

# Cahier des charges technique



Groupe Lead Dev

Amin MOHAMMED, Madani BENSIKHALED, Maxime LIDY, Brice BAYARD, Thibault COSATTINI

Glossaire	3
<b>Introduction</b>	<b>5</b>
But du document	5
Portée du document	5
Présentation du projet	5
Processus de validation	5
<b>Solutions techniques proposées</b>	<b>6</b>
Les avantages :	6
Facilité de maintenance et mise à jour :	6
Faille de sécurité moindre :	6
Disponibilité multi support et optimisation des ressources :	7
Les inconvénients :	7
Disponibilité hors ligne :	7
Conclusion :	7
<b>Spécifications techniques et contraintes</b>	<b>8</b>
API	8
Objectifs	8
Contraintes matériel	8
Framework :	9
Architecture :	9
A mettre en tableau	9
Authentification et gestion des rôles :	10
Authentification :	10
Gestion des rôles :	10
Accès aux données :	11
Documentation	11
Tests	11
Déploiement et livraison continue	12
Informations complémentaires	12
Base de données	13
Objectifs	13
Contraintes matériel	13
Sécurité et backups	13
SGBDR	13
Client Front - Interface utilisateur	14
Objectifs	14
Déploiement et livraison continue	14

# Glossaire

**API** : Une API (application programming interface ou « interface de programmation d'application ») est une interface logicielle qui permet de « connecter » un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités.

**Client léger** : Un client léger est un périphérique informatique de base qui exécute des services et des logiciels à partir d'un serveur centralisé.

**Client lourd** : Un client lourd est un logiciel qui propose des fonctionnalités complexes avec un traitement autonome. La notion de client s'entend dans une architecture client-serveur. Et contrairement au client léger, le client lourd ne dépend du serveur que pour l'échange des données dont il prend généralement en charge l'intégralité du traitement.

**URI** : **U**niform **R**esource **I**dentifier permet d'identifier les ressources abstraites ou physiques sur internet. Une URL est un type d'URI.

**JWT** : **J**SON **W**eb **T**oken est un standard ouvert défini dans la RFC 7519. Il permet l'échange sécurisé de jetons entre plusieurs parties. Cette sécurité de l'échange se traduit par la vérification de l'intégrité et de l'authenticité des données

**HTTP** : Protocole de transmission permettant à l'utilisateur d'accéder à des pages web par l'intermédiaire d'un navigateur.

**Intranet** : Un intranet est un réseau informatique privé utilisé par les employés d'une entreprise ou de toute autre entité organisationnelle et qui utilise les mêmes protocoles qu'Internet (TCP, IP, HTTP, SMTP, IMAP, etc.)

# Introduction

## Versions

Version	Date	Modification	Auteur
1.0	21 déc. 2...	Création du document	Equipe projet
1.1	22 déc. 2...	Modification suite réunion avec le P.O.	Equipe projet

## Approbations

Etales	Date	Description	Auteur
Validation	21 déc....	Présentation client	Equipe projet
Approbation	23 déc....	Validation par le client	Equipe projet

## But du document

Grâce à ce document, nous pouvons lister les différentes exigences et contraintes techniques liées au projet de refonte de fromagerie.com

Sont décrits, l'environnement technique dans sa globalité, les outils utilisés ainsi que sa faisabilité.

Pour chaque couche du projet un chapitre consacré permet de lister plus en détails les spécifications techniques qui lui sont propres.

## Portée du document

Ce document est destiné aux différents acteurs de ce projet, les concepteurs, encadrants techniques ainsi que le client final.

# Présentation du projet

Ce projet nous est confié par notre client, la PME fromagerie.com.

Le but est de proposer une refonte pour son application de gestion des cadeaux clients. Le principal objectif est de répondre aux problématiques rencontrées actuellement par l'utilisateur, qui sont décrites ci-après :

- Forte instabilité (bugs réguliers).
- Problèmes de maintenance.
- Faible possibilité d'évolution de développement.
- Manque de fluidité, d'accessibilité et de visibilité pour les utilisateurs.

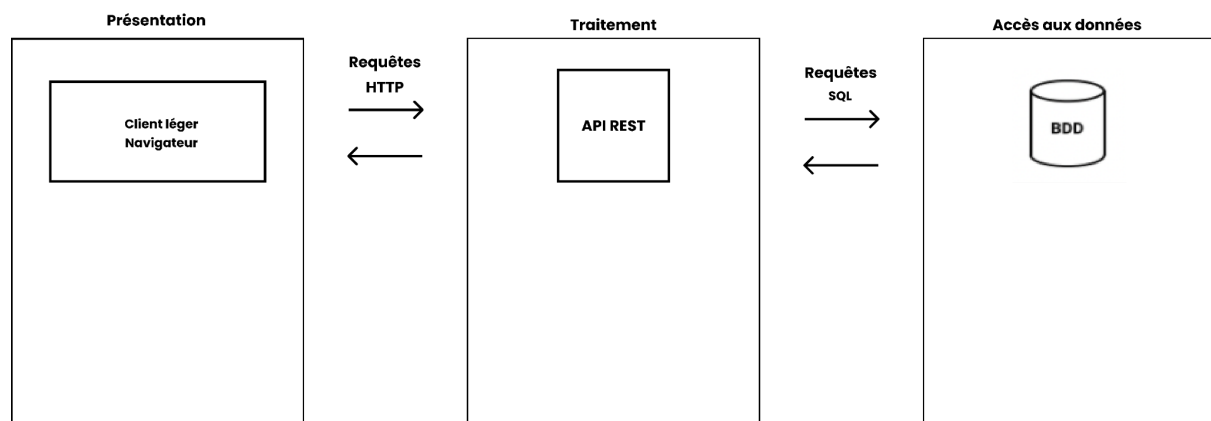
## Processus de validation

Avant toute communication vers l'extérieur la lecture et relecture de ce document sont assurées par le chef de projet et les responsables technique.

## Solutions techniques proposées

Pour répondre aux besoins du client, nous décidons de proposer un client léger réalisé grâce à une architecture en 3 tiers, client, serveur et base données.

C'est-à-dire que la solution sera intégralement hébergée sur un serveur et accessible aux utilisateurs depuis un navigateur.



## **Les avantages**

### **Facilité de maintenance et mise à jour**

Les mises à jour sont grandement simplifiées, en effet il suffit aux équipes de développement de déployer une nouvelle version afin que celle-ci soit immédiatement disponible et de façon totalement transparente pour l'utilisateur.

Lors de la remontée d'un bug, le temps de correction est réduit, de fait, les coûts de maintenance sont significativement réduits.

### **Faible de sécurité moindre**

L'application ainsi que les données sont centralisées sur un serveur, ainsi leur surveillance et leur administration sont grandement facilitées. De plus, la correction d'une faille existante est largement simplifiée. Pour finir, un client léger est uniquement disponible dans la dernière version possible, ce qui empêche l'utilisation de versions non corrigées.

### **Disponibilité multi support et optimisation des ressources**

Grâce à cette architecture, l'application est disponible sur différents supports, ordinateur, tablette, téléphone...

Contrairement à un client lourd, ici le calcul des ressources est déporté sur un serveur. Ainsi les utilisateurs ne sont pas dépendants du terminal qu'ils utilisent.

## **Conclusion**

Grâce à cette solution plus moderne que le modèle existant, nous pouvons proposer un coût de développement et de maintenance bien inférieur à celui d'un client lourd. Le produit est plus fréquemment mis à jour et donc plus stable.

# Spécifications techniques et contraintes

Nous abordons dans cette section les différentes spécifications et contraintes relatives à ce projet. Chaque couche du projet fait l'objet d'un chapitre qui lui est consacré.

## API

### Objectifs

Cette couche est l'interface permettant aux différents clients de communiquer avec la base de données. Elle doit pouvoir s'interfacer facilement et fournir les données demandées et ce, même à de nouveaux clients qui ne sont pas encore définis à ce stade. Cette couche est critique en termes de sécurité et de disponibilité, c'est pourquoi l'ensemble des besoins et contraintes sont décrits ci-après.

### Contraintes matériel

Notre back-end demande un serveur cloud.

Demandé dans les spécifications client, il sera infogéré et aux caractéristiques suivantes :

**OS** : Debian 11

**CPU** : 4 core

**Mémoire vive** : 8go

**Serveur web** : Nginx

Accès illimité à la base de données

300.000 requêtes HTTP mensuelle

Infogéré 24h/24

Redondance multi-site

*Notes :*

L'application est stateless, un **scaling horizontal** est donc possible dans le cadre ou, suite à une montée en charge, les ressources prévues initialement ne seraient plus suffisantes.

## Framework

Conformément aux demandes du client, le framework **Fast Api** doit être utilisé afin de concevoir l'api.

Les principaux avantages apportés par cette solution sont :

- Les performances, *ce framework est parmi les plus rapides disponibles.*
- La rapidité de développement.
- Le support standard de l'architecture REST et du format JSON.
- La documentation est auto générée grâce à Swagger.

## Architecture

L'architecture doit être réalisée grâce aux conventions définies par la norme REST.

- Les URI permettent d'identifier les ressources.
- Le verbe HTTP permet de cibler la méthode souhaitée.
- Les réponses retournent les objets complets.
- Un jeton JWT permet de l'authentification de l'utilisateur

Exemple pour la gestion du conditionnement :

Verbe HTTP	URI	Description
GET	/conditionnement	<i>Récupérer l'ensemble des conditionnements</i>
GET	/conditionnement/{id}	<i>Récupérer le conditionnement spécifié par l'id.</i>
POST	/conditionnement	<i>Créer un nouveau conditionnement</i>
PUT	/conditionnement/{id}	<i>Modifier le conditionnement spécifié par l'id.</i>
DELETE	/conditionnement/{id}	<i>Supprimer le conditionnement spécifié par l'id.</i>

L'ensemble des points d'entrée sont définis dans la documentation.



## Authentification et gestion des rôles :

### Authentification

Librairie à utiliser : **Jose JWT**

Les différentes méthodes d'accès aux données doivent être protégées, seul l'utilisateur enregistré peut y avoir accès.

Le endpoint /login, permet d'accéder à la méthode d'authentification, l'utilisateur doit utiliser sa combinaison nom d'utilisateur et mot de passe, si corrects, il reçoit en réponse un Json web token d'une durée de 30 mins.

Pour accéder aux ressources protégées, ce token doit être passé en header de la requête HTTP.

Grâce à la lecture de ce token, l'application peut déterminer de quel utilisateur il s'agit. Cela à pour but de récupérer les informations associées à l'utilisateur et uniquement celles-ci.

C'est un middleware qui permet cette tâche.

### Gestion des rôles

En fonction de son rôle l'utilisateur à accès certaines ressources et pas à d'autres, c'est l'autorisation.

Comme pour la partie authentification c'est un middleware qui permet cette tâche.

Les rôles sont les suivants :

Rôle	Permissions
ROLE_ADMIN	Administrateur du système, toutes les méthodes lui sont accessibles
ROLE_OS	Opérateur stock il accède uniquement aux méthodes du domaine "gestionStock", <u>voir annexe 13</u>
ROLE_OC	Opérateur colis, il accède uniquement aux méthodes du domaine "gestionColis", <u>voir annexe 13.</u>

## Accès aux données

*Librairie à utiliser : **Django ORM***

Afin de pouvoir accéder aux données, l'ORM Django doit être utilisé. Celui-ci permet de mapper les entités objets en base de données et ainsi les manipuler dans un sens bidirectionnel, c'est-à-dire :

Requêter les données dont nous avons besoin, puis hydrater notre modèle objet.

Assurer la persistance en base de données.

L'avantage est de pouvoir se protéger des potentielles failles rendant possibles les injections SQL.

## Documentation

*Librairie à utiliser : **Swagger***

Les différents endpoint doivent faire l'objet d'une documentation, autogénérée par SWAGGER celle-ci liste :

- Les points d'accès (endpoints)
- Les formats de données attendues
- Les réponses de succès possibles
- Les réponses d'erreurs possible

## Tests

*Librairie à utiliser : **FastApi Test, HTTPX***

Des tests unitaires complets devront être réalisés. Ils doivent tester les différents endpoints ainsi que les méthodes associées.

La série de tests doit être exécutée avec succès à chaque intervention. De plus, lors du développement d'une nouvelle fonctionnalité, celle-ci doit faire l'objet de nouveaux tests afin de s'assurer de la non régression du projet.

## Sécurité

Nous recommandons d'utiliser un protocole de chiffrement TLS afin de chiffrer les requêtes.

Afin de se protéger des attaques bruteforce un quota d'utilisation par adresse IP doit être mis en place, au-delà de cette limite les requêtes retournent une erreur de type 429.

Les détails plus précis des recommandations de sécurité seront à définir lors d'une réunion ultérieure entre le client et l'équipe chargée de la sécurité.

## Déploiement et livraison continue

Technologie utilisée : Gitlab

Un processus d'automatisation du déploiement doit être mis en place grâce à Gitlab, celui-ci permet lors d'un push, de déployer la branche main du projet.

A chaque nouveau déploiement, une série de test d'intégration doivent être exécutés afin de s'assurer

## Versions utilisées

Technologie	Version
FastApi	0.88
Django	4.0
Swagger	3.0
Jose JWT	4.0

## Informations complémentaires

Le repository officiel est disponible à cette adresse : <https://github.com/tiangolo/fastapi>

La documentation officielle : <https://fastapi.tiangolo.com/>

# Base de données

## Objectifs

La base de données permet la persistance des données du système.

Dans le cadre où les informations personnelles des clients y sont stockées celle-ci doit être conforme aux règles RGPD en vigueur, voir le site de la CNIL :

<https://www.cnil.fr/reglement-europeen-protection-donnees>

Celle-ci doit être hautement disponible et sécurisée, les différents besoins et contraintes sont décrits ci-après.

## Contraintes matériel

RAM évolutive de 512 Mo à 4 Go

Stockage SSD

Sauvegardes automatiques journalières

## Sécurité et backups

Deux sauvegardes journalières doivent être réalisées afin de s'assurer de ne perdre aucune donnée en cas d'erreur ou d'intrusion sur le système.

Ces backups doivent être stockées sur un serveur différent de celui utilisé par la base de données en cas de non accessibilité de ce dernier.

Les procédures d'automatisation de ces tâches sont prises en charge.

## SGBDR

Mysql est recommandé comme serveur de base de données pour plusieurs raisons :

Il est open source.

Il possède de bonnes performances.

Il est multi-utilisateur.

## Versions utilisées

Technologie	Version
Mysql	5.7
InnoDB	5.6

# Client Front - Interface utilisateur

## Objectifs

Conformément aux demandes du client, l'objectif est de fournir une interface graphique moderne aux utilisateurs.

Réalisée grâce à la librairie ReactJS, celle-ci permet la création d'une SPA et une navigation sans rafraîchissement de la page.

Les utilisateurs peuvent se connecter et accéder à leurs espaces. En fonction de leur rôle, ils retrouvent un espace qui leur est propre.

## Déploiement et livraison continue

Technologie utilisée : **Jenkins**

Un processus d'automatisation du déploiement doit être mis en place grâce à Jenkins, celui-ci permet lors d'un push, de déployer la branche main du projet.

A chaque nouveau déploiement, une série de tests d'intégration doivent être exécutés afin de s'assurer.

## Versions utilisées

Technologie	Version
ReactJS	17.0
Jenkins	2.3

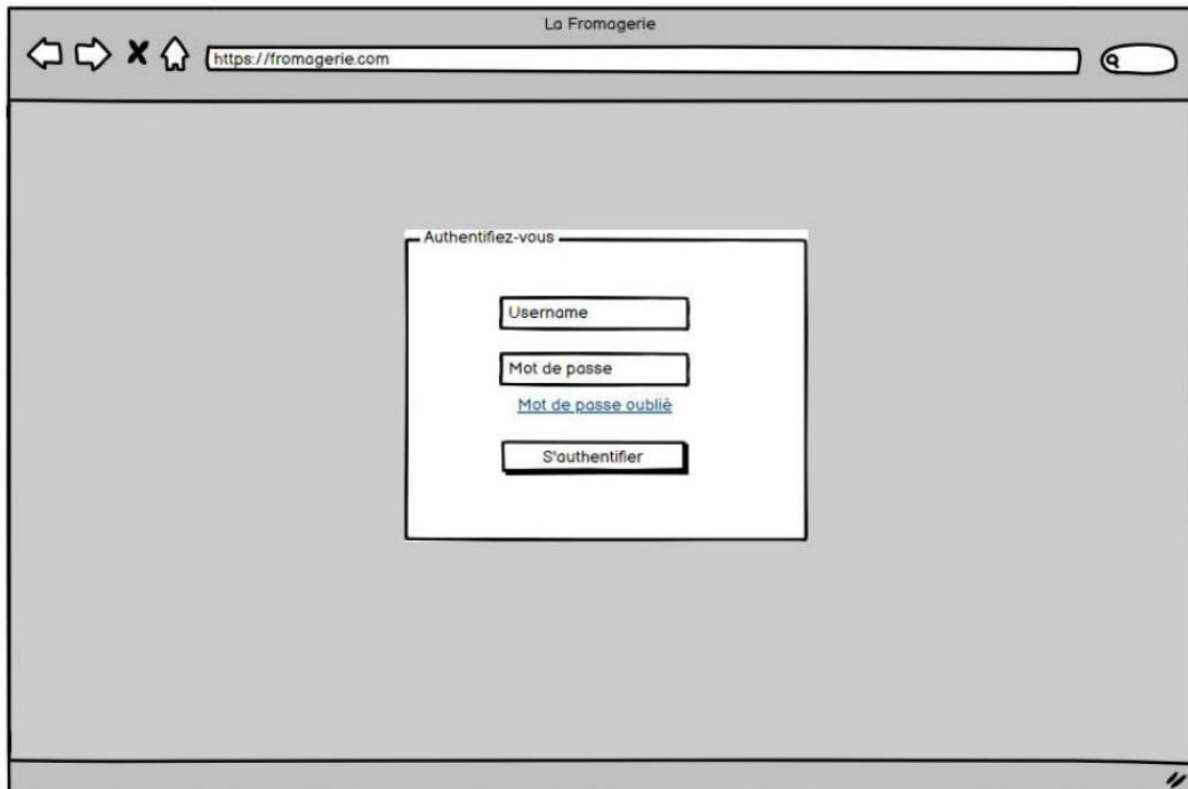
## Informations complémentaires

Le repository officiel est disponible à cette adresse : <https://github.com/facebook/react/>

La documentation officielle : <https://fr.reactjs.org/docs/getting-started.html>

## Annexe 1

### Maquette page authentification



La Fromagerie

https://fromagerie.com

Authentifiez-vous

Username

Mot de passe

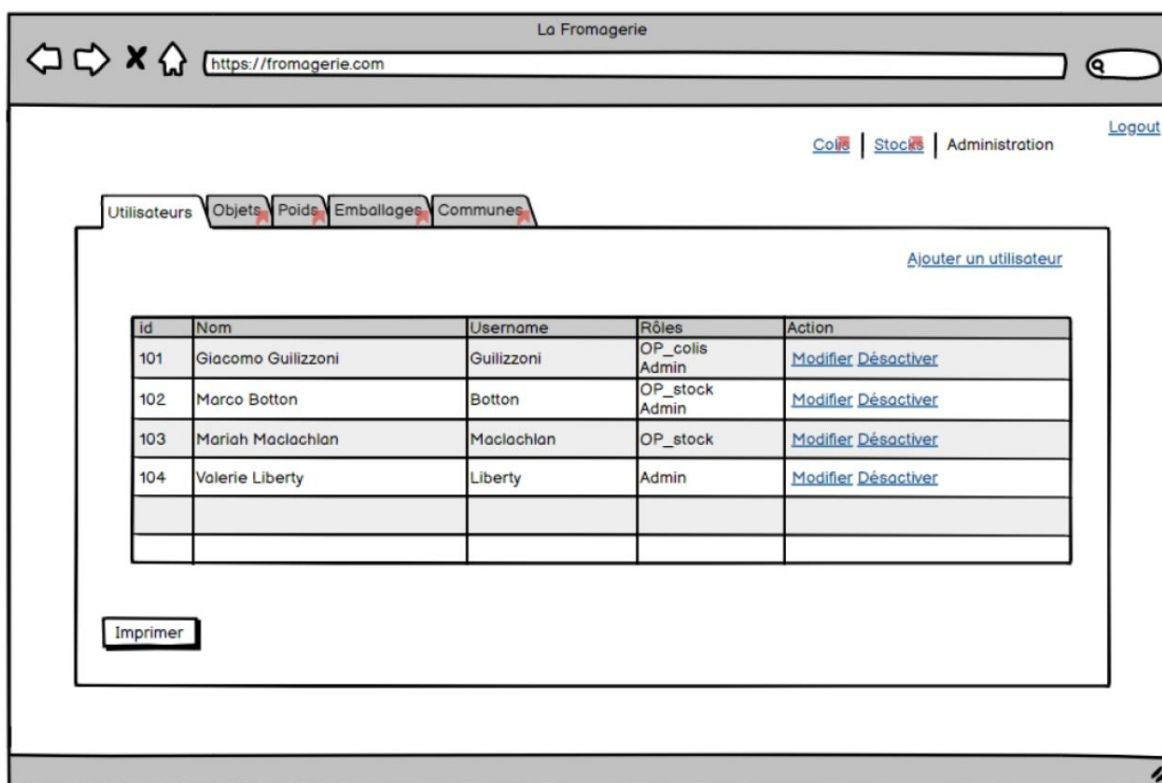
[Mot de passe oublié](#)

S'authentifier

The image is a wireframe of a web browser window. The browser's title bar says 'La Fromagerie'. The address bar shows 'https://fromagerie.com'. The main content area is a light gray rectangle. In the center of this area is a white rectangular box. Inside this box, at the top, is the text 'Authentifiez-vous'. Below this text are four elements: a text input field labeled 'Username', another text input field labeled 'Mot de passe', a blue underlined link labeled 'Mot de passe oublié', and a button labeled 'S'authentifier'. The browser window has standard navigation icons (back, forward, stop, home) on the left and a search icon on the right.

## Annexe 2

### Maquette gestion des utilisateur





## Annexe 3

### Maquette gestion des objets

La Fromagerie

https://fromagerie.com

Colis | Stoc | Administration [Logout](#)

Utilisateurs | **Objets** | Poids | Emballages | Communes

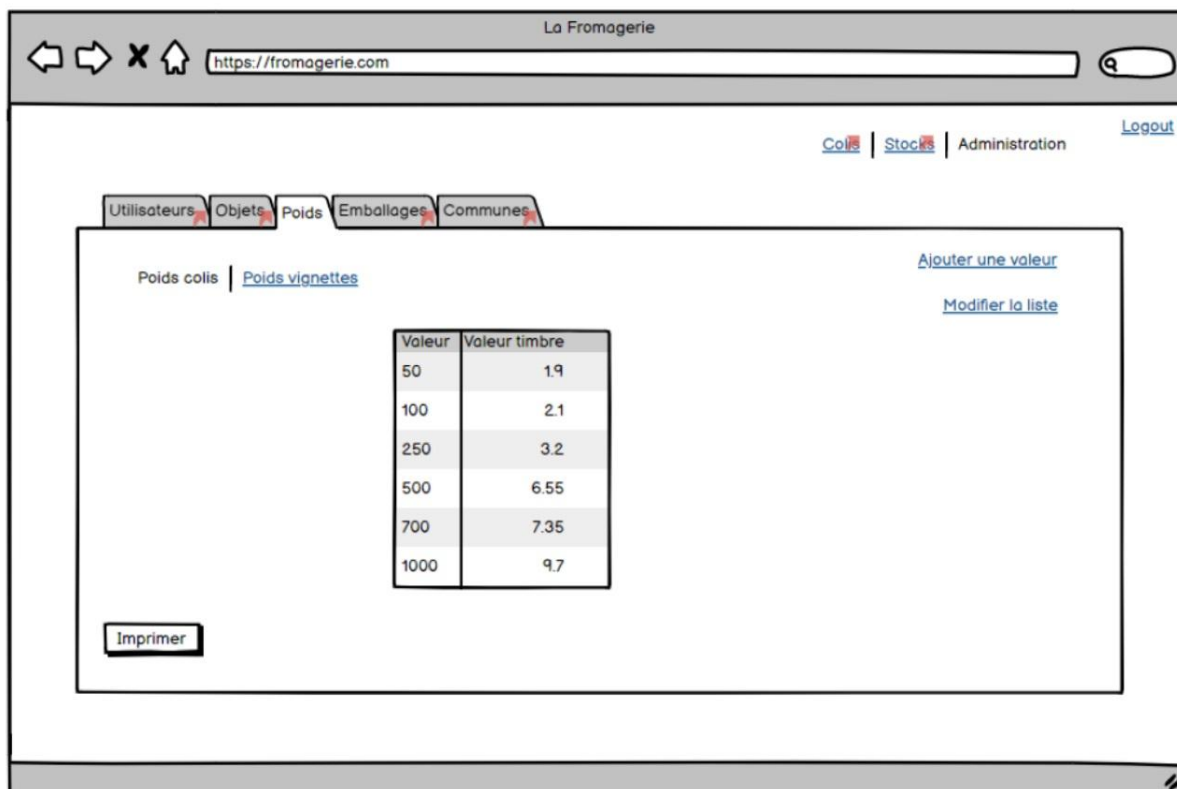
[Ajouter un objet](#)

Id	Désignation	Modèle	Prix Unitaire	Poids	Ordre IMP	Publicitaire	Conditionnement	Points	Action
101	Appareil photo	none	50€	115	1	<input checked="" type="checkbox"/>	Enveloppe Appareil photo	20	<a href="#">Modifier</a> <a href="#">Désactiver</a>
34	Autocolant	none	0.50€	5	0	<input type="checkbox"/>	Distingo 50g	25	<a href="#">Modifier</a> <a href="#">Désactiver</a>
103	Casquette	Orange	10€	70	1	<input type="checkbox"/>	Distingo 50g	30	<a href="#">Modifier</a> <a href="#">Désactiver</a>
104	Collecteur	none	50€	10	1	<input checked="" type="checkbox"/>	Courrier	0	<a href="#">Modifier</a> <a href="#">Désactiver</a>

Imprimer

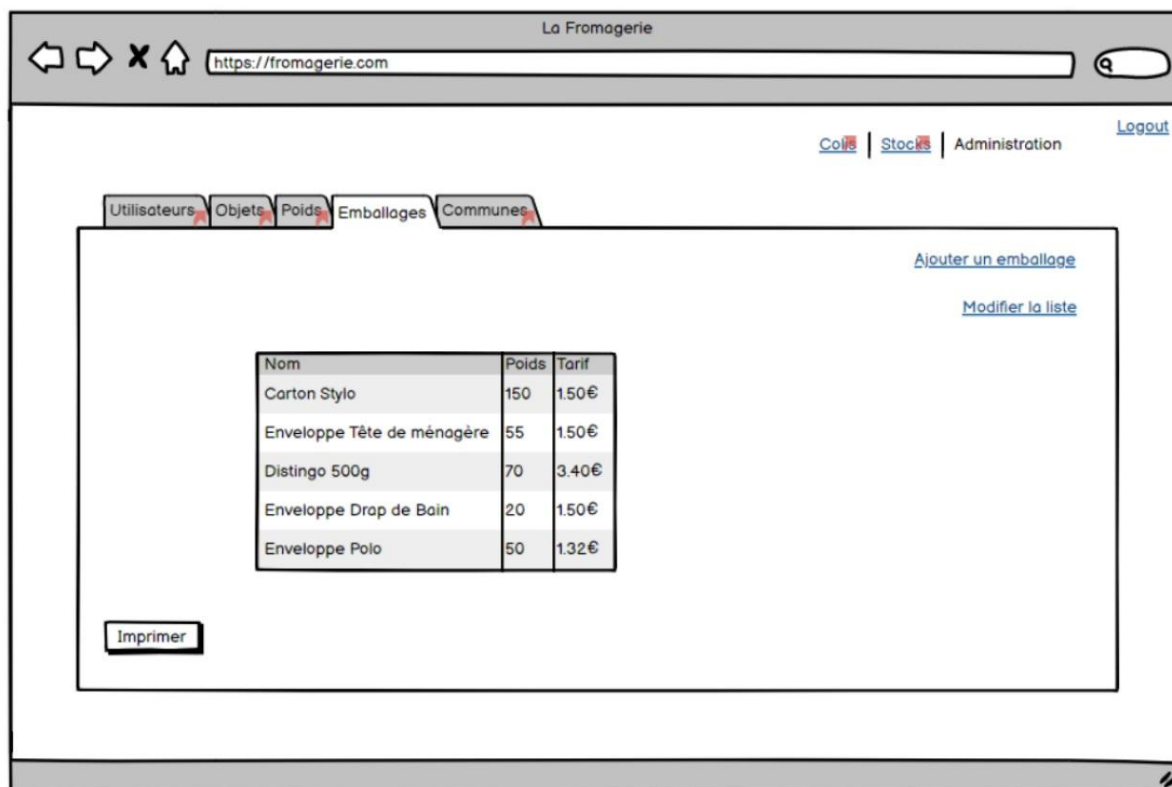
## Annexe 4

### Maquette gestion des poids



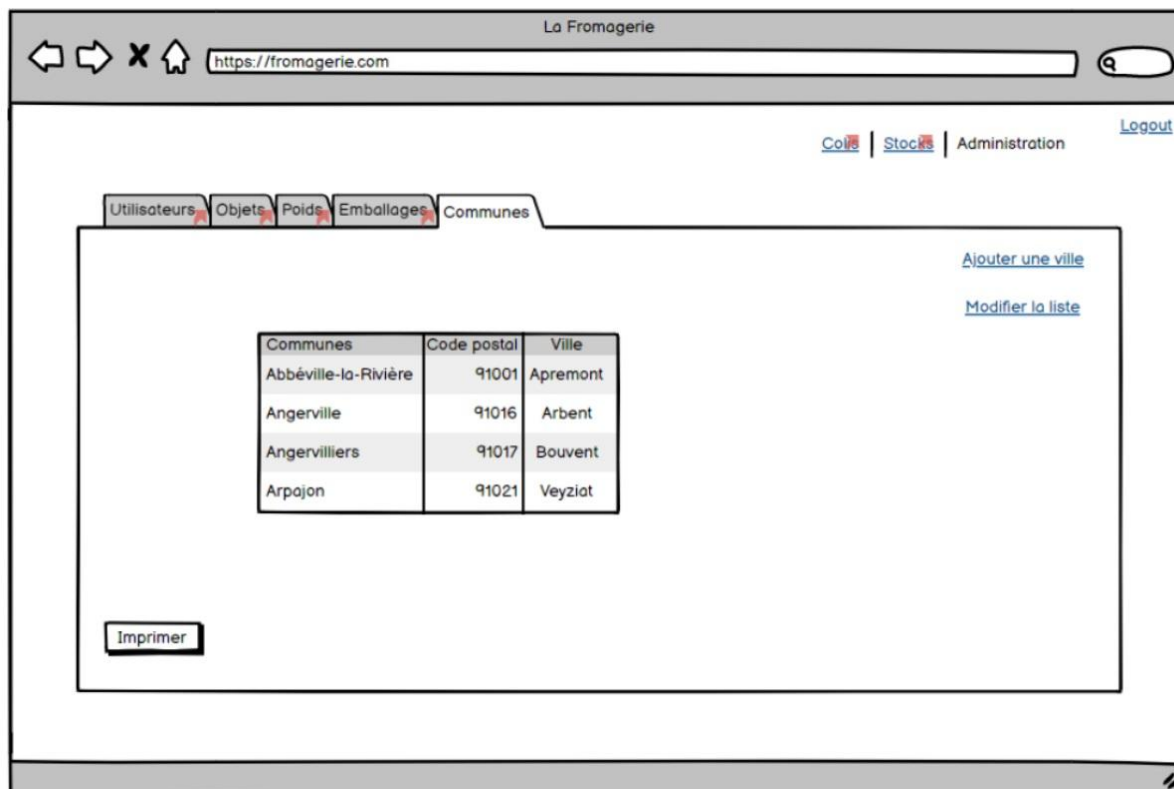
## Annexe 5

### Maquette gestion des emballages



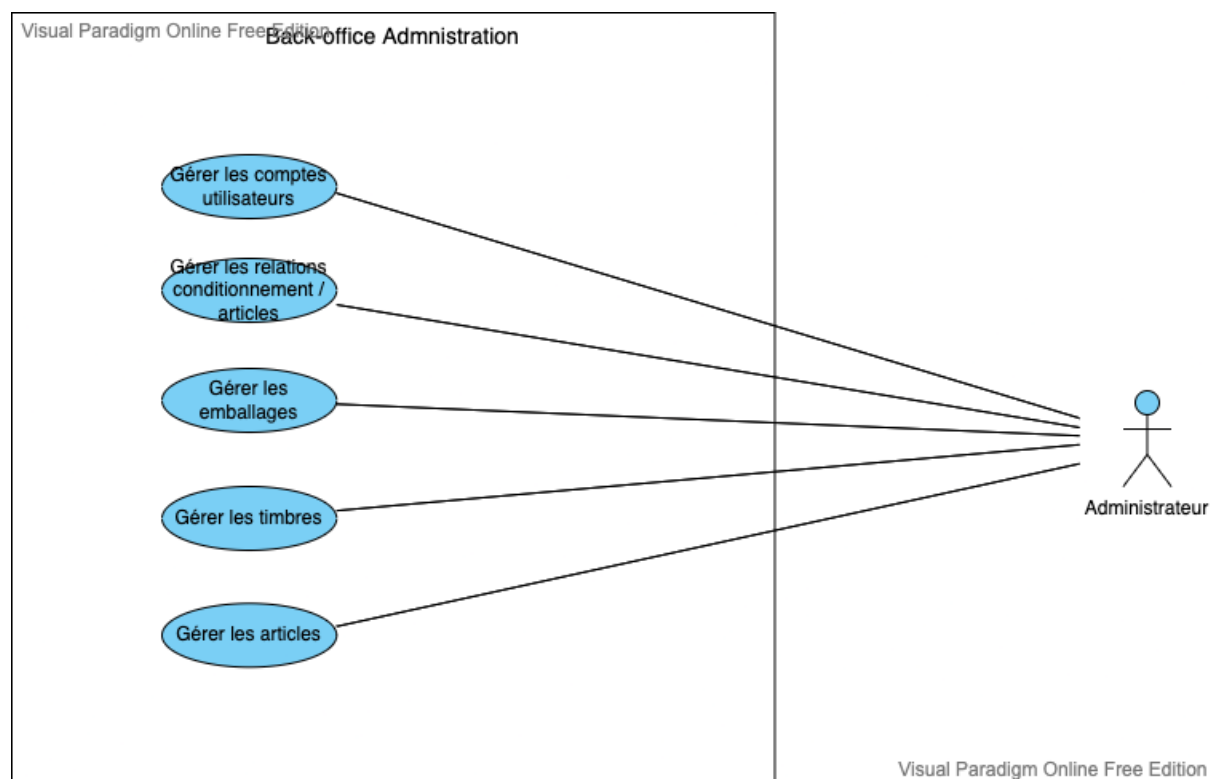
## Annexe 6

### Maquette gestion des communes



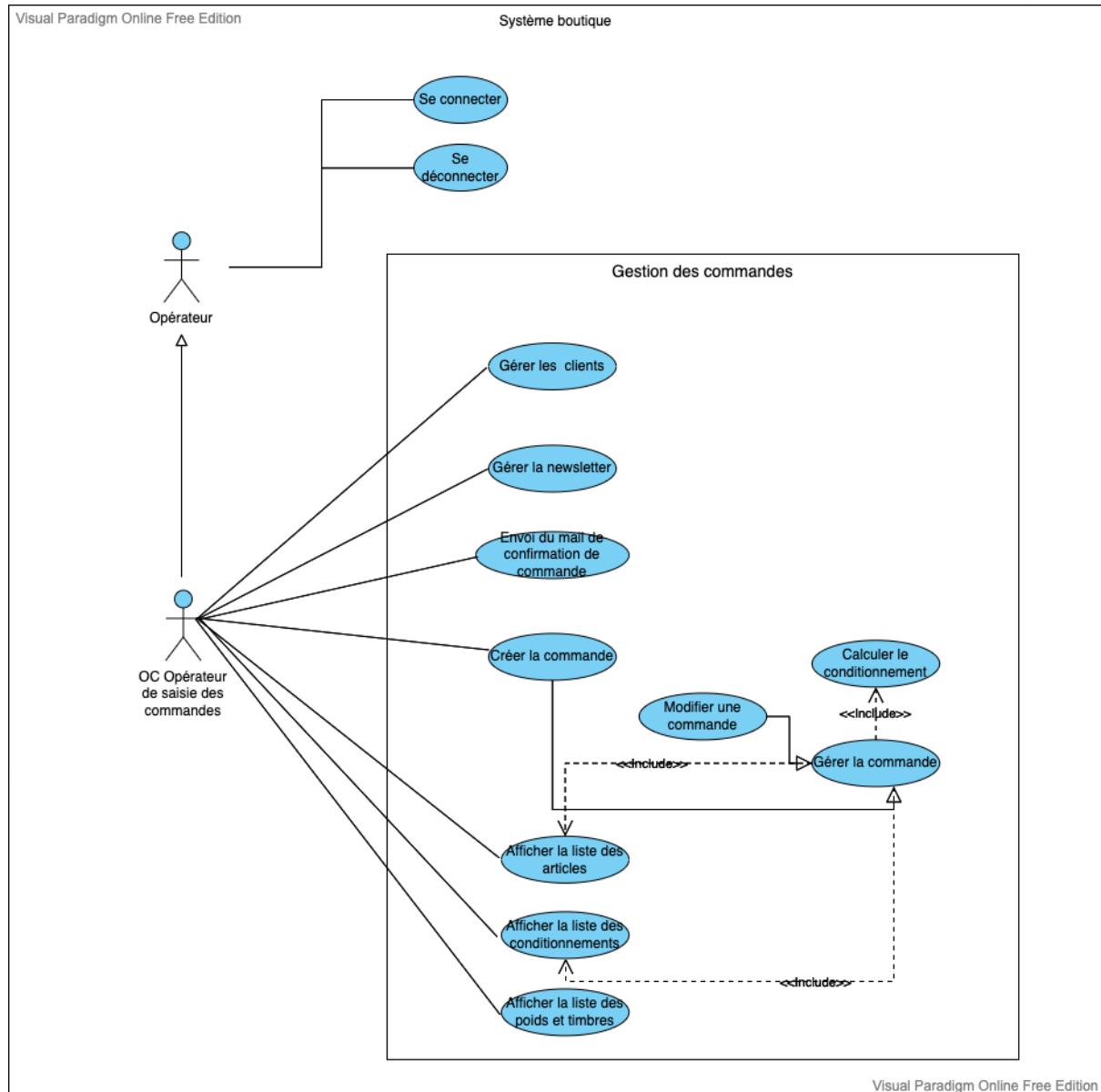
## Annexe 7

### Diagramme de cas d'utilisation back office



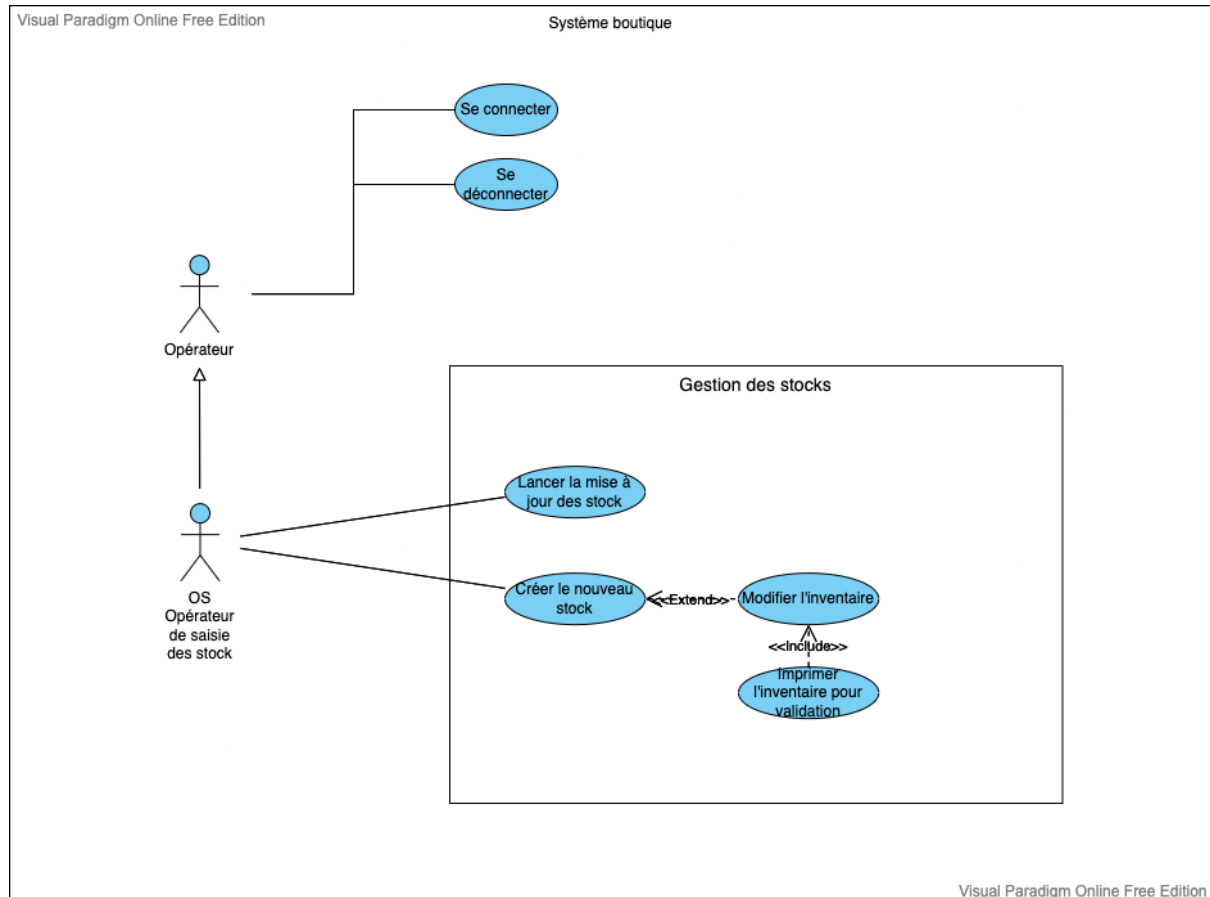
## Annexe 8

### Diagramme de cas d'utilisation gestion des commandes



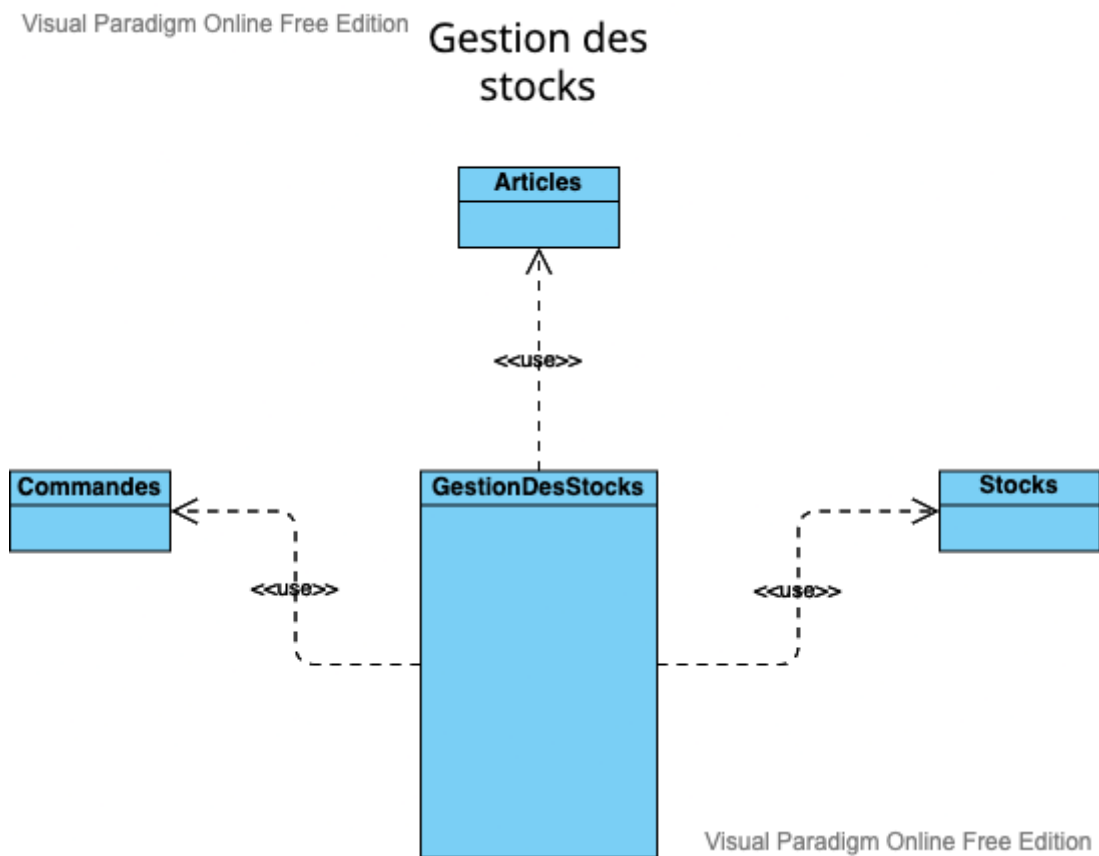
## Annexe 9

### Diagramme de cas d'utilisation gestion des stocks



## Annexe 10

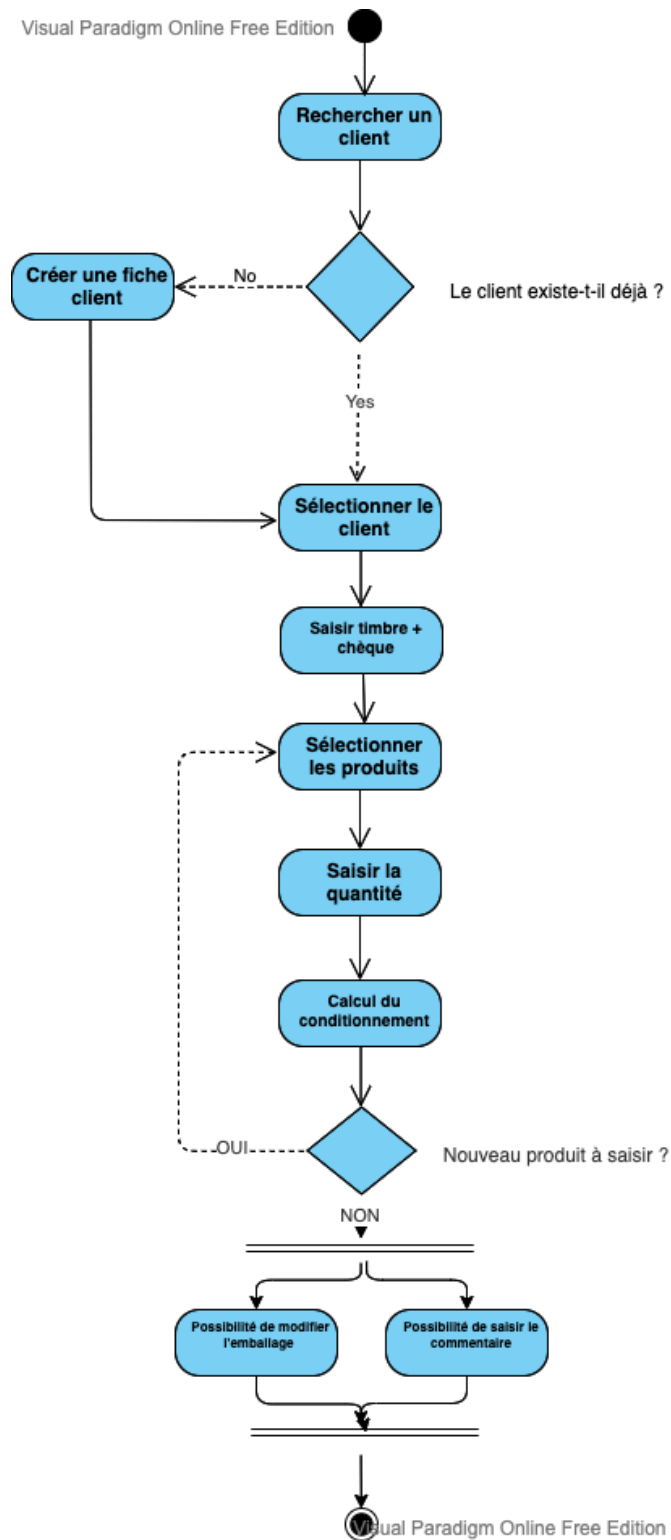
### Diagramme de classe gestion des stocks





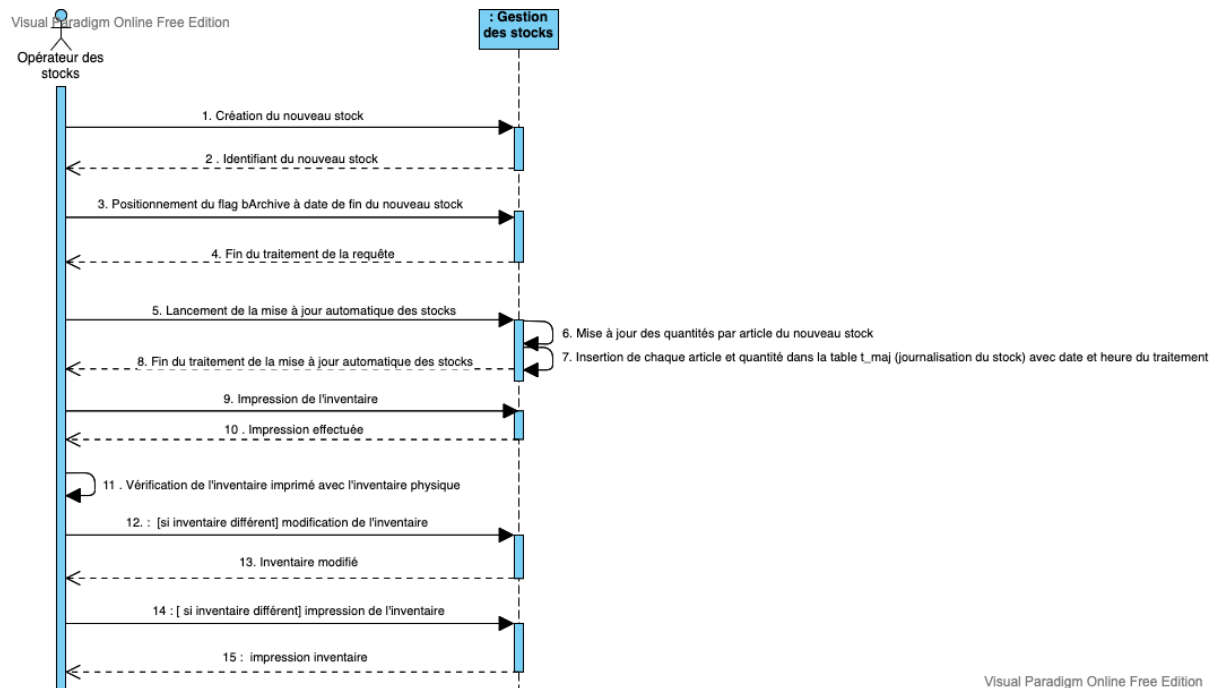
## Annexe 11

### Diagramme d'activité création d'un nouveau client



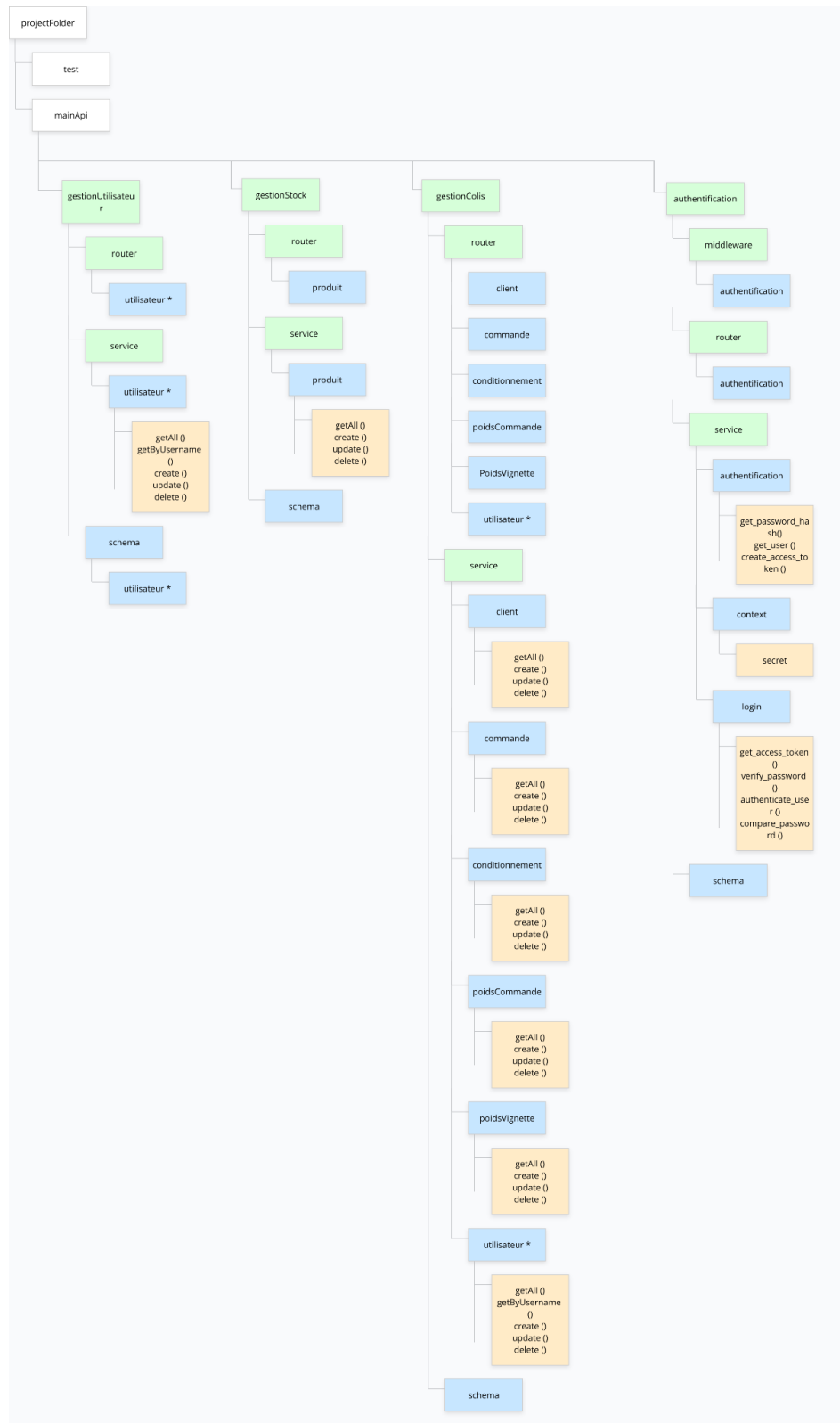
## Annexe 12

### Diagramme de séquence création d'un nouveau stock



## Annexe 13

### Architecture de l'API



## Annexe 14

### Dictionnaire de données 1/3

Table : T_boutique				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
Id_boutique	Entier long	Faux	Faux	Faux
Date_saisie	Date Heure	Faux	Faux	Vrai
Id_article	Entier long	Vrai	Faux	Vrai
Qte_boutique	Entier long	Faux	Faux	Vrai

Table : T_commandes				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
Id_commande	Entier long	Faux	Faux	Faux
Id_objet	Entier long		Faux	Vrai
Qte_commande	Entier long		Faux	Vrai
Date_creation	Date heure		Faux	Vrai

Table : T_maj				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
Id_maj	Entier long	Faux	Faux	
Date_maj	Date heure	Faux	Faux	Vrai
Id_article	Entier long	Faux	Faux	Vrai
Qte_maj	Entier long	Faux	Faux	Vrai

Table : T_stock				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
Id_stock	Entier long	Faux	Faux	Vrai
Nom_stock	Text court	Faux	Faux	Vrai
Date_deb	Date heure	Faux	Faux	Vrai
Date_fin	Date Heure	Faux	Faux	Vrai

## Annexe 14

### Dictionnaire de données 2/3

Table : T_utilisateur				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
Code_utilisateur	Entier long	Faux	Faux	Vrai
Nom_utilisateur	Texte court	Faux	Faux	Vrai
Couleur_fond_utilisateur	Entier long	Faux	Faux	Vrai
Date_cde_utilisateur	Date et heure	Faux	Faux	Vrai

Table : T_client				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
cod_cli	Entier long	Faux	Faux	Faux
genrecli	Texte court	Faux	Faux	Vrai
nomcli	Texte court	Vrai	Faux	Vrai
prenomcli	Texte court	Faux	Faux	Vrai
adresse1cli	Texte court	Faux	Faux	Vrai
adresse2cli	Texte court	Faux	Faux	Vrai
adresse3cli	Texte court	Faux	Faux	Vrai
cpcli	Texte court	Faux	Faux	Vrai
villecli	Texte court	Faux	Faux	Vrai
telcli	Texte court	Faux	Faux	Vrai
emailcli	Texte court	Faux	Faux	Vrai
portcli	Texte court	Faux	Faux	Vrai
newsletter	Oui/Non	Faux	Faux	Vrai

Table : T_enseigne				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
Id_enseigne	Entier long	Faux	Faux	Vrai
lb_enseigne	Texte court	Faux	Faux	Vrai
Ville_enseigne	Texte court	Faux	Faux	Vrai
Dept_enseigne	Entier long	Faux	Faux	Vrai

## Annexe 14

### Dictionnaire de données 3/3

Table : T_dtlcode				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
codcde	Entier long	Faux	Faux	Vrai
codobj	Entier long	Faux	Faux	Vrai
qte	Entier	Faux	Faux	Vrai
Colis	Entier long	Faux	Faux	Vrai
Commentaire	Texte court	Faux	Faux	Vrai

Table : T_objet				
Nom de la colonne	Type	Required	DataUpdatable	AllowZeroLenght
codobj	Entier long	Faux	Faux	Vrai
libobj	Texte court	Faux	Faux	Vrai
Tailleobj	Texte court	Faux	Faux	Vrai
puobj	Monétaire	Faux	Faux	Vrai
Poidsobj	Texte court	Faux	Faux	Vrai
indispobj	Oui/Non	Faux	Faux	Vrai
O_imp	Entier long	Faux	Faux	Vrai
O_aff	Entier long	Faux	Faux	Vrai
O_cartp	Oui/Non	Faux	Faux	Vrai
idcondit	Entier long	Faux	Faux	Vrai
points	Entier long	Faux	Faux	Vrai
O_ordre_aff	Entier long	Faux	Faux	Vrai