Fiche Technique et Documentation de l'API

PestAI - IA Analysis Microservice (v8.0)

Auteur : L'Équipe IA de PestAI

Version du Document : 1.0

Date : 19 juillet 2025

1. Aperçu & Mission du Service

1.1. Mission

Ce document détaille le fonctionnement du Microservice d'Analyse IA, le "cerveau" de la plateforme PestAI. Sa seule et unique responsabilité est de recevoir une image brute (d'une plante ou d'un ravageur) et de retourner une analyse experte complète, structurée et riche en données au format JSON.

1.2. Audience

Ce document est destiné à toutes les équipes interagissant avec ce service :

Équipe Backend (Nest.js) : Pour comprendre comment appeler le service, gérer ses réponses et ses erreurs.

Équipe Front : Pour connaître la structure exacte des données à afficher dans l'interface utilisateur.

Équipe NLP: Pour comprendre les données textuelles et les tags générés, en vue de construire la base de connaissances et les modèles d'analyse.

2. Rôle dans l'Architecture de la Plateforme

Ce service est un composant découplé et autonome. Il ne gère ni les utilisateurs, ni les sessions, ni la base de données. Il est appelé par le backend principal (Nest.js), qui orchestre la logique métier.

Flux de données typique :

Generated code

Utilisateur -> App Mobile/Web -> Backend Nest.js -> [CE SERVICE IA] -> Backend Nest.js -> Base de Données (PostgreSQL) -> Utilisateur

Use code with caution.

3. Spécification de l'Endpoint

3.1. Point d'accès

Méthode : POST

URL : /api/v8/analyze-image

3.2. Description

Prend une image en entrée et retourne une analyse agronomique complète, incluant l'identification du sujet, la détection des problèmes, l'évaluation de la sévérité et des recommandations sourcées.

3.3. Requête

Le service attend une requête au format multipart/form-data.

Paramètres :

Nom du champ	Type	Description	Obligatoire
file	Fichier	Le fichier image à analyser. Formats supportés : image/jpeg, image/png.	Oui

3.4. Réponses

En cas de succès, le service retourne un objet JSON avec la structure AIAnalysisResponse suivante.

Structure du JSON de Réponse :

Generated json

{

“subject”: {

“subjectType”: “string”,

“description”: “string”,

“confidence”: “float”

},

“detections”: [

{

“className”: “string”,

“confidenceScore”: “float”,

“severity”: “string”,

“boundingBox”: {

“x_min”: “float”,

“y_min”: “float”,

“x_max”: “float”,

“y_max”: “float”

},

“details”: {

“description”: “string”,

“impact”: “string”,

“recommendations”: {

“biological”: [

{
“solution”: “string”,
“details”: “string”,
“source”: “string”
}
],
“chemical”: [],
“cultural”: []
},
“knowledgeBaseTags”: [“string”]
}
}
]
}
Use code with caution.
Json
Dictionnaire des Données :

Champ Type Description
subject Objet Le sujet principal identifié dans l’image.
subject.subjectType String Type de sujet : ‘PLANT’, ‘PEST’, ou ‘UNKNOWN’. Crucial pour l’aiguillage de l’UI.
subject.description String Nom commun et scientifique du sujet. Ex: “Plant de Tomate (Solanum lycopersicum)”.
subject.confidence Float Score de confiance (0.0 à 1.0) de l’IA dans son identification du sujet.
detections Liste d’Objets Liste des problèmes (maladies/ravageurs) détectés. Peut être une liste vide.
detections[].className String Nom commun et scientifique du problème détecté.

detections[].confidenceScore Float Score de confiance de l'IA pour cette détection spécifique.

detections[].severity String (Enum) Niveau de gravité estimé : 'LOW', 'MEDIUM', 'HIGH', 'CRITICAL'. KPI clé pour les tableaux de bord.

detections[].boundingBox Objet Coordonnées normalisées (0.0 à 1.0) pour dessiner un rectangle sur l'image.

detections[].details Objet Toutes les informations actionnables sur la détection.

details.description String Description textuelle du problème. Idéal pour l'affichage et la synthèse vocale.

details.impact String Description de l'impact sur les cultures.

details.recommendations Objet Recommandations de traitement, groupées par catégorie.

recommendations.biological Liste Liste des solutions biologiques.

recommendations.chemical Liste Liste des solutions chimiques.

recommendations.cultural Liste Liste des solutions culturales.

solution.source String (URL) Très important : Lien vérifiable vers une source scientifique/gouvernementale.

details.knowledgeBaseTags Liste de Strings Très important pour la recherche : Mots-clés pour indexer la détection dans une base de connaissances.

Le service utilise les codes de statut HTTP standards pour indiquer les erreurs.

Code HTTP Message d'Erreur (detail) Cause Probable

415 Unsupported Media Type "Format d'image non supporté. Utilisez JPEG ou PNG." Le fichier envoyé n'est ni un JPEG, ni un PNG.

429 Too Many Requests "You exceeded your current quota…" (Non géré par le code, mais possible) L'API Gemini est surchargée. Le backend appelant (Nest.js) devrait implémenter une logique de "retry".

502 Bad Gateway "L'API a retourné une réponse vide." ou "Réponse invalide du service d'IA." Problème de communication avec l'API Gemini, ou l'IA n'a pas renvoyé un JSON valide.

503 Service Unavailable "Erreur interne du service IA: [message d'erreur de Gemini]" Une erreur inattendue s'est produite lors de l'appel à l'API Gemini.

4. Guide Pratique d'Utilisation

4.1. Exemple de Requête avec cURL

Voici comment tester l'API depuis un terminal avec une image nommée feuille-malade.jpg.

Generated bash

curl -X POST "http://127.0.0.1:8000/api/v8/analyze-image"

-H “accept: application/json”

-H “Content-Type: multipart/form-data”

-F “;type=image/jpeg”

Use code with caution.

Bash

5. Déploiement & Configuration

5.1. Variables d’Environnement

Le service requiert une seule variable d’environnement pour fonctionner :

GEMINI_API_KEY : Votre clé API secrète pour l’API Google Gemini.

Cette clé doit être fournie via un fichier .env en développement, ou via un gestionnaire de secrets en production.

5.2. Conteneurisation (Docker)

Le service est conçu pour être déployé en tant que conteneur Docker. Un Dockerfile est fourni dans le dépôt du projet. Il utilise Gunicorn comme serveur de production pour gérer plusieurs requêtes simultanées.