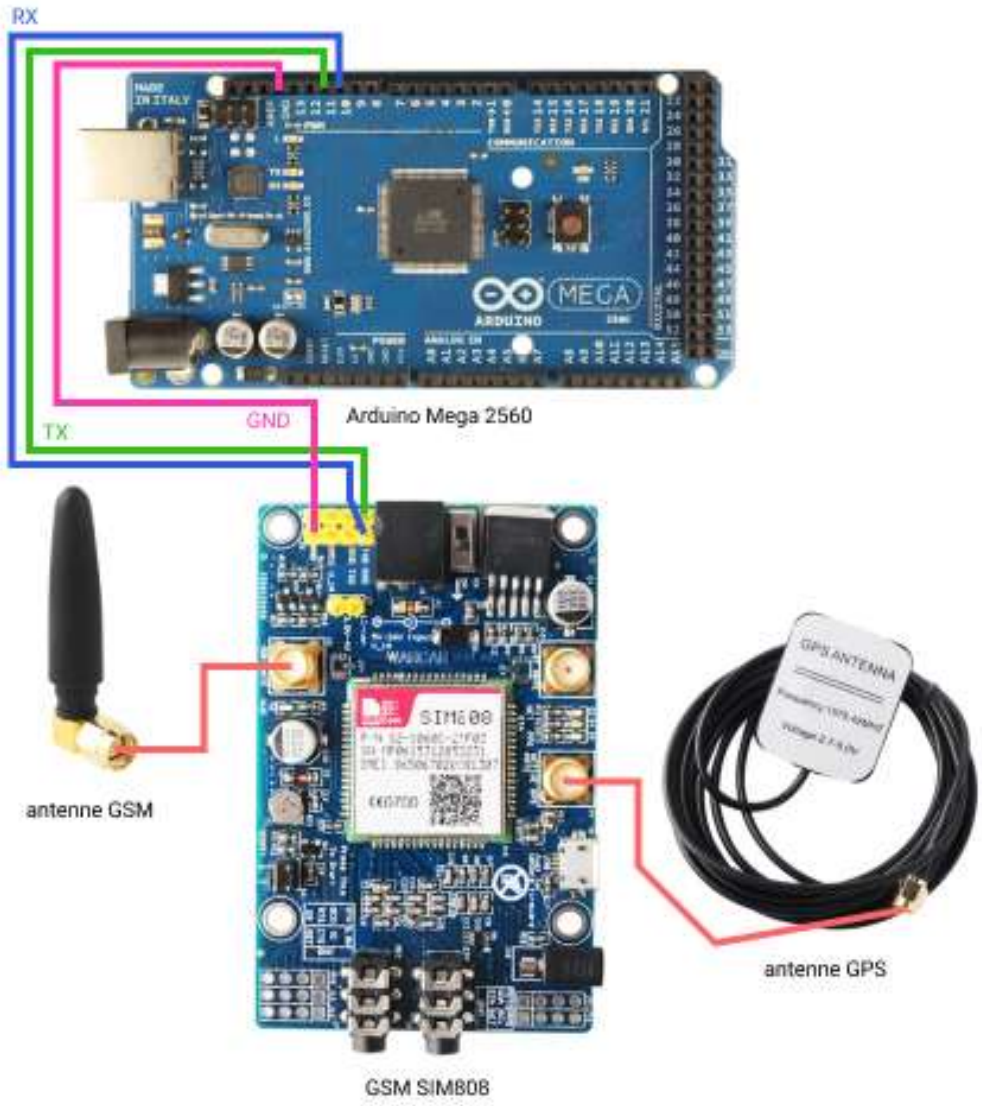

Figure IV-3 - Schéma de montage

IV.4. Application web

Sur le côté Client, notre application web, écrite en Javascript en utilisant la bibliothèque React.JS ainsi que le Framework CSS Tailwind, est divisée en deux sections, un back-office permettant la gestion et l'administration des données de l'application, et une

section publique affichant les données relatives à l'entreprise de livraison, et qui permet de suivre un colis. Le tout est géré par le côté serveur, c'est ce dernier qui fait le lien entre les modules GPS, le back-office, la section publique et l'application mobile. Nous avons réalisé le serveur sous forme d'API suivant l'architecture REST, et ce grâce au Framework Express.JS, et le stockage de données sous forme de documents grâce à MongoDB.

IV.4.1. Client

IV.4.1.a. React.JS

ReactJS [23] est une bibliothèque JavaScript libre développée par Facebook pour la création d'interfaces utilisateur interactive. Cette bibliothèque permet de créer des interfaces (UI) complexes grâce à un système de composants simples et réutilisables, et d'actualiser les données présentes dans ces composants indépendamment du reste de la page.

IV.4.1.b. Tailwind

Tailwind [24] CSS est essentiellement un Framwok CSS axé sur l'utilité qui permet de créer rapidement des interfaces utilisateur personnalisées. Il s'agit d'un cadre CSS de bas niveau, hautement personnalisable, qui offre tous les éléments nécessaires à la création de conceptions sur mesure.

L'aspect le plus attractif Tailwind est qu'il n'impose pas de spécifications de conception ou la façon dont un site devrait ressembler, il suffit de rassembler de minuscules composants pour construire une interface utilisateur qui est unique. Ce que fait Tailwind, c'est simplement prendre un fichier CSS "brut", traiter ce fichier CSS par le biais d'un fichier de configuration et produire une sortie.

IV.4.2. Serveur (API)

IV.4.2.a. Notion d'API

Une interface de programme d'application (API) [25] est un code qui permet à deux programmes de communiquer entre eux. Une API définit la manière correcte pour un développeur de demander des services à un système d'exploitation (OS) ou à une autre application, et d'exposer des données dans différents contextes et à travers plusieurs canaux.

Les API sont composées de deux éléments liés. Le premier est une spécification qui décrit comment les informations sont échangées entre les programmes, sous la forme d'une demande de traitement et d'un retour des données nécessaires. Le second est une interface logicielle écrite selon cette spécification et publiée d'une certaine manière pour être utilisée.

IV.4.2.b. Architecture REST

REST (Representational State Transfer) [26] est un style architectural pour le développement de services web. REST est populaire en raison de sa simplicité et du fait qu'il s'appuie sur les systèmes et les caractéristiques existants du protocole HTTP (Hypertext Transfer Protocol) de l'internet pour atteindre ses objectifs, plutôt que de créer de nouvelles normes, de nouveaux cadres et de nouvelles technologies.

IV.4.2.c. JavaScript et Node.Js

JavaScript [27] (souvent abrégé en JS) est un langage de script léger, orienté objet, principalement connu comme le langage de script des pages web. Mais il est aussi utilisé dans de nombreux environnements extérieurs aux navigateurs web tels que Node.js, Apache CouchDB voire Adobe Acrobat. Le code JavaScript est interprété ou compilé à la volée (JIT). C'est un langage à objets utilisant le concept de prototype, disposant d'un typage faible et dynamique qui permet de programmer suivant plusieurs paradigmes de programmation : fonctionnelle, impérative et orientée objet.

Node.js (Node) est un environnement d'exécution open source permettant d'exécuter du code JavaScript côté serveur. Node est utile pour développer des applications qui nécessitent une connexion persistante du navigateur au serveur et est souvent utilisé pour des applications en temps réel telles que le chat, les flux d'informations et les notifications push web.

Node.js [28] est destiné à fonctionner sur un serveur HTTP dédié et à employer un seul thread avec un seul processus à la fois. Les applications Node.js sont basées sur des événements et s'exécutent de manière asynchrone. Le code construit sur la plateforme Node ne suit pas le modèle traditionnel de réception, traitement, envoi, attente, réception. Au lieu de cela, Node traite les demandes entrant dans une pile d'événements constants et envoie de petites demandes les unes après les autres sans attendre les réponses.

IV.4.2.d. Express.JS

Express [29] est le Framework web Node.JS le plus populaire, et est la bibliothèque sous-jacente pour un certain nombre d'autres Frameworks web Node populaires. Il fournit des mécanismes pour :

- Écrire des gestionnaires pour les demandes avec différents verbes HTTP à différents chemins URL (routes).
- S'intégrer aux moteurs de rendu "view" afin de générer des réponses en insérant des données dans des modèles.
- Définir les paramètres communs des applications Web, comme le port à utiliser pour la connexion et l'emplacement des modèles utilisés pour le rendu de la réponse.
- Ajouter des "Middlewares" de traitement des demandes supplémentaires à n'importe quel point du pipeline de traitement des demandes.
- Bien que Express soit en soi assez minimaliste, les développeurs ont créé des paquets de Middlewares compatibles pour répondre à presque tous les problèmes de développement Web. Il existe des bibliothèques permettant de travailler avec des cookies, des sessions, des connexions d'utilisateurs, des paramètres d'URL, des données POST, des en-têtes de sécurité, et bien d'autres encore.

IV.4.2.e. MongoDB

MongoDB [30] est un programme de gestion de base de données NoSQL open source. NoSQL est utilisé comme une alternative aux bases de données relationnelles traditionnelles. Les bases de données NoSQL sont très utiles pour travailler avec de grands ensembles de

données distribuées. MongoDB est un outil qui permet de gérer des informations orientées documents, de stocker ou de récupérer des informations.

MongoDB prend en charge différentes formes de données. C'est l'une des nombreuses technologies de bases de données non relationnelles qui ont vu le jour au milieu des années 2000 sous la bannière NoSQL - normalement, pour être utilisée dans les applications big data et autres travaux de traitement impliquant des données qui ne s'intègrent pas bien dans un modèle relationnel rigide. Au lieu d'utiliser des tables et des lignes comme dans les bases de données relationnelles, l'architecture MongoDB est constituée de collections et de documents.

IV.4.3. Interfaces

Nous présenterons quelques interfaces parmi celles que nous avons réalisé lors du développement de l'application web.

IV.4.3.a. Page d'accueil

Ici nous allons dévoiler la première interface sur laquelle un client atterrit *(Figure IV-1, Figure IV-2)*. Celle-ci affiche toutes les informations concernant l'entreprise qui livre, dans notre cas l'entreprise fictive Delivo, avec une description l'entreprise en général, des services qu'elle propose et de la démarche à suivre pour pouvoir utiliser ces derniers. Elle affiche aussi, dans le pied de page, les informations de contact, une navigation rapide et un petit résumé utile pour l'aspect SEO (référencement naturel) du site.

Cette interface permet aussi à un utilisateur lambda disposant d'un numéro de suivi, de trouver et de suivre en temps réel son colis.

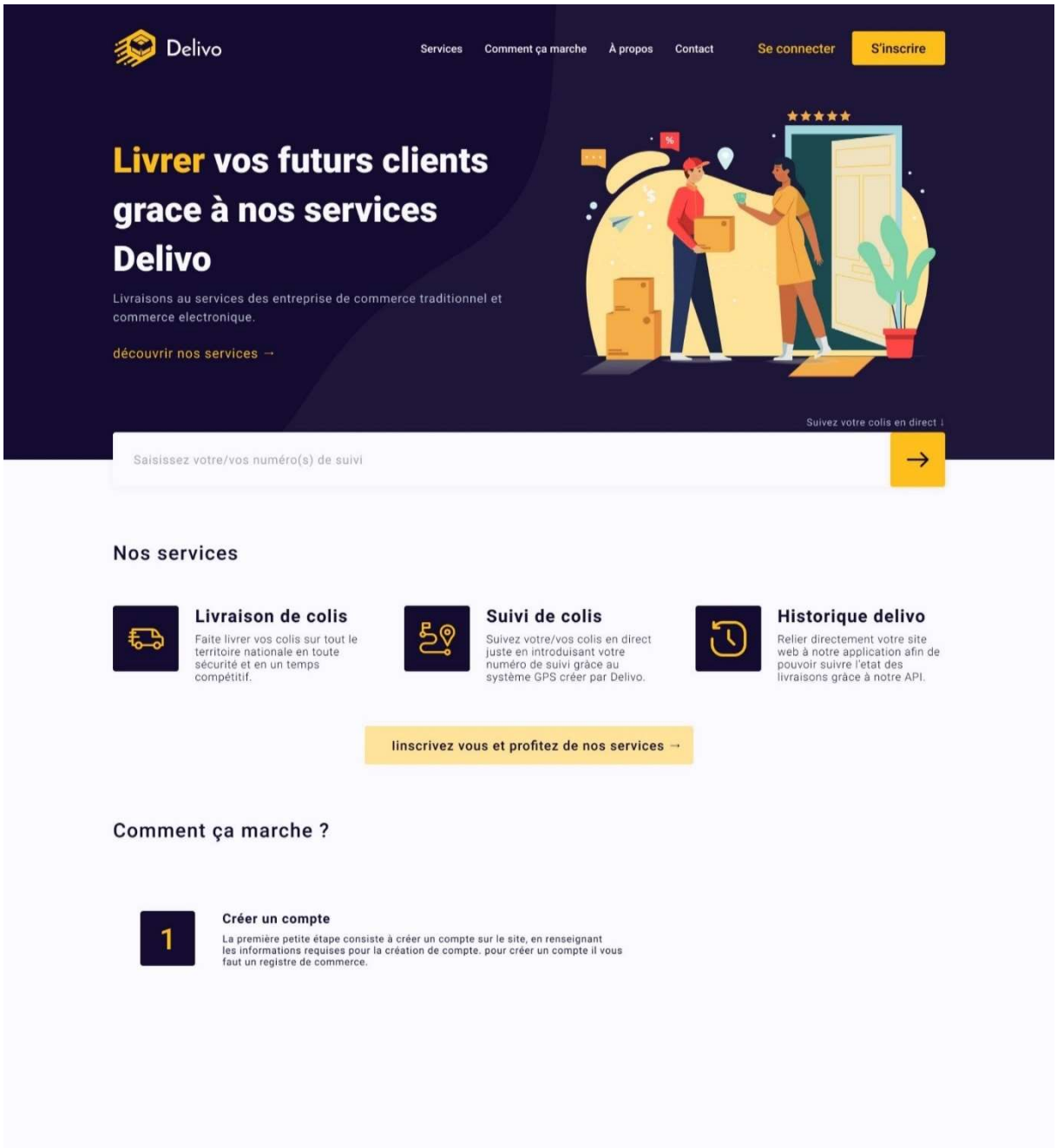

Figure IV-4 - Page d'accueil - partie 1

Figure IV-5 - Page d'accueil - partie 2

IV.4.3.b. Interface de connexion

Cette interface permet à un client de se connecter et d'avoir accès au back-office pour la gestion de ses colis. On y retrouve aussi les principaux liens de navigation de la page d'accueil, ainsi que le pied de page avec les pratiques utiles à l'aspect SEO.

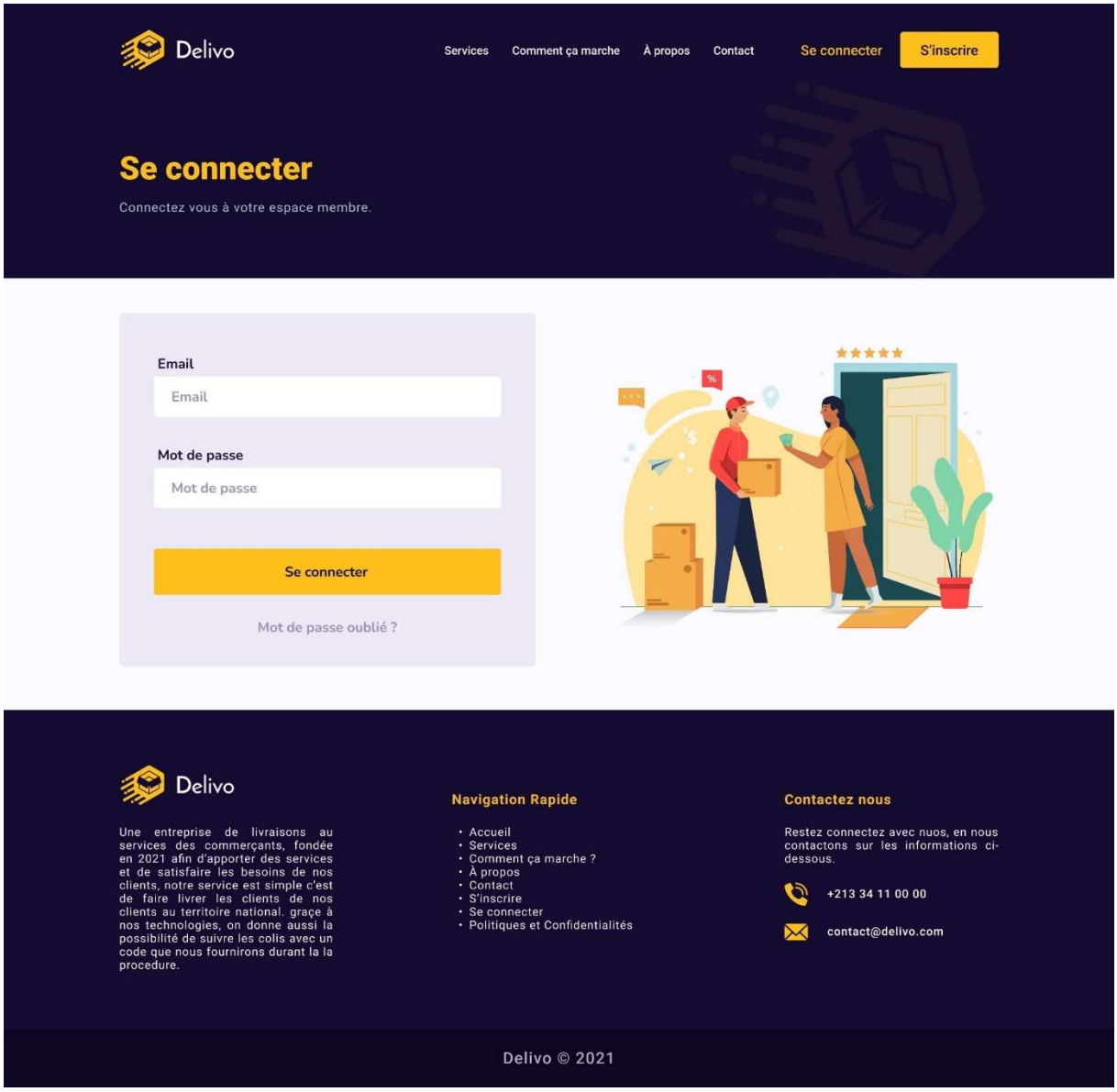

Figure IV-6 - Page connexion

IV.4.3.c. Tableau de bord

Il s'agit de la première interface du back-office, elle permet d'avoir une vue d'ensemble sur les services proposés par Delivo, ainsi que les statistiques et les différents états d'avancement sur ces services.

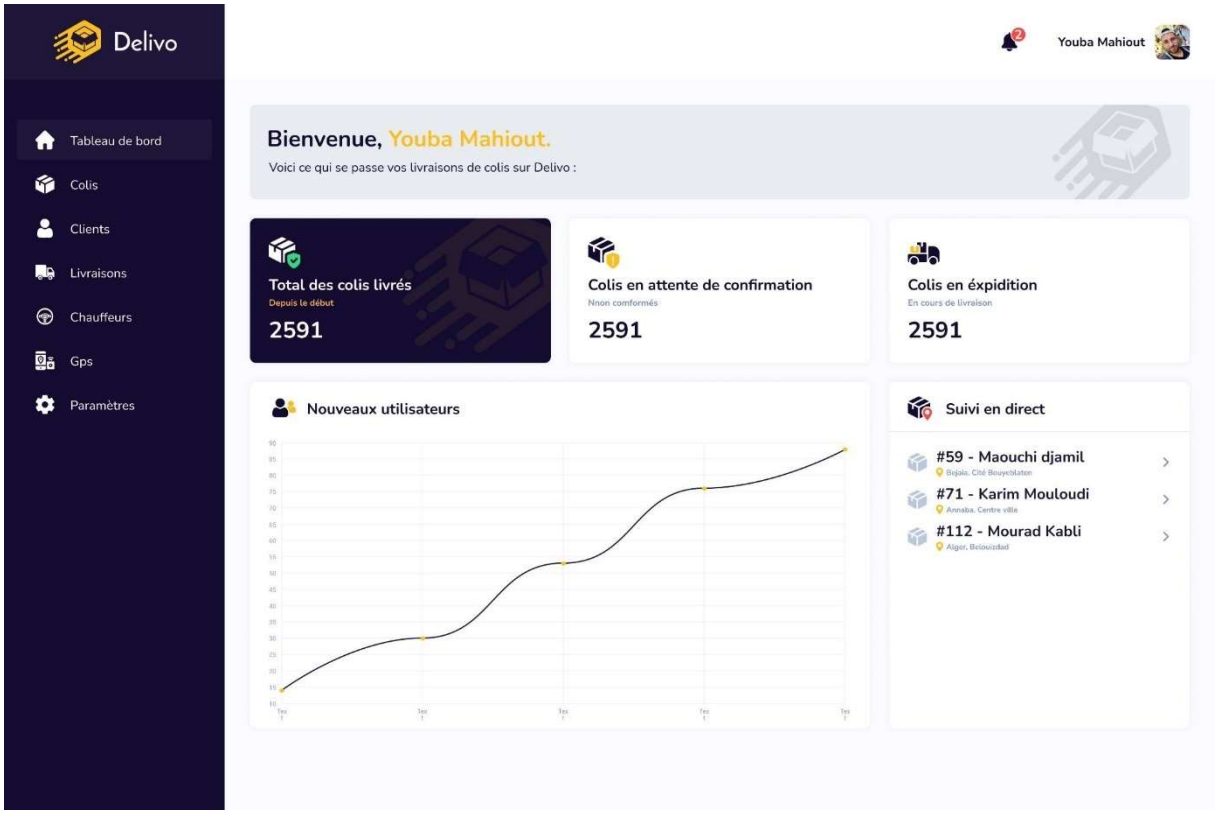

Figure IV-7 - Interface « Tableau de bord »

IV.4.3.d. Page gestion de livraisons

Cette page affiche les livraisons en cours et permet d'affecter une livraison à un chauffeur.

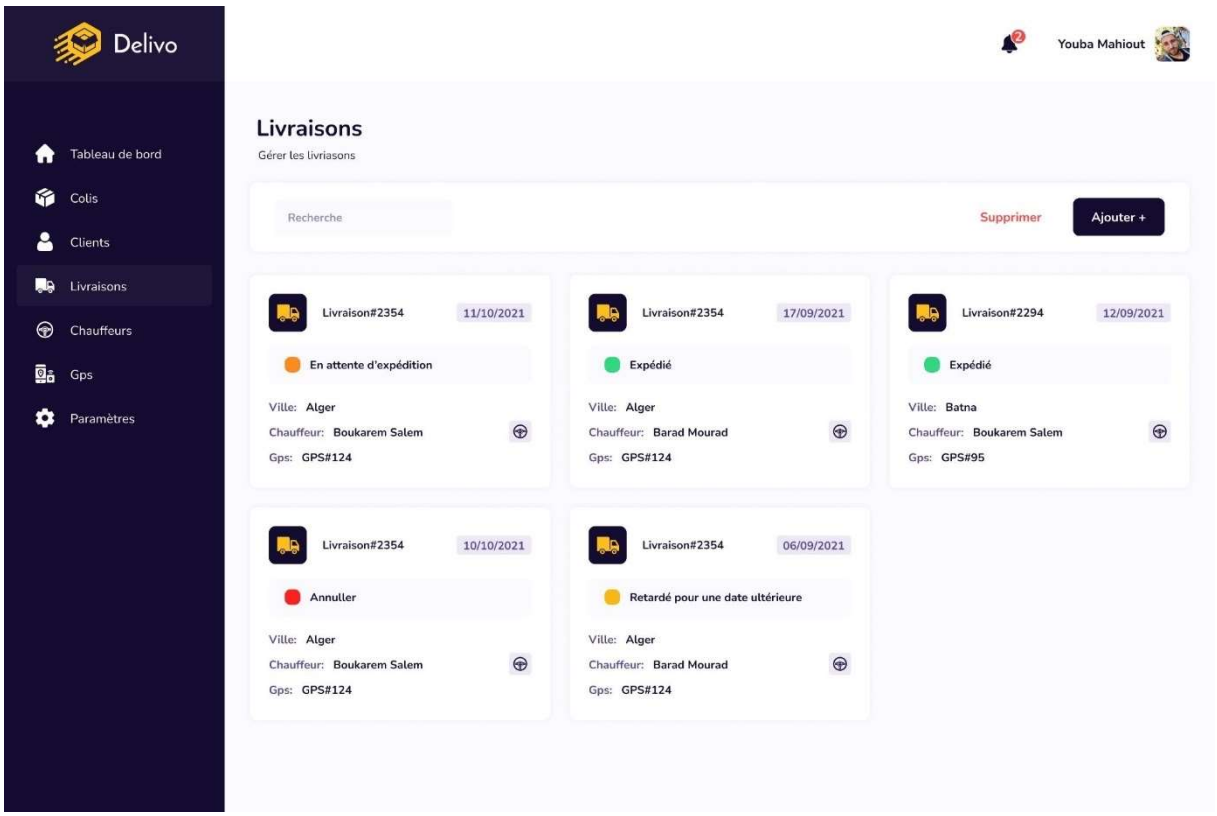

Figure IV-8 - Interface « gestion de livraisons »

IV.4.3.e. Page gestion des colis

Cette page représente le cœur du Dashboard c'est ici que l'on gère les données relatives au colis, on peut ajouter, modifier et supprimer des colis, comme on peut rechercher un ou plusieurs colis et afficher leurs détails.

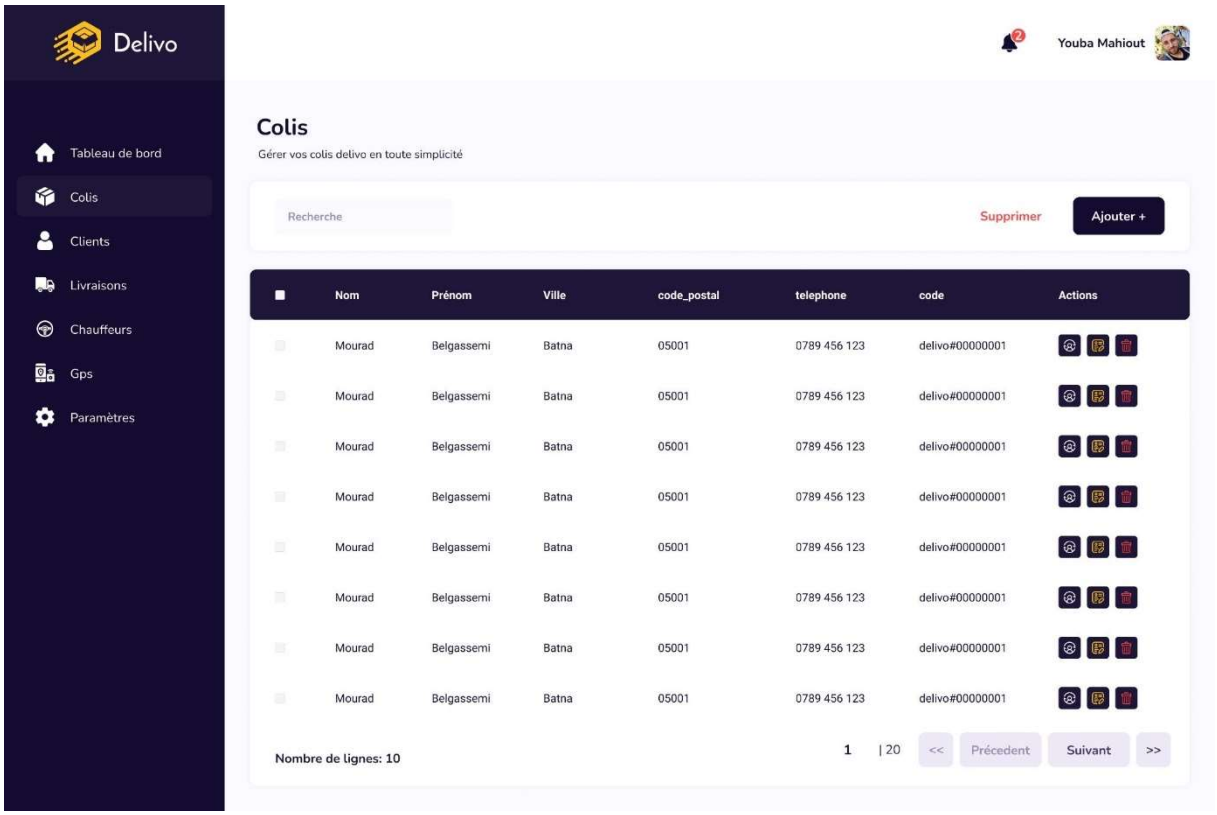

Figure IV-9 - Interface « Gestion des colis »

Lorsque l'on appuie sur le bouton ajouter le formulaire d'ajout de colis apparaît (Figure IV-7) ce formulaire permet de saisir toutes les données relatives au colis et génère automatiquement un QR-code qui sera, par la suite, imprimé sur le colis. Si l'on appuie sur le bouton modifier d'un des colis le formulaire de modification de colis apparaît, il s'agit d'un formulaire similaire à celui de la création, à la seule différence qu'il est déjà pré rempli avec les données du colis en question.

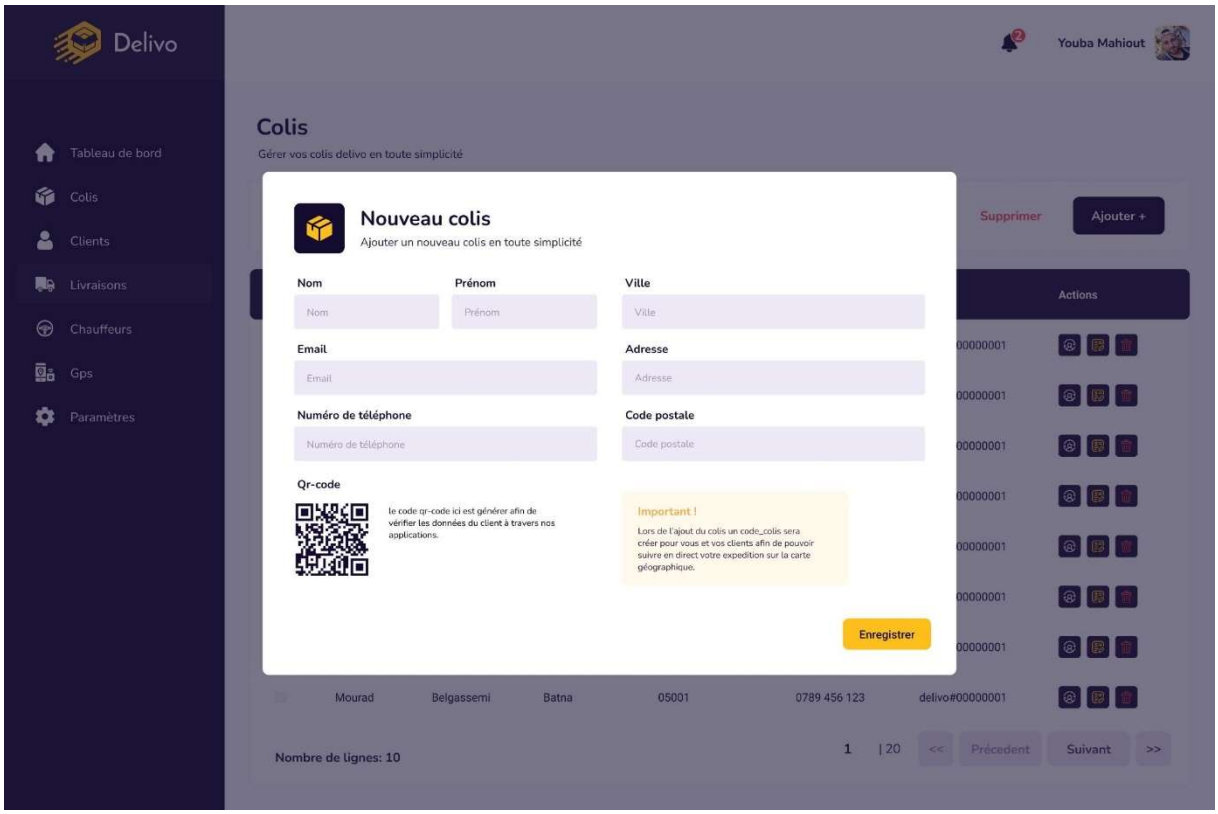

Figure IV-10 - Interface « Nouveau colis »

IV.5. Application mobile

Notre application mobile est quant à elle écrite en Dart, via le Framework cross plateforme Flutter. Cette application est réservée au chauffeur, elle lui permet de recevoir toutes les données dont il a besoin pour effectuer ses livraisons.

IV.5.1. Dart et Flutter

Dart est un langage optimisé pour le client, créé par Google, permettant de développer des applications rapides sur n'importe quelle plateforme. Son objectif est d'offrir le langage de programmation le plus productif pour le développement multi-plateforme, associé à une plateforme d'exécution flexible pour les Frameworks d'applications [31].

Flutter est quant à lui un Framework, lui aussi créé par Google, dont l'objectif principal est de créer des applications mobiles cross plateforme optimisées nativement. Récemment flutter permet aussi de créer des applications web et desktop multi-plateformes [32].

IV.5.2. Interfaces

Nous présenterons quelques interfaces parmi celles que nous avons réalisé lors du développement de l'application mobile.

IV.5.2.a. Interface de connexion

Cette interface est la première à accueillir le chauffeur lorsqu'il installe l'application.

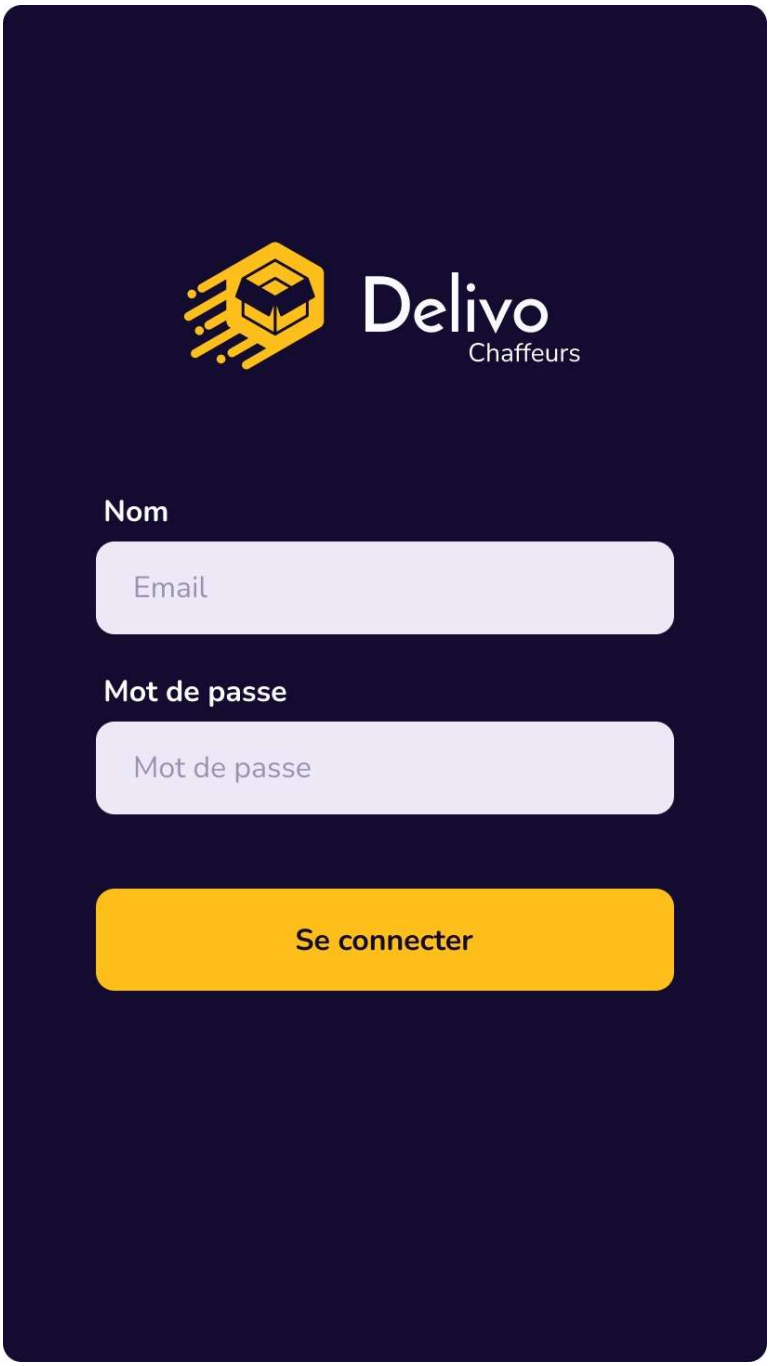

Figure IV-11 - Interface mobile « Connexion »

IV.5.2.b. Interface d'accueil (missions)

Quant à cette interface, elle sera celle que le chauffeur verra en premier à chaque fois qu'il ouvrira l'application après s'être connecté pour la première fois. Elle lui permet de voir quelles sont les livraisons qu'il doit effectuer.

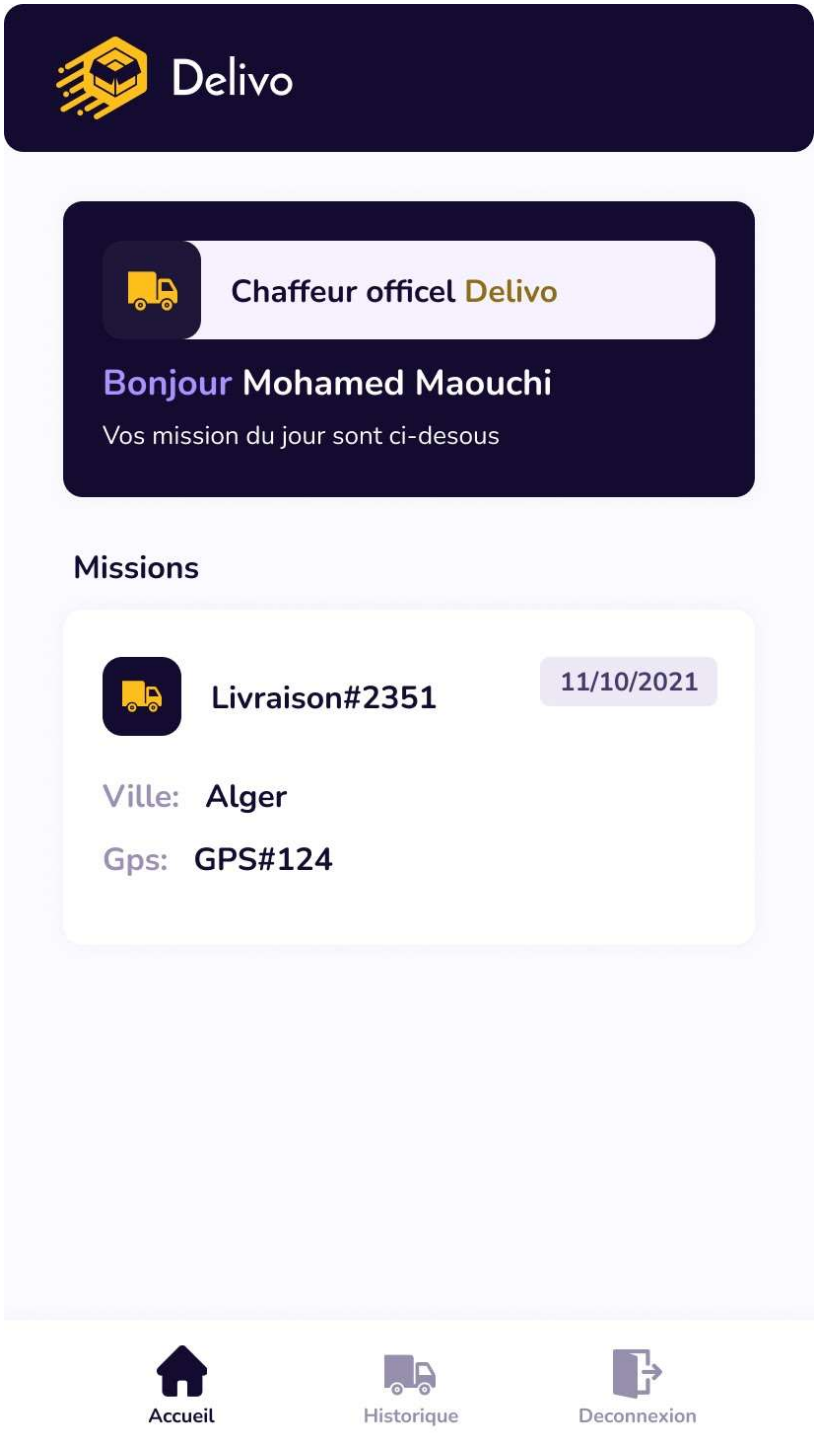

Figure IV-12 - Interface mobile « Accueil (Missions) »

IV.5.2.c. Interface livraison

Après avoir cliqué sur une livraison, le chauffeur aura devant lui l'interface livraison contenant les détails de la livraison et lui permettant de valider ou retarder la livraison d'un des colis de cette livraison.

Figure IV-13 - Interface mobile « Livraison détails »

IV.5.2.d. Interface historique des livraisons

Le chauffeur peut consulter un historique des livraisons qu'il a effectué.

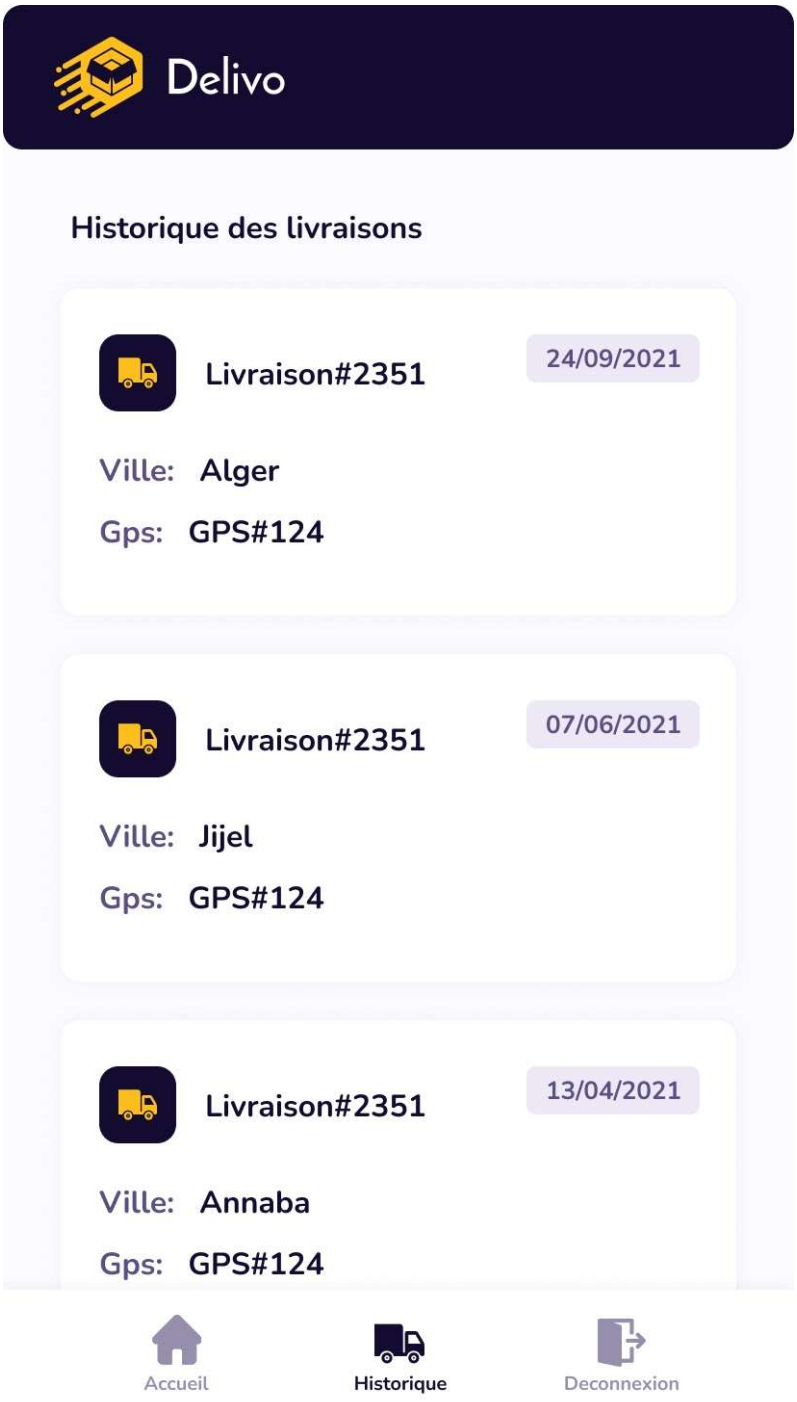

Figure IV-14 - Interface mobile « historiques des livraisons »

IV.6. Tests et déploiement

Lors du développement, nous avons soumis notre application à plusieurs tests automatiques que nous avons préalablement écrits grâce au framework de test Jest [37] et au générateur de données Faker [37], tout en tirant parti des fonctionnalités de gestion de projet offertes par le gestionnaire de version Git concernant l'automatisation des tests et du déploiement.

IV.6.1. Test

Les tests automatisés [33] sont l'application d'outils logiciels pour automatiser un processus manuel d'examen et de validation d'un produit logiciel. Les plans de test sont élaborés parallèlement au développement régulier des fonctionnalités de la feuille de route, puis exécutés automatiquement par les outils d'intégration continue des logiciels. Les tests automatisés favorisent la réduction de la taille de l'équipe d'assurance qualité et permettent à cette dernière de se concentrer sur des fonctionnalités plus sensibles.

IV.6.2. Déroulement des tests

Nous avons commencé par mettre en place les fixtures pour nos tests, dans le but de fixer certaines valeurs de notre environnement de test, à partir de là, nous avons écrit nos tests unitaires, en parallèle du développement, qui permettent de tester le comportement de chacun de nos modules. Après avoir réalisé une fonctionnalité entière et pratiqué nos tests unitaires sur chacun des modules de cette fonctionnalité, nous sommes passés aux tests d'intégrations de cette fonctionnalité qui nous permettent de vérifier si ces modules interagissent entre eux comme souhaité. Finalement lorsque nous avons assez de fonctionnalité pour réaliser une interface utilisable nous sommes passé aux tests dits end-to-end avec Puppeteer [37] qui nous permet de tester chaque entrée de notre application avec les fonctionnalités reliées à cette entrée.

Pour orchestrer le déroulement des tests ainsi que le déploiement nous avons écrit des Git Hooks [38] qui, avant chaque "commit", lancent les tests unitaires et ne valident ce commit que si tous les tests sont passés. Puis avant chaque "merge" avec la branche principale du projet, ce sont les tests d'intégrations qui doivent valider l'opération de fusion des branches. Quant au test end-to-end, ce sont eux qui ont la main sur le déploiement de la version en cours de développement de l'application, ils sont appelés juste avant l'opération de "push" vers le dépôt de production.

IV.6.3. Test unitaire

Le test unitaire [35] est défini comme un type de test de logiciel où les composants individuels d'un logiciel sont testés.

Dans le cas de notre modèle commerçant, nous dressons une liste d'assertions, que le Framework de test va vérifier à chaque itération, nous devons d'abord vérifier que l'objet créé avec des valeurs correctes définies dans nos fixtures soit validé sans erreur. Ensuite nous vérifions que pour chaque champ de l'objet, une erreur de validation est générée si ce champ est erroné, tel que le champ « nom » ne doit pas contenir de chiffres. Finalement nous testons les méthodes de notre modèle, dans notre cas si l'on consulte un commerçant le résultat ne doit pas contenir de mot de passe.

IV.6.4. Test d'intégration

Les tests d'intégration [35] sont le processus de test de l'intégration entre deux unités ou modules logiciels.

Dans le cas où nous testons la fonctionnalité d'authentification de notre API, nous dressons une liste d'assertions concernant les codes d'erreurs http que l'API doit retourner. Comme dans les tests unitaires nous vérifions tout d'abord le comportement de notre fonctionnalité lorsque les valeurs sont correctes, puis nous testons chaque valeur erronée. Dans notre cas l'API doit retourner le code 200 pour la connexion, 201 pour l'inscription, 400 en cas d'erreur de validation en précisant le champ erroné, et 401 si l'authentification échoue.

Le test échoue lorsqu'on reçoit un code inattendu, comme une erreur 500 dans l'authentification.

IV.6.5. Test end to end

Les tests end-to-end [36] reproduisent le comportement d'un utilisateur avec le logiciel dans un environnement applicatif complet.

Dans le test de l'authentification, l'interface d'inscription est générée et les formulaires sont remplis automatiquement et chaque bouton est testé, le but de ce test est de vérifier que chaque code d'erreur retournés par l'API, soit traité par l'interface, par exemple pour l'erreur 400 un message doit apparaitre pour préciser pour quoi l'inscription à échoué et quel champ du formulaire est invalide, et pour le code 201 l'interface doit pouvoir rediriger vers l'interface « Dashboard »

IV.7. Conclusion

Dans ce dernier chapitre, nous avons présenté l'environnement de développement et les logiciels utilisés ainsi que le module GPS et ses composants afin d'avoir une idée de comment le module fonctionne. Nous avons par la suite présenté tous les langages, librairies et Framework utilisés sur la partie web et mobile de l'application. Enfin nous avons clos le chapitre en présentant quelques tests que nous avons pu réaliser tout au long du développement de l'application.

Conclusion générale et perspectives

Le travail présenté est réalisé dans le cadre de projet de fin d'études en master génie logiciel, il consiste en la création et réalisation d'un système de livraison automatisé avec un tracking pour une entreprise de livraison au niveau national, ce système est constitué d'une application web pour les clients, une application mobile pour les chauffeurs et un GPS pour le tracking des colis.

Ce travail est décomposé en quatre étapes. La première a été consacrée à étudier le contexte général du projet à travers des détails et une problématique. L'étape suivante a été dévolue à la spécification et à l'analyse des besoins fonctionnels et non fonctionnels des utilisateurs, ce qui nous a permis de classer les fonctionnalités du système en plusieurs itérations selon la priorité et pouvoir identifier les acteurs du système et de réaliser les diagrammes de cas d'utilisations. Une fois les acteurs bien définis et le diagramme de cas d'utilisations complet nous sommes passé à la troisième étape, qui fut consacrée à la conception, nous avons pu réaliser les différents diagrammes, à savoir les diagrammes de séquence système, diagramme de classe et modèle documents. Enfin nous somme finalement arrivés à la dernière étape qui consisté à la réalisation des deux application web et mobile et nous avons pu utiliser différentes technologies et plateformes (Flutter, Dart, MongoDB, NodeJs, ExpressJS, ReactJs, Arduino, C, Tailwind CSS, etc.) pour implémenter notre solution que nous avons prototyper et maquetter tout en utilisant une charte graphique que nous avons créé pour illustrer l'identité visuelle reflétant le domaine d'activité.

La partie matérielle qui consiste à la réalisation du GPS a été réalisée au fur à mesure du projet et tout au long des étapes. Étant novices dans le domaine de l'électronique nous avons tout de même essayer de réaliser un GPS en assemblant quelques composants électronique (Arduino, GSM-SIM808, GPS-Antenne) et cela nous a permis l'apprentissage de certains points fondamentaux de ce domaine.

Après la mise en place de l'application, nous avons effectué plusieurs tests, ce qui nous a permis de mettre en évidence certaines lacunes de notre application, mais ces tests ont également ouvert des pistes d'amélioration des applications et des possibilités concernant nos perspectives, comme donner la possibilité aux clients d'utiliser des services, que l'on pourra mettre en place, afin de leur donner la possibilité d'intégrer le suivi directement sur leur sites, et même pourquoi pas ajouter une sécurité au GPS afin de notifier les administrateurs au cas où celui-ci sort du périmètre prédéfinit.

Ce travail nous a permis d'apprendre plus sur le domaine d'électronique et surtout d'avoir la perspective du monde professionnelle du travail et développer nos propres compétences en tant que développeur.

Références

[1] Sage Accounting and Business Management Software. https://www.sage.com/en-us

[2] Application Web - Définition et Explications. https://www.techno-science.net/glossaire-definition/Application-Web.html

[3] Qu'est-ce qu'une application Web ? Aperçu des formats. https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-quune-application-web-apercu-des-formats/

[4] Définition : Application mobile. https://www.wearecom.fr/dictionnaire/application-mobile/

[5] Cross-Platform Apps vs Native Apps vs Progressive Web Apps. https://www.businessofapps.com/insights/cross-platform-apps-vs-native-apps-vs-progressive-web-apps/?fbclid=IwAR1EDRmC2cBFP9FwDcMBahWZH-VuZ4Q_fDgUuVHnyvkcj1GP6aOd_PAmfJk

[6] Définition | GPS - Global Positioning System. https://www.futura-sciences.com/tech/definitions/technologie-gps-1897/

[7] Unified Process - an overview. https://www.sciencedirect.com/topics/computer-science/unified-process

[8] What is Extreme Programming (XP)?. www.agilealliance.org/glossary/xp/

[9] Kent Beck. *Extreme Programming Explained*. Addison Wesley, 2004.

[10] Qu'est-ce que le langage UML. https://www.lucidchart.com/pages/fr/langage-uml

[11] Joseph Gabay David Gabay. *UML 2 Analyse et Conception*. DUNOD, 2008.

[12] Besoins non fonctionnels / Non functional requirements. https://dantotsupm.com/2009/07/09/besoins-non-fonctionnels-non-functional-requirements/

[13] Maquette de site web : Définition. https://linkweb.fr/creation-site-internet-toulouse/maquette-site-web/

[14] : Pascal Roques. *UML 2 Modéliser une Application Web*. EYROLLES, 2008, 4ème édition.

[15] Diagramme de classe - UML SysML. https://www.uml-sysml.org/diagrammes-uml-et-sysml/diagramme-uml/diagramme-de-classe/

[16] NoSQL vs SQL Databases. https://www.mongodb.com/nosql-explained/nosql-vs-sql

[17] How to Design Schema for NoSQL Data Models https://www.mongodb.com/nosql-explained/data-modeling

[18] What is Figma? (And How to Use Figma for Beginners). https://www.theme-junkie.com/what-is-figma/

[19] Software – Arduino. https://www.arduino.cc/en/software

[20] Learn Git- Git tutorials, workflows and commands. https://www.atlassian.com/git

[21] Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/

[22] Introduction – Arduino. https://www.arduino.cc/en/Guide/Introduction

[23] React – A JavaScript library for building user interfaces. https://reactjs.org

[24] Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. https://tailwindcss.com/

[25] What Is an API (Application Program Interface)?. https://searchapparchitecture.techtarget.com/definition/application-program-interface-API

[26] What is REST (REpresentational State Transfer)?. https://searchapparchitecture.techtarget.com/definition/REST-REpresentational-State-Transfer

[27] JavaScript | MDN – Mozilla. https://developer.mozilla.org/fr/docs/Web/JavaScript

[28] What is Node.js? - Definition from WhatIs.com. https://whatis.techtarget.com/definition/Nodejs

[29] Express JS. https://expressjs.com/

[30] MongoDB: The most popular database for modern apps. https://www.mongodb.com/

[31] Dart programming language. https://dart.dev/

[32] Flutter: Beautiful native apps in record time. https://flutter.dev/

[33] Automated software testing for continuous delivery. https://www.atlassian.com/continuous-delivery/software-testing/automated-testing/

[34] Test-Fixtures. https://github.com/junit-team/junit4/wiki/test-fixtures

[35] Difference between Unit Testing and Integration Testing. https://www.geeksforgeeks.org/difference-between-unit-testing-and-integration-testing/

[36] The different types of testing in software. https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

[37] React end-to-end testing using Jest and Puppeteer. https://blog.logrocket.com/react-end-to-end-testing-jest-puppeteer/

[38] Git Hooks | Atlassian Git Tutorial https://www.atlassian.com/git/tutorials/git-hooks