DigitalGate SARL
Bangui, République centrafricaine
E-mail : info@digitalgateca.com
Octobre 2025

DigitalGate

ARCHITECTURE DES DONNÉES

Système national d'étiquetage et de suivi des marchandises.

RCA.

Collecte et traitement des données. Description technique

Version 2.0.1

Table des matières

1. Termes et abréviations

Termes et abréviations	**Définitions**
UAA	User Authentication and Authorization
AMS	Access Management System. DigitalGate SARL subsystem for managing user access rights
API	Application Programming Interface
ATTA	Authorized Territorial Tax Authority
B2B	Business-to-Business
B2C	Business-to-Consumer
C2C	Consumer-to-Consumer
DB	Database
DDA	DigitalGate SARL Digital Desk Audit
DEA	DigitalGate SARL Digital Track and Trace for Governments
OECR	Online Electronic Cash Register
PDF	Portable Document Format
TAW	Taxpayer Workplace
TIN	Taxpayer Identification Number
TOW	Tax Officer Workplace
VAT	Value-Added Tax
Bit register	Fixed-length sequence of bits, numbered from the right; the length of the register is measured in bytes
FCA	Fiscal Confirmation Attribute
FD	Fiscal Document
FDA	Fiscal Document Attribute
FDF	Fiscal Document/Data Format

FDO	DigitalGate SARL Fiscal Data Operator. A subsystem or module responsible for receiving, storing, and processing fiscal data, including receipts
FDT	Fiscal Data
FLC	Format Logic Control
FM	Fiscal Module
FMA	Fiscal Message Attribute for Fiscal Data Operator or Tax Authority
FMVC	Fiscal Module Validation Code

Termes et abréviations	**Définitions**
UAA	Authentification et autorisation des utilisateurs
AMS	Système de gestion des accès. Sous-système DigitalGate SARL pour la gestion des droits d'accès des utilisateurs
API	Interface de programmation d'application
ATTA	Autorité fiscale territoriale agréée
B2B	Entreprise à entreprise
B2C	Entreprise à consommateur
C2C	Entre particulier à particulier
DB	Base de données
DDA	DigitalGate SARL Audit numérique
DEA	DigitalGate SARL Suivi et traçabilité numériques pour les gouvernements
OECR	Caisse enregistreuse électronique en ligne
PDF	Format de document portable
TAW	Lieu de travail du contribuable
TIN	Numéro d'identification fiscale
TOW	Lieu de travail de l'agent fiscal

TVA	Taxe sur la valeur ajoutée
Registre de bits	Séquence de bits de longueur fixe, numérotée à partir de la droite ; la longueur du registre est mesurée en octets.
FCA	Attribut de confirmation fiscale
FD	Document fiscal
FDA	Attribut du document fiscal
FDF	Format du document/des données fiscales
FDO	DigitalGate SARL Opérateur de données fiscales. Sous-système ou module chargé de recevoir, stocker et traiter les données fiscales, y compris les reçus.
FDT	Données fiscales
FLC	Contrôle logique de format
FM	Module fiscal
FMA	Attribut de message fiscal pour l'opérateur de données fiscales ou l'autorité fiscale
FMVC	Code de validation du module fiscal

2. Résumé

DigitalGate SARL Digital Track and Trace for Governments fait partie de l'écosystème DigitalGate SARL et est une plateforme conçue pour assurer la traçabilité et le contrôle complets des produits soumis à accise marqués, de la production à la vente au détail. La plateforme prend en charge l'émission, l'application, l'agrégation, le mouvement et la radiation des codes de marquage uniques des produits, garantissant ainsi la conformité réglementaire et la transparence du marché.

L'architecture DigitalGate SARL Digital Track and Trace for Governments comprend une variété de services, notamment l'étiquetage, l'acceptation, la gestion des documents, l'émission et l'authentification des utilisateurs (User Authentication Authority, UAA), dans un environnement évolutif basé sur Kubernetes. L'architecture prend en charge les applications web et mobiles pour la gestion et la vérification en temps réel des codes de marquage.

Les principaux composants de l'infrastructure de données sont les suivants :

- PostgreSQL pour les données structurées,
- ClickHouse pour l'analyse à grande vitesse de volumes importants de données,
- ScyllaDB pour l'archivage à grande échelle,
- Redis pour la mise en cache en mémoire,
- Neo4j pour le suivi des relations graphiques.

Ces composants offrent une haute disponibilité, des performances efficaces et des capacités avancées d'analyse des données.

Grâce à des API sécurisées, DigitalGate SARL Digital Track and Trace for Governments permet l'interaction entre les fabricants, les détaillants et les autorités fiscales, en prenant en charge la déclaration automatisée, la gestion électronique des documents et l'analyse de la conformité réglementaire.

L'architecture modulaire et les flux de données en temps réel permettent d'améliorer la conformité fiscale, la détection des fraudes et la transparence tout au long de la chaîne d'approvisionnement des produits soumis à accise marqués.

3. Gestion et traitement des codes de marquage

La plateforme fournit une infrastructure unifiée pour générer, appliquer, agréger, vérifier et annuler les codes de marquage des produits à toutes les étapes de la chaîne d'approvisionnement. Tous les codes de marquage sont stockés dans un référentiel centralisé avec des métadonnées sur les types de produits, les statuts et les données de traçabilité.

La plateforme comprend un module intégré de délivrance de codes et prend en charge la génération de formats de codes en fonction des catégories de produits et des rôles des contribuables. Les codes invalides ou en double sont isolés dans une file d'attente distincte, enregistrés pour une analyse plus approfondie et affichés dans des rapports détaillés et des tableaux de bord de la plateforme.

Toutes les interactions avec les codes d'étiquetage, qu'il s'agisse d'opérations manuelles effectuées par l'utilisateur ou de processus lancés par la plateforme, sont enregistrées dans un journal d'événements unique. L'intégration avec des systèmes de stockage de journaux externes est prise en charge, ce qui garantit la traçabilité et la transparence.

Tous les participants au système d'étiquetage sont authentifiés via une couche d'accès unique. Les acteurs du marché utilisent un contrôle d'accès basé sur l'identification Keycloak.

Toutes les communications et tous les échanges de données dans le système d'étiquetage sont protégés par des canaux HTTPS sécurisés, qui garantissent la confidentialité et l'intégrité des informations transmises.

5. DigitalGate SARL Architecture de transfert de données Digital Track and Trace pour les gouvernements

5.1. Informations générales

La plateforme DigitalGate SARL Digital Track and Trace for Governments est un environnement unifié permettant de générer, d'émettre, d'appliquer, d'agréger, de suivre les mouvements et de supprimer les codes de marquage. Tous les événements liés aux codes de marquage sont collectés via des points de terminaison API sécurisés et stockés dans un référentiel centralisé à des fins de traçabilité et de reporting aux acteurs du marché.

La plateforme garantit la protection cryptographique des codes de marquage et fournit une vérification en temps réel ou asynchrone à toutes les étapes clés du cycle de vie du produit marqué : émission, application, agrégation, mise en circulation et retrait de la circulation.

5.1.1. Service de passerelle API DigitalGate SARL Suivi et traçabilité numériques pour les gouvernements

Le service API Gateway fournit des points d'accès sécurisés pour les systèmes externes (fabricants, distributeurs, détaillants) et les applications mobiles, et assure l'authentification et l'autorisation des requêtes API.

L'authentification est basée sur des protocoles d'échange de jetons (OAuth 2.0). L'authentification initiale est effectuée lors de l'enregistrement d'un jeton, et le jeton d'accès reçu est utilisé pour les interactions ultérieures.

Les jetons et leurs attributs sont stockés dans le système et vérifiés pour chaque demande entrante.

5.1.2. Service de réception des données du code de marquage

Le service de réception des données relatives aux codes de marquage traite les données entrantes liées aux événements suivants :

- Application des codes de marquage ;
- Événements d'agrégation ;
- Événements liés au mouvement des produits (mouvement entre les participants) ;
- Événements liés aux ventes ;

- Événements de radiation (en raison d'une perte, d'un dommage ou d'autres raisons).

Les données reçues doivent être conformes aux schémas JSON prédéfinis correspondant aux différents types d'événements.

Le service d'acceptation vérifie l'intégrité structurelle des données reçues, y compris la vérification des attributs obligatoires et des règles de validation de base.

5.1.3. Marquage des files d'attente de traitement des données de code

Après acceptation et validation initiale, les données sont envoyées vers des files d'attente de traitement :

- **File d'attente des événements propres** : comprend les événements valides et structurellement corrects concernant les codes de marquage, prêts à être utilisés dans les modules de traçabilité et d'analyse.
- **File d'attente des événements rejetés** : contient les données qui n'ont pas passé le contrôle structurel initial, qui présentent des attributs obligatoires manquants ou qui contiennent des références incorrectes aux codes de marquage.

Les participants reçoivent un retour immédiat si un événement échoue aux contrôles initiaux.

Les autorités réglementaires peuvent suivre les événements rejetés grâce à des rapports.

5.1.4. Collecte de statistiques à des fins d'analyse et de reporting

La plateforme DigitalGate SARL Digital Track and Trace for Governments comprend un service de collecte de statistiques qui recueille des indicateurs clés à partir de tous les événements liés aux codes de marquage. Les statistiques collectées couvrent :

- Le nombre de codes de marquage émis ;
- Le volume des événements d'agrégation ;
- Le nombre de mouvements et de ventes ;
- Le nombre et les raisons des annulations de codes.

Les statistiques des deux files d'attente (propres et rejetées) sont stockées dans une base de données centrale et sont disponibles pour des rapports analytiques.

5.1.5. Calcul et stockage des agrégats pour les tâches analytiques

Les événements vérifiés concernant les codes de marquage sont stockés dans une base de données analytique, où les valeurs agrégées sont calculées pour faciliter la création de rapports réglementaires et l'analyse opérationnelle. Les données analytiques comprennent les flux de mouvements de produits, les soldes des stocks et le suivi des statuts des codes de marquage.

5.1.6. Fonctions API de DigitalGate SARL Digital Track and Trace for Governments

L'API REST de la plateforme DigitalGate SARL Digital Track and Trace for Governments offre les fonctions clés suivantes :

- Authentification et autorisation ;
- Renouvellement des jetons ;
- Gestion de l'émission des codes de marquage ;
- Rapports sur l'application des codes ;
- Rapports d'agrégation ;
- Rapports sur les transferts de produits ;
- Rapports sur les ventes basés sur les documents de dépenses ;
- Rapports sur les événements de radiation.

Toutes les méthodes API garantissent la sécurité des communications et l'intégrité des données, permettant une intégration transparente avec les systèmes internes et fiscaux des participants.

6. Méthodes API DigitalGate SARL Suivi et traçabilité numériques pour les gouvernements

6.1. de connexion

Méthode : **POST /api/v1/uaa/default/login**

Objectif : authentifie l'utilisateur à l'aide d'un identifiant et d'un mot de passe.

Corps de la requête (x-www-form-urlencoded) :

{

« login » : « user@example.com »,

« password » : « password123 »

}

Réponse réussie : renvoie accessToken, refreshToken, accessExpiresAt, refreshExpiresAt et profile.

Réponses :

- 200 OK – Connexion réussie
- 400 Bad Request – Identifiants non valides
- 401 Non autorisé – Échec de l'authentification
- 500 Erreur interne du serveur – Erreur du service de connexion

6.2. Actualiser le jeton d'accès

Méthode : **POST /api/v1/uaa/default/refresh-token**

Objectif : Émet un nouveau jeton d'accès pour l'utilisateur à l'aide d'un jeton d'actualisation valide.

Corps de la requête (x-www-form-urlencoded) :

{

« refreshToken » : « eyJhbGciOiJIUzI1... »

}

Réponse réussie : renvoie accessToken, refreshToken, accessExpiresAt, refreshExpiresAt.

Réponses :

- 200 OK – Jeton actualisé
- 400 Bad Request – Jeton de rafraîchissement invalide ou expiré
- 500 Internal Server Error – Erreur de rafraîchissement du jeton

6.3. Déconnexion

Méthode : **POST /api/v1/uaa/default/logout**

Objectif : déconnecter l'utilisateur en invalidant le jeton d'actualisation.

Corps de la requête (x-www-form-urlencoded) :

{

« refreshToken » : « eyJhbGciOiJIUzI1... »

}

Réponses :

- 200 OK – Déconnexion réussie
- 400 Bad Request – Jeton invalide
- 500 Erreur interne du serveur – Erreur du service de déconnexion

6.4. Obtenir le profil de l'utilisateur authentifié

Méthode : **GET /api/v1/uaa/default/profile**

Objectif : renvoie le profil détaillé de l'utilisateur authentifié.

Réponse réussie :

Champs renvoyés : nom d'utilisateur, e-mail, e-mail vérifié, téléphone, téléphone vérifié, prénom, deuxième prénom, nom, nom complet, autorités.

Réponses :

- 200 OK – Données du profil renvoyées
- 401 Non autorisé – Jeton invalide ou manquant
- 500 Erreur interne du serveur – Erreur de récupération du profil

6.5. Ajouter un nouveau reçu

Méthode : **POST /api/v1/receipt**

Objectif : soumet un nouveau reçu fiscal pour traitement.

Corps de la requête (application/json) :

{
« receipt » : {
"kktRegId": "123456789",
« userInn » : « 9876543210 »,
« dateTime » : « 2025-04-29T12:00:00Z »,
« fiscalDriveNumber » : « 9287000100000001 »,
« fiscalDocumentNumber » : 12345,
« shiftNumber » : 1,
« requestNumber » : 1,
« taxationType » : 1,
« type d'opération » : 1,
« opérateur » : « Nom de l'opérateur »,
« utilisateur » : « Utilisateur particulier »,
« retailAddress » : « 123 Retail St »,
« totalSum » : 1000,00,
« cashTotalSum » : 500,00,
« ecashTotalSum » : 500,00,
« prépayéTotal » : 0,00,
« creditSum » : 0,00,
« provisionSum » : 0,00,
« ndsNo » : 0,00,
« nds0 » : 0,00,
« nds7 » : 0,00,
« nds20 » : 0,00,
« items » : [
{

« productCode » : « 01234567891234 »,
« name » : « Produit 1 »,
« typeDePaiement » : 1,
« prix » : 500,00,
« quantité » : 1,
« nds » : 20,
« somme » : 500,00
}
]
}
}

Réponse réussie : renvoie le receiptId (UUID).

Réponses :

- 200 OK – Reçu accepté
- 400 Bad Request – Données non valides
- 500 Internal Server Error – Erreur de traitement du reçu

6.6. Obtenir les informations relatives au reçu

Méthode : **GET /api/v1/receipt/info**

Objectif : récupère des informations détaillées sur un reçu spécifique à l'aide de son identifiant receiptId.

Paramètres de requête :

- receiptId (obligatoire, chaîne)

Réponse réussie :

Renvoie le receiptId, le statut (CREATED, DECLINED, APPROVED, PROCESSED), les détails du reçu et les éventuelles erreurs.

Réponses :

- 200 OK – Informations sur le reçu renvoyées
- 400 Bad Request – receiptId manquant ou invalide
- 404 Not Found – Reçu introuvable
- 500 Internal Server Error – Erreur de récupération des données

6.7. Ajouter un rapport d'utilisation

Méthode : **POST /api/v1/marking/utilization**

Objectif : création d'un document pour l'application de codes de marquage (apposition)

Corps de la requête (application/json) :

```
{
 « codes » : [« code1 », « code2 »],
 "factoryId": "factory-uuid",
 "productionLineId": "line-uuid"
}
```

Réponse réussie :

Renvoie reportId (UUID).

Réponses :

- 200 OK – Rapport accepté
- 400 Bad Request – Données non valides
- 500 Internal Server Error – Erreur de traitement du rapport

6.8. Ajouter un rapport UPD

Méthode : **POST /api/v1/marking/upd**

Objectif : soumet un rapport UPD (Universal Transfer Document) sous forme de chaîne JSON brute.

Corps de la requête (application/json) :

Chaîne JSON brute (format défini en externe).

Réponse réussie :

Renvoie le reportId (UUID).

Réponses :

- 200 OK – Rapport accepté
- 400 Bad Request – Format UPD non valide
- 500 Erreur interne du serveur – Erreur de traitement du rapport

6.9. Ajouter un rapport d'abandon

Méthode : **POST /api/v1/marking/dropout**

Objectif : Envoie un rapport concernant les pertes (ou détériorations) de codes de marquage.

Corps de la requête (application/json) :

```
{
 « codes » : [« code1 », « code2 »],
 "reason": "Perdu pendant le transport",
 « withChild » : true
}
```

Réponse réussie : renvoie le reportId (UUID).

Réponses :

- 200 OK – Rapport accepté
- 400 Bad Request – Données non valides
- 500 Erreur interne du serveur – Erreur de traitement du rapport

6.10. Ajouter un rapport de mise en service

Méthode : **POST /api/v1/marking/commissioning**

Objectif : soumet un rapport pour la mise en service (introduction en circulation) des codes de marquage.

Corps de la requête (application/json) :

```
{
 « productionDate » : « 2025-04-29T00:00:00Z »,
 "codes": ["code1", "code2"]
}
```

Réponse réussie : renvoie le reportId (UUID).

Réponses :

- 200 OK – Rapport accepté
- 400 Bad Request – Données non valides
- 500 Internal Server Error – Erreur de traitement du rapport

6.11. Ajouter un rapport d'agrégation

Méthode : **POST /api/v1/marking/aggregation**

Objectif : soumet un rapport pour l'agrégation des codes de marquage en unités logistiques plus importantes.

Corps de la requête (application/json) :

```
{
 « aggregate » : [
  {
   "aggregationType": "PALLET",
   « aggregationNumber » : « PALLET-123 »,
   « codes » : [« code1 », « code2 »]
  }
 ]
}
```

Réponse réussie : renvoie le reportId (UUID).

Réponses :

- 200 OK – Rapport accepté
- 400 Bad Request – Données non valides
- 500 Internal Server Error – Erreur de traitement du rapport

6.12. Obtenir les informations du rapport

Méthode : **GET /api/v1/marking/info**

Objectif : récupère les informations relatives à un rapport de notation spécifique à partir du rapportId.

Paramètres de requête :

- reportId (facultatif, chaîne, format UUID) – Identifiant unique du rapport.

Réponse réussie :

Renvoie un objet contenant :

- reportId (chaîne, UUID) – Identifiant du rapport.
- status (chaîne) – Statut du rapport. Valeurs possibles :
  - CREAÉ
  - REFUSÉ
  - APPROVÉ
  - TRAITÉ
- reportType (chaîne) – Type du rapport. Valeurs possibles :
  - UTILISATION (rapport d'utilisation)
  - ABONDON (rapport sur les abandons)
  - AGGREGATE (rapport d'agrégation)
  - INTRODUCTION (rapport de mise en service)
  - UPD (rapport de document UPD)
- reportInfo (tableau) – Données détaillées relatives au rapport, le format dépend du type de rapport.
- errors (facultatif, tableau) – Liste des erreurs associées au rapport, chacune contenant :
  - code (chaîne) – Code d'erreur.
  - message (chaîne) – Description de l'erreur.

Exemple de réponse réussie :

{
« result » : [
{

"reportId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
« status » : « APPROVED »,
« reportType » : « UTILISATION »,
« reportInfo » : [
{
// Données spécifiques au rapport
}
],
« erreurs » : [
{
« code » : « ERR001 »,
« message » : « Format de code non valide »
}
]
}
]
}

Réponses d'erreur :

- 400 Bad Request – Paramètres de requête invalides ou manquants.
- 404 Not Found – Aucun rapport trouvé avec l'identifiant spécifié.
- 500 Erreur interne du serveur – Erreur inattendue du serveur lors de la récupération des informations du rapport.

6.13. Créer une commande d'émission (clients externes)

Méthode : **POST /api/v1/emission/orders**

Objectif : crée un ordre d'émission pour le marquage des codes par un client externe, en spécifiant les produits, l'usine et la chaîne de production.

Corps de la requête (application/json) :

{
« country » : « RU »,
"factoryId": "factory-uuid",
"productionLineId": "line-uuid",
« products » : [
{

« type » : « CONSUMER_UNIT »,
« gtin » : « 04601234567890 »,
« quantity » : 1000,
« templateId » : « template-uuid »
}
]
}

Réponse réussie : renvoie l'ID de commande créé au format UUID.

Exemple de réponse :

{
« orderId » : « f27a34b9-5f64-40da-81d6-5db46dbab43e »
}

Codes de réponse :

- 200 OK – Commande créée avec succès
- 400 Bad Request – Données non valides ou champs manquants
- 500 Internal Server Error – Erreur du système d'émission

6.14. Obtenir le statut d'une commande d'émission (clients externes)

Méthode : **GET /api/v1/emission/orders**

Objectif : récupère le statut et les informations détaillées d'une commande d'émission créée par un client externe.

Paramètres de requête (facultatifs) :

- orderId (chaîne, format UUID) – Identifiant de la commande d'émission.

Réponse réussie : renvoie une liste des statuts des ordres correspondants.

Exemple de réponse :

{
« result » : [
{
"orderId": "f27a34b9-5f64-40da-81d6-5db46dbab43e",
« status » : « APPROVED »,
« orderInfos » : [
{
« gtin » : « 04601234567890 »,

« totalCodes » : 1000,
« availableCodes » : 950
}
],
« erreurs » : []
}
]
}

Valeurs d'état :

- CRÉÉ
- REFUSÉ
- VÉRIFIÉ
- APPROUVÉ
- PRÊT
- CLOS

Codes de réponse :

- 200 OK – Informations sur la commande renvoyées
- 400 Bad Request – ID de commande invalide ou manquant
- 404 Not Found – Commande introuvable
- 500 Internal Server Error – Erreur de récupération

6.15. Obtenir les codes de marquage

Méthode : **GET /api/v1/emission/codes**

Objectif : récupère les codes de marquage générés dans le cadre d'une commande d'émission spécifique.

Paramètres de requête (facultatifs) :

- orderId (chaîne, UUID) – Identifiant de l'ordre d'émission
- gtin (chaîne) – GTIN du produit
- quantity (entier) – Nombre de codes à récupérer
- lastBlockId (chaîne, UUID) – Pour la pagination ou la continuation

Réponse réussie : renvoie un ou plusieurs blocs de codes.

Exemple de réponse :

{
« result » : [

{
"blockId": "15b6d2d4-92b8-472e-9391-91f92b750b47",
« codes » : [
« 010460123456789021abcd1234 »,
« 010460123456789021abcd1235 »
],
« erreurs » : []
}
]
}

Codes de réponse :

- 200 OK – Codes renvoyés
- 400 Bad Request – Paramètres non valides
- 404 Not Found – Codes introuvables
- 500 Internal Server Error – Erreur de récupération

6.16. Obtenir les blocs de codes de marquage

Méthode : **GET /api/v1/emission/codes/blocks**

Objectif : récupère les blocs de codes de marquage regroupés par GTIN et ID de commande. Utile pour analyser les lots de codes précédemment émis.

Paramètres de requête (facultatifs) :

- orderId (chaîne, UUID) – Identifiant de la commande d'émission
- gtin (chaîne) – GTIN du produit

Réponse réussie : renvoie une liste de blocs pour chaque GTIN correspondant.

Exemple de réponse :

{
« result » : [
{
« orderId » : « 2e7f885e-c91e-4a41-a06e-f94d33edee15 »,
"gtin": "04601234567890",
« blocks » : [
{
« blockId » : « 6d7c88ef-59c9-4229-8bfe-7d928e434b20 »,
« blockDateTime » : 1714406400000,

« quantité » : 500
}
],
« erreurs » : []
}
]
}

Codes de réponse :

- 200 OK – Blocs renvoyés
- 400 Bad Request – Requête invalide
- 404 Not Found – Aucun bloc trouvé
- 500 Internal Server Error – Erreur interne

6.17. Obtenir les détails des usines avec lignes

Méthode : **GET /api/v1/service/profile/factories/{factoryId}**

Objectif : renvoie des informations détaillées sur une usine spécifique, y compris toutes les lignes de production associées.

Paramètre de chemin :

- factoryId (entier, obligatoire) – ID de l'usine

Réponse réussie : renvoie des informations complètes sur l'usine et une liste de ses lignes de production.

Exemple de réponse :

{
« id » : 101,
"address": "Zone industrielle 5, bloc 12",
« manufacturerId » : 150,
« nom » : « Usine A »,
« code » : « FAC-A1 »,
« pays » : « Allemagne »,
« code pays » : « DE »,
« factoryLines » : [
{
« id » : 501,
« factoryId » : 101,
« code » : « LINE-01 »,

« name » : « Ligne 1 »,
« productName » : « Produit X »,
« lastActivityTime » : « 2025-04-20T15:32:00Z »,
« productsCount » : 5000,
« packsCount » : 1000,
« defectPercentage » : 0,5,
« actif » : vrai
}
]
}

Codes de réponse :

- 200 OK – Données récupérées avec succès
- 400 Bad Request – ID d'usine non valide

6.18. Obtenir la liste des usines

Méthode : **GET /api/v1/service/profile/factories**

Objectif : récupère la liste des usines pour un client FABRICANT.

Réponse réussie : renvoie une liste d'usines.

Exemple de réponse :

[
{
« id » : 1,
« address » : « Industrial Street 1 »,
« manufacturerId » : 100,
"name": "Usine n° 1",
« code » : « FAC-ONE »,
« factoryLinesTotalCount » : 3,
« pays » : « États-Unis »,
« countryCode » : « US »
},
{
« id » : 2,
« adresse » : « Industrial Street 2 »,
« manufacturerId » : 100,

« nom » : « Usine deux »,
« code » : « FAC-TWO »,
« factoryLinesTotalCount » : 5,
« pays » : « États-Unis »,
« countryCode » : « US »
}
]

Codes de réponse :

- 200 OK – Usines récupérées avec succès
- 400 Bad Request – Demande non valide

6.19. Obtenir la liste des pays

Méthode : **GET /api/v1/service/profile/countries**

Objectif : récupère la liste de tous les pays disponibles.

Réponse réussie : renvoie un tableau de pays.

Exemple de réponse :

[
{
« code » : « US »,
« country » : « États-Unis »
},
{
« code » : « DE »,
« country » : « Germany »
}
]

Codes de réponse :

- 200 OK – Pays récupérés avec succès
- 400 Bad Request – Demande non valide

6.20. Obtenir tous les groupes de produits

Méthode : **GET /api/v1/service/product-groups**

Objectif : renvoie une liste de tous les groupes de produits dans un format court, adapté à la logique d'émission.

Réponse positive : renvoie un tableau de groupes de produits.

Exemple de réponse :

[
{
« code » : « TOBACCO »,
« sampleDescription » : « Produits du tabac »
},
{
« code » : « ALCOOL »,
« descriptionDeL'Échantillon » : « Boissons alcoolisées »
}
]

Codes de réponse :

- 200 OK – Groupes de produits récupérés avec succès
- 400 Bad Request – Demande non valide

6.21. Valider le code de marquage

Méthode : **POST /api/v1/validation/codes**

Objectif : valider un seul code de marquage.

Corps de la requête : (multipart/form-data)

markingCode (chaîne, obligatoire) — Code de marquage à valider.

Réponse positive : renvoie le résultat de la validation du code de marquage, y compris le statut de confirmation et des informations supplémentaires sur le produit.

Exemple de réponse :

{
« markingCodeConfirmed » : true,
"errors": [],
"info": {
"markingCode": "0101234567890123abcd",
« producedAt » : « 2025-04-25T09:10:00Z »,
« productName » : « Produit A »,
« nom du fabricant » : « Fabricant X »,
« factoryName » : « Usine Y »,
« factoryLineName » : « Ligne 1 »,
« factoryLineCode » : « L001 »,

« nomDuMagasin » : « Magasin ABC »,
« storeAddress » : « 123, rue Principale »,
« productGroupCode » : « TOBACCO »,
« gtin » : « 01234567890123 »
}
}

Codes de réponse :

- 200 OK – Code de marquage validé avec succès
- 400 Bad Request – Données d'entrée non valides

6.22. Obtenir la liste des reçus

Méthode : **POST /api/v1/service/receipts**

Objectif : récupère une liste des reçus disponibles pour le client actuel avec pagination.

Corps de la requête :

{
« paging » : {
"pageSize": 10,
"pageRef": []
}
}

Réponse positive : renvoie une liste paginée des reçus avec les informations de base.

Exemple de réponse :

{
« nextPageRef » : [« c29tZS1uZXh0LXBhZ2UtcmVm »],
« data » : [
{
« receiptId » : « a3bb88a5-7f4e-4b98-a05b-2c3c7d92f5ec »,
« createdAt » : « 2025-04-26T13:00:00Z »,
« operationType » : 1,
« status » : « APPROVED »,
« nomDuFabricant » : « Fabricant A »
},
{
« receiptId » : « d5fa2cbe-6b10-49c3-b688-5f8145b82476 »,

« createdAt » : « 2025-04-26T15:30:00Z »,
« operationType » : 2,
« status » : « PROCESSED »,
« nomDuFabricant » : « Fabricant B »
}
]
}

Codes de réponse :

- 200 OK – Liste des reçus récupérée avec succès
- 400 Bad Request – Données d'entrée non valides

6.23. Obtenir les détails du reçu

Méthode : **POST /api/v1/service/receipts/details**

Objectif : récupère les informations détaillées d'un reçu spécifique.

Corps de la requête :

{
« receiptId » : « a3bb88a5-7f4e-4b98-a05b-2c3c7d92f5ec »
}

Réponse positive : renvoie des informations détaillées sur le reçu, notamment les informations utilisateur, les données fiscales et le statut.

Exemple de réponse :

{
« createdAt » : « 2025-04-26T13:00:00Z »,
« receiptId » : « a3bb88a5-7f4e-4b98-a05b-2c3c7d92f5ec »,
« user » : « John Doe »,
« retailAddress » : « 123 Market Street »,
« userInn » : « 1234567890 »,
« taxationType » : 1,
« operator » : « Opérateur X »,
« shiftNumber » : 12,
« dateTime » : « 2025-04-26T13:15:00Z »,
« fiscalDocumentNumber » : 987654,
« fiscalDriveNumber » : « DRIVE123456 »,
« productsTotalCount » : 5,

« status » : « APPROVED »,

« erreurs » : []

}

Codes de réponse :

- 200 OK – Détails du reçu récupérés avec succès
- 400 Bad Request – Données d'entrée non valides

6.24. Obtenir les détails du produit par GTIN

Méthode : **POST /api/v1/service/products/details**

Objectif : récupère toutes les informations sur un produit à partir de son GTIN ou de son code de marquage.

Le GTIN peut être nul si la recherche s'effectue uniquement à partir du code de marquage.

Corps de la requête :

{

« gtin » : « 04612345678901 »,

"markingCode": "0104612345678901212vA7dE39rL0G1Z"

}

Réponse positive : renvoie des informations détaillées sur le produit et son statut.

Exemple de réponse :

{

« markingCode » : « 0104612345678901212vA7dE39rL0G1Z »,

« status » : « INTRODUCED »,

« packagingType » : « PACK »,

« packagingLevel » : 1,

« produit » : {

« gtin » : « 04612345678901 »,

« nom du fabricant » : « Tobacco Corp »,

« productName » : « Marlboro Red »,

« productTrademark » : « Marlboro »,

« productVendorCode » : « MR123 »,

« manufacturerId » : 1001,

« productGroupCode » : « TOBACCO »

},

« statuses » : {

« INTRODUCED » : {

« status » : « INTRODUCED »,
« operationTime » : « 2025-04-28T10:00:00Z »
}
},
« opérations » : [
{
« operationAction » : « INTRODUCED »,
« operationTime » : « 2025-04-28T10:00:00Z »,
« clientName » : « Retailer Ltd »
}
],
« childrenTotalCount » : 0,
« propriétaire » : « Retailer Ltd »
}

Codes de réponse :

- 200 OK – Détails du produit récupérés avec succès
- 400 Bad Request – Données d'entrée non valides

6.25. Obtenir les produits agrégés

Méthode : **POST /api/v1/service/products/details/children**

Objectif : récupère une liste de produits agrégés par GTIN et/ou numéro d'agrégation. Le GTIN peut être nul si le numéro d'agrégation est fourni.

Corps de la requête :

{
« gtin » : « 04612345678901 »,
"aggregationNumber": "AGGREGATE-001",
"paging": {
« pageNumber » : 0,
« pageSize » : 10
}
}

Réponse positive :

Renvoie une liste des produits contenus dans l'agrégat spécifié.

Exemple de réponse :

[
{
« gtin » : « 04612345678901 »,
"markingCode": "0104612345678901212vA7dE39rL0G1Z",
« status » : « AGGREGATED »,
« productName » : « Marlboro Red »,
« manufacturerName » : « Tobacco Corp »,
« packagingType » : « CARTON »,
« modifiedAt » : « 2025-04-28T10:30:00Z »,
« packagingLevel » : 2,
« productGroupCode » : « TOBACCO »,
« marque » : « Marlboro »,
« propriétaire » : « Retailer Ltd »
}
]

Codes de réponse :

- 200 OK – Produits agrégés récupérés avec succès
- 400 Bad Request – Données d'entrée non valides

6.26. Obtenir la liste des fiches produits

Méthode : **POST /api/v1/service/product-cards**

Objectif : renvoie une liste paginée des fiches produits disponibles pour le client actuel. Prend en charge le filtrage et le tri.

Corps de la requête :

{
« paging » : {
"pageNumber": 0,
"pageSize": 10
},
« filtres » : [
{
« id » : « productName »,
« value » : {
« value » : [« Marlboro Red »]

}
}
],
« tri » : [
{
« id » : « productVendorCode »,
« desc » : false
}
]
}

Réponse positive : renvoie une liste d'objets de fiche produit avec pagination.

Exemple de réponse :

[
{
« gtin » : « 04612345678901 »,
"manufacturerName": "Tobacco Inc.",
« productName » : « Marlboro Red »,
« productVendorCode » : « MR123 »,
« productTrademark » : « Marlboro »,
« manufacturerId » : 101,
« productGroupCode » : « TOBACCO »,
« packageType » : « PACK »
}
]

Codes de réponse :

- 200 OK – Liste des fiches produits renvoyée avec succès
- 400 Bad Request – Filtres, tri ou pagination invalides

6.27. Obtenir les commandes d'émission

Méthode : **POST /api/v1/service/orders**

Objectif : récupère une liste paginée des commandes d'émissions disponibles pour le client actuel. Cette opération renvoie N+1 éléments pour une taille de page N si la page suivante existe.

Corps de la requête :

{

« paging » : {
"pageNumber": 0,
"pageSize": 10
},
« filtres » : [
{
« id » : « status »,
« value » : {
« value » : « APPROVED »
}
}
],
« tri » : [
{
« id » : « createdAt »,
« desc » : true
}
]
}

Paramètres de requête :

- paging (objet, facultatif) – Contient pageNumber et pageSize
- filters (tableau, facultatif) – Critères de filtrage pour les ordres d'émission (par exemple, par statut, date, etc.)
- sortings (tableau, facultatif) – Règles de tri des résultats (par exemple, par date de création)

Réponse positive : renvoie une liste des ordres d'émission correspondant aux filtres et à la pagination fournis.

Exemple de réponse :

[
{
« orderId » : « 4e14cf3e-21ef-4b2a-9fd2-8421c2f0e1e4 »,
"createdAt": "2025-04-25T08:10:00Z",
« closedAt » : « 2025-04-26T15:00:00Z »,
« status » : « APPROVED »,
« totalMarkingCodes » : 100000,

« participant » : « Exemple de fabricant »

}

]

Codes de réponse :

- 200 OK – Ordres d'émission récupérés avec succès
- 400 Bad Request – Structure de requête ou paramètres de filtrage non valides

6.28. Obtenir les détails des ordres d'émission

Méthode : **POST /api/v1/service/orders/details**

Objectif : récupère des informations détaillées sur un ordre d'émission spécifique.

Corps de la requête :

{

« orderId » : « 4e14cf3e-21ef-4b2a-9fd2-8421c2f0e1e4 »

}

Paramètres de la requête :

- orderId (chaîne, obligatoire) – Identifiant unique (UUID) de l'ordre d'émission

Réponse positive : renvoie des informations détaillées sur l'ordre d'émission, notamment le statut, l'ID d'usine, le groupe de produits et les codes disponibles.

Exemple de réponse :

{

« result » : {

"orderId": "4e14cf3e-21ef-4b2a-9fd2-8421c2f0e1e4",

"createdAt": "2025-04-25T08:10:00Z",

« closedAt » : « 2025-04-26T15:00:00Z »,

« status » : « APPROVED »,

« totalMarkingCodes » : 100000,

« factoryId » : « 12345 »,

« country » : « Allemagne »,

« productionLineId » : « 67890 »,

« productsTotalCount » : 2,

« productGroupCode » : « TOBACCO »,

« availableCodes » : 98765,

« erreurs » : []

}

}

Codes de réponse :

- 200 OK – Détails de la commande d'émission récupérés avec succès
- 400 Bad Request – ID de commande invalide ou manquant

6.29. Obtenir la liste des documents

Méthode : **POST /api/v1/service/documents**

Objectif : renvoie une liste paginée des documents (rapports) disponibles pour le client actuel, filtrés par type, date de création ou propriétaire.

Corps de la requête :

{
« paging » : {
"pageSize": 10,
"pageRef": []
},
« type » : « UTILISATION »,
« createdAt » : « 2024-12-01T00:00:00Z »,
« docOrOwner » : « 1234567890 »
}

Paramètres de requête :

paging (objet, facultatif) – Paramètres de pagination :

pageSize (entier) – Nombre de résultats par page

pageRef (tableau de chaînes base64) – Référence pour la page suivante

type (chaîne, facultatif) – Type de rapport (UTILISATION, DROPOUT, AGGREGATE, INTRODUCE, UPD)

createdAt (chaîne, facultatif) – Filtrer par date de création (ISO 8601)

docOrOwner (chaîne, facultatif) – ID du document ou identifiant du propriétaire

Réponse positive : renvoie une liste paginée des résumés de rapports.

Exemple de réponse :

{
« nextPageRef » : [],
"data": [
{
« reportId » : « d38a3ed2-2ef2-4d2b-a1e9-b4fd03aa98a3 »,
« status » : « CREATED »,

« type » : « UTILISATION »,
« createdAt » : « 2024-12-01T10:30:00Z »,
"owner": "1234567890",
"seller": "Company A",
"buyer": "Company B"
}
]
}

Codes de réponse :

- 200 OK – Liste récupérée avec succès
- 400 Bad Request – Données d'entrée non valides

6.30. Obtenir la liste des factures

Méthode : **GET /api/v1/service/invoice/list**

Objectif : renvoie une liste paginée des factures disponibles pour le client actuel, filtrées par date de création.

Corps de la requête :

{
« gtin » : « chaîne »,
"paging": {
"pageNumber": 0,
"pageSize": 0
},
« filtres » : [
{
"id": "string",
« value » : {}
}
],
"sortings": [
{
"id": "string",
"desc": true
}

]

}

Exemple de réponse :

{
« totalElements » : 0,
« totalPages » : 0,
« data » : [
{
« id » : 0,
« createdAt » : « 2025-12-25T06:41:39.537Z »,
« updatedAt » : « 2025-12-25T06:41:39.537Z »,
« status » : « string »,
« url » : « chaîne »,
« senderTin » : « chaîne »,
« receiverTin » : « chaîne »
}
]
}

Codes de réponse :

- 200 OK – Liste récupérée avec succès
- 400 Bad Request – Données d'entrée non valides

6.31. Obtenir les détails de la facture

Méthode : **GET /api/v1/service/invoice/info**

Objectif : renvoie les informations détaillées de la facture disponibles pour le client actuel.

Paramètre de requête :

- id (entier) – Pour obtenir des informations sur une facture, vous devez fournir l'ID de la facture à partir de la liste des factures disponibles pour le client actuel.

Exemple de réponse :

{
« id » : 0,
"createdAt": "2025-12-25T06:44:13.281Z",
"updatedAt": "2025-12-25T06:44:13.281Z",
"status": "string",

"url": "string",
"senderTin": "string",
"receiverTin": "string"
}

Codes de réponse :

- 200 OK – Liste récupérée avec succès
- 400 Bad Request – Données d'entrée non valides