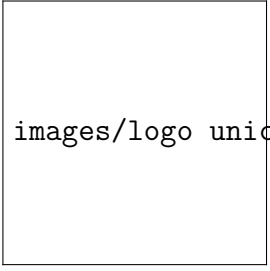


images/logoemsi.png

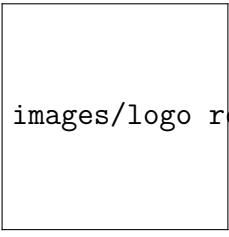


images/logo_unica.jpg

PROJET DE FIN D'ÉTUDES

Développement d'une application mobile basée sur l'IA pour la détection du niveau d'aflatoxine dans les grains de maïs

Filière : **Ingénierie Informatique et Réseaux (5^e année)**



images/logo_roquette.png

Organisme d'accueil : ROQUETTE FRÈRES

Réalisé par : QORAICHI IBTIHAL

**Encadrant(s) professionnel(s) : PIERRE-LOUIS BESCOND
MATHIEU KLIMCZAK**

Encadrant pédagogique : MOHAMMED BOUSSEMAH

Année universitaire : 2024/2025

Dédicaces

Je dédie ce travail, fruit de plusieurs mois d'efforts, de recherches et d'engagement,

À mes chers parents, Laila Akrouit et Hammadi Qoraichi,
pour leur amour infini, leurs sacrifices, leurs prières et leur soutien indéfectible dans chaque étape de ma vie. Vous êtes ma plus grande source d'inspiration et de force.

À la mémoire de mon grand-père, Abdessalam Qoraichi,
dont le souvenir continue d'éclairer mes pas et d'inspirer mes ambitions.

À mes frères et sœurs Wiam, Rim et Amine,
pour leur affection, leur complicité et leurs encouragements constants, qui m'ont soutenue dans les moments les plus exigeants.

À mes grands-parents et à toute ma famille,
pour leur bienveillance, leurs prières et leur confiance en moi.

À mes amis,
pour leur présence précieuse, leur joie et leurs encouragements qui ont rendu ce parcours plus humain et plus agréable.

Enfin,
à toutes celles et ceux qui, de près ou de loin, ont cru en moi, m'ont soutenue et motivée tout au long de ce chemin.

Merci à vous tous.

Remerciements

J'exprime ma profonde reconnaissance à mon encadrant professionnel, **Monsieur Pierre-Louis Bescond**, Head of Data Analytics chez Roquette, pour la confiance qu'il m'a accordée, ses conseils avisés et son accompagnement tout au long de cette expérience.

Mes sincères remerciements vont également à **Monsieur Mathieu Klimczak**, Head of DevOps, pour son aide précieuse, sa disponibilité et son soutien constant dans la réalisation de ce projet.

Je tiens à remercier chaleureusement **toute l'équipe Data de Roquette**, pour leur accueil, leur disponibilité et leur bienveillance, qui ont largement contribué à rendre ce stage formateur et agréable.

Je souhaite également remercier mon encadrant pédagogique, **Monsieur Mohammed Bousmah**, pour son suivi rigoureux, ses conseils précieux et son accompagnement tout au long de ces six mois.

Enfin, j'exprime ma gratitude à l'**École Marocaine des Sciences de l'Ingénieur (EMSI)** pour la qualité de l'enseignement reçu durant ces cinq années de formation, qui m'ont permis de mener à bien ce projet.

*À toutes celles et ceux qui, de près ou de loin, ont contribué à cette réussite :
merci infiniment.*

Résumé

Ce projet de fin d'études, mené au sein de Roquette Frères, a porté sur le développement d'une solution innovante pour la détection du niveau d'aflatoxine dans les grains de maïs. Le projet, nommé ADOM, vise à remplacer un processus d'analyse en laboratoire long et coûteux par une application mobile basée sur l'intelligence artificielle. La solution utilise un modèle de Deep Learning pour analyser des images de grains de maïs et fournir une classification instantanée, permettant d'accélérer la prise de décision d'achat et d'améliorer la sécurité alimentaire. Le développement a suivi une méthodologie agile Scrum et s'est appuyé sur un écosystème technologique moderne incluant Python, TensorFlow, Flask pour le backend, Flutter pour l'application mobile, et les services cloud de Microsoft Azure pour l'entraînement des modèles et le déploiement.

Mots clés : Intelligence Artificielle, Deep Learning, Vision par Ordinateur, Application Mobile, Flutter, Flask, Azure, Aflatoxine, Sécurité Alimentaire, Maïs.

Abstract

This final year project, carried out at Roquette Frères, focused on developing an innovative solution for detecting aflatoxin levels in corn kernels. The project, named ADOM, aims to replace a lengthy and costly laboratory analysis process with a mobile application based on artificial intelligence. The solution uses a Deep Learning model to analyze images of corn kernels and provide instant classification, thereby speeding up purchasing decisions and improving food safety. The development followed an Agile Scrum methodology and relied on a modern technological ecosystem including Python, TensorFlow, Flask for the backend, Flutter for the mobile application, and Microsoft Azure cloud services for model training and deployment.

Keywords : Artificial Intelligence, Deep Learning, Computer Vision, Mobile Application, Flutter, Flask, Azure, Aflatoxin, Food Safety, Corn.

Liste des abréviations

Abréviation	Signification
IA	Intelligence Artificielle
API	Application Programming Interface
CNN	Convolutional Neural Network (Réseau de Neurones Convolutif)
CI/CD	Continuous Integration/Continuous Deployment (Intégration Continue/Déploiement Continu)
RSE	Responsabilité Sociale des Entreprises
DADS	Data Analysis & Data Science
DS	Data Science
EDA	Exploratory Data Analysis
PPB	Parts Per Billion (Parties par milliard)
RGPD	Règlement Général sur la Protection des Données
UML	Unified Modeling Language
IDE	Integrated Development Environment (Environnement de Développement Intégré)
IoT	Internet of Things (Internet des objets)
IAM	Identity and Access Management (Gestion des identités et des accès)

Table des matières

Table des figures

Introduction générale

L'entreprise fait actuellement face à une problématique majeure lors de l'achat de grains de maïs. Chaque fois que ses acheteurs se rendent chez des fournisseurs, ils doivent prélever des échantillons de grains et les envoyer ensuite au laboratoire de l'entreprise pour analyse. Ce processus, bien que fiable, est long et retarde considérablement la prise de décision d'achat. Il vise à détecter le niveau d'**aflatoxine** présent dans les grains de maïs. L'aflatoxine est une toxine produite par certains champignons susceptibles de contaminer les céréales comme le maïs, et qui représente un risque important pour la santé humaine et animale.

Dans le cadre du **projet ADOM**, nous avons successivement entrepris plusieurs étapes visant à concevoir, développer et déployer une solution complète de détection de l'aflatoxine dans les grains de maïs. La première étape a consisté à préparer l'environnement technique en récupérant et en configurant le matériel nécessaire. Une attention particulière a été portée à l'utilisation des ressources cloud, notamment **Azure Machine Learning**, ainsi qu'à la mise en place des outils de développement tels que Python et Azure DevOps pour assurer une gestion efficace du projet.

Une fois l'environnement opérationnel, nous avons développé nos premiers modèles d'intelligence artificielle en local, en utilisant des techniques de **Deep Learning** adaptées à la reconnaissance et à l'analyse d'images. La phase suivante a été la création d'une **API Flask**, permettant la communication entre nos modèles d'IA et une application mobile. Ce dispositif initial, fonctionnant en local, a constitué une étape cruciale pour valider la faisabilité du système.

Par la suite, nos efforts se sont concentrés sur l'amélioration des modèles en optimisant leur précision et leur robustesse, afin de garantir des prédictions fiables dans des conditions variées. Enfin, pour assurer une gestion optimale du code et une automatisation efficace des processus, nous avons migré l'ensemble du développement vers **Azure DevOps**, en y intégrant des pipelines de compilation et de déploiement automatique.

Ce processus structuré a permis de bâtir un système complet, **scalable** et prêt à être déployé dans un environnement plus large, tout en nous permettant d'adopter de bonnes pratiques en gestion de projet logiciel et en développement de solutions d'intelligence artificielle.

Ce mémoire retrace l'ensemble des étapes de ce projet. Il est structuré de la manière suivante :

- **Chapitre I :** Présente le contexte général du projet, l'organisation d'accueil, les enjeux industriels ainsi que les objectifs poursuivis.
- **Chapitre II :** Détaille l'étude préliminaire, les solutions existantes et les besoins identifiés.
- **Chapitre III :** Expose l'analyse fonctionnelle et conceptuelle, avec les cas d'utilisation, les user stories et les diagrammes de séquence et de classes.
- **Chapitre IV :** Présente l'étude technique, les choix d'architecture, l'environnement de développement et la mise en œuvre concrète du système, illustrée par les interfaces clés de l'application.

Chapitre 1

Présentation du cadre de projet

1.1 Introduction

Roquette est un leader mondial dans la transformation des végétaux, notamment du maïs, ...

en ingrédients et solutions destinés aux secteurs de l'alimentation, de la nutrition et des biotechnologies. Soucieuse de garantir la qualité et la sécurité de ses produits, l'entreprise accorde une importance capitale à la détection des contaminants dans ses matières premières, en particulier l'**aflatoxine**, une toxine naturelle pouvant affecter le maïs et représenter un risque important pour la santé humaine et animale.

Cependant, le procédé actuel de contrôle, qui impose l'envoi systématique d'échantillons en laboratoire pour analyse, se révèle à la fois **long**, **coûteux** et peu flexible face aux besoins opérationnels du terrain. Ces contraintes retardent la prise de décision et limitent l'efficacité du processus d'approvisionnement.

C'est dans ce contexte qu'est né le **projet ADOM**, dont l'objectif principal est de concevoir et de mettre en œuvre une solution innovante, basée sur l'intelligence artificielle et l'analyse d'images, afin de permettre une détection rapide et fiable de l'aflatoxine directement chez les fournisseurs.

Ainsi, ce projet poursuit plusieurs objectifs :

- Optimiser la chaîne d'approvisionnement ;
- Accélérer la prise de décision lors des achats de grains ;
- Renforcer le contrôle qualité ;
- Inscrire l'entreprise dans une démarche d'innovation technologique durable.

1.2 Présentation de l'entreprise

Historique et identité

Roquette Frères est une entreprise familiale fondée en 1933 par **Dominique et Germain Roquette**. Elle s'est spécialisée dans la transformation de matières premières d'origine végétale, fournissant des ingrédients de haute qualité pour l'alimentation, la pharmacie, la nutrition animale et l'industrie. Avec près d'un siècle d'expérience, Roquette s'impose aujourd'hui comme un acteur mondial de référence dans son domaine.

Domaines d'activité

1. **Alimentation** : Roquette propose des protéines végétales, des fibres, des amidons et des polyols, permettant d'enrichir la qualité nutritionnelle des produits alimentaires.
2. **Pharmacie et Santé** : Développement d'excipients et ingrédients actifs répondant aux normes strictes de l'industrie pharmaceutique.
3. **Nutrition et Santé animale** : Fourniture de solutions pour le bien-être et la performance des animaux d'élevage.
4. **Industrie** : Conception d'ingrédients pour divers usages industriels : adhésifs, produits chimiques, textile, cosmétique ou fermentation.

Présence internationale


Roquette est présente dans plus de **100 pays**, avec **30 sites industriels** et un réseau mondial de centres de recherche et développement. Cette implantation lui permet de répondre efficacement aux attentes des clients sur tous les continents, avec une capacité d'innovation forte.

Engagements RSE

Fidèle à sa devise « *Offering the best of nature* », Roquette place le développement durable au cœur de sa stratégie. L'entreprise s'est engagée à réduire de **25 %** ses émissions de gaz à effet de serre d'ici 2030, illustrant une volonté claire de contribuer à une industrie plus responsable et respectueuse de l'environnement.

Présence globale

Aujourd'hui, l'entreprise est implantée dans plus de 100 pays, avec un total de 30 sites répartis à travers le monde.



images/les sites de lent.png

Figure 1.1 – Présence mondiale de Roquette Frères

Au-delà de l'excellence produit, la devise *Offering the best of nature* reflète une volonté d'adopter des pratiques respectueuses de l'environnement et socialement responsables.

1.3 Présentation de l'équipe Data & Advanced Analytics

Le rôle de l'équipe **Data & Advanced Analytics** est d'aider l'entreprise et ses collaborateurs à exploiter pleinement le potentiel et la valeur de leurs données. Cette équipe est divisée en deux sous-équipes principales : **Data Analysis & Data Science (DADS)** et **Data Governance & Data Platform**.

images/organigramme de l'entreprise.png

Figure 1.2 – Organigramme de l'équipe Data & Advanced Analytics

Équipe DADS (Data Analysis & Data Science)

L'équipe **DADS** se concentre sur l'analyse de données et la science des données :

- **DS Manager** : coordination, supervision des projets et appui aux décisions stratégiques ;

- **Data Analyst** : valorisation des données via des tableaux d'analyse et accompagnement des métiers ;
- **Data Scientist** : modélisation prédictive et analyses avancées.

1.4 Présentation du projet

1.4.1 Étude de l'existant

Actuellement, la détection du niveau d'**aflatoxine** dans les grains de maïs repose sur un processus manuel, long et coûteux. En Inde, lors de chaque transaction commerciale, les acheteurs de maïs de l'entreprise **Roquette Frères** prélèvent des échantillons directement chez les fournisseurs et les envoient dans des laboratoires spécialisés. Ces analyses permettent de mesurer la concentration d'aflatoxine, mais elles nécessitent plusieurs jours et entraînent des coûts logistiques importants. De plus, ce processus retarde les décisions d'achat et complique la traçabilité.

1.4.2 Problématique

L'**aflatoxine** est une toxine produite par certaines moisissures (comme *Aspergillus flavus* et *Aspergillus parasiticus*) qui se développent sur les cultures de maïs et d'autres céréales. Elle représente un danger majeur pour la santé humaine et animale, car elle peut contaminer la chaîne alimentaire et provoquer des intoxications graves, voire des cancers en cas d'exposition chronique.

Pour Roquette, la problématique est double :

- **Assurer la qualité et la sécurité sanitaire** des matières premières (maïs) dès le point d'achat ;
- **Réduire les coûts et délais** liés aux analyses en laboratoire, tout en permettant aux acheteurs sur le terrain de prendre une décision immédiate (accepter ou rejeter un lot).

En l'absence de technologie simple et rapide, l'entreprise est confrontée à un risque économique, sanitaire et réglementaire.

1.4.3 Solution proposée

Afin de répondre à ces enjeux, le projet vise à développer une **application mobile intégrant un modèle d'intelligence artificielle** basé sur la vision par ordinateur et le **Deep Learning**. L'application permettra aux acheteurs de :

- Prendre une photo des grains de maïs directement sur le marché ;
- Envoyer l'image au modèle IA (intégré ou via une API) ;
- Obtenir une **classification instantanée** du niveau d'aflatoxine (acceptable / non acceptable, ou par bande de risque) ;
- Décider immédiatement du sort du lot (achat, rejet, ou contrôle complémentaire).

Les principaux bénéfices attendus sont :

- **Rapidité** : un diagnostic en quelques secondes au lieu de plusieurs jours ;

- **Réduction des coûts** : moins d’analyses en laboratoire et de logistique ;
- **Accessibilité terrain** : utilisation directe par les acheteurs de maïs en Inde ;
- **Amélioration de la traçabilité et de la sécurité alimentaire.**

Ce projet s’inscrit dans une stratégie plus large de **digitalisation des processus qualité et achats** de Roquette Frères, et constitue une étape importante vers une chaîne d’approvisionnement plus efficace et sécurisée.

1.5 Conduite du projet

Le processus de développement constitue un facteur clé dans la réussite d'un projet logiciel, car il permet de structurer les différentes phases de travail et de suivre l'évolution de la solution dans un cadre clair et itératif. Afin de répondre efficacement aux exigences de performance, de fiabilité et de rapidité de notre système de détection d'aflatoxine, nous avons adopté une méthode de gestion de projet agile : **Scrum**.

1.5.1 Méthodologie Scrum adaptée au projet

Le projet a été conduit en s'inspirant de la méthodologie agile Scrum, tout en l'adaptant au cadre d'un stage de fin d'études. Cette approche a permis de structurer les différentes phases du développement de manière itérative, tout en assurant un suivi régulier avec le tuteur et l'équipe Data Science de Roquette.

L'organisation du travail s'est faite sous forme de **sprints de deux semaines**, au terme desquels un ensemble de fonctionnalités était livré, testé et validé. Chaque sprint visait la construction progressive d'un composant clé de la solution (préparation du dataset, entraînement du modèle CNN, API Flask, application mobile Flutter, pipeline CI/CD).

1.5.2 Modalités de suivi

Le suivi du projet a reposé sur une organisation régulière de réunions permettant d'assurer à la fois un accompagnement technique et un pilotage global :

- **Réunions hebdomadaires avec l'équipe Data** pour suivre l'avancement des projets en cours et garantir l'alignement avec les standards techniques.
- **Revue ADOM hebdomadaire avec le Lead DevOps** afin de traiter les aspects techniques du projet (intégration, CI/CD, déploiement) et lever rapidement les blocages.
- **Réunions bimensuelles avec le tuteur de stage** pour présenter l'avancement (réalisations, tâches à venir), discuter des difficultés rencontrées et recueillir des retours constructifs.

1.5.3 Éléments Scrum appliqués

Même adapté à un cadre académique, plusieurs principes fondamentaux de Scrum ont été conservés :

- **Sprint Planning** : définition des objectifs et livrables en début de chaque cycle.
- **Sprint Review** : démonstration et validation en fin de sprint.

1.5.4 Organisation des livrables

Grâce à cette approche itérative, le projet a progressé de manière structurée, avec l'intégration successive des composants suivants :

- Préparation et nettoyage du **dataset d'images de maïs** ;
- Entraînement du **modèle CNN** avec TensorFlow ;
- Développement d'une **API Flask** pour la prédiction ;
- Intégration du modèle dans l'**application mobile Flutter** ;
- Mise en place des **pipelines CI/CD** avec Azure DevOps.

Cette démarche agile a permis de rester réactif face aux retours, d'intégrer progressivement les différents modules techniques et de garantir une meilleure maîtrise du cycle de développement du projet ADOM.



Figure 1.3 – Cycle de la méthodologie agile Scrum appliqué au projet ADOM

1.6 Planning prévisionnel

Le travail a été structuré en **sprints de deux semaines**, permettant une progression incrémentale et itérative. Chaque phase du projet a été planifiée afin de garantir une bonne répartition des tâches et une cohérence entre les livrables. Le planning (figure ??) illustre les étapes principales :

- **Préparation de l'environnement** : installation des outils, configuration des environnements de développement et mise en place des dépôts Git ;
- **Conception** : définition de l'architecture, choix des technologies, et conception des modules principaux ;
- **Réalisation** : développement progressif des composants (modèle CNN, API Flask, application Flutter, CI/CD) réparti sur plusieurs sprints ;
- **Test et Validation** : phase de tests unitaires, intégration et validation fonctionnelle ;
- **Déploiement** : mise en production et finalisation du livrable, incluant les ajustements nécessaires.

images/planification de projet.png

Figure 1.4 – Planning prévisionnel du projet ADOM

1.7 Conclusion

Ce premier chapitre a permis de dresser un panorama de **Roquette**, de ses activités et de ses engagements, tout en soulignant le rôle central de la **Data & Advanced Analytics** dans ses projets d'innovation.

L'analyse de l'existant a mis en évidence les limites du processus actuel de détection de l'aflatoxine dans le maïs : lenteur, coûts élevés, dépendance aux laboratoires et manque de réactivité. Ces constats révèlent la nécessité de disposer d'une solution plus rapide, fiable et accessible directement sur le terrain.

Le projet **ADOM** s'inscrit dans cette perspective en proposant une application mobile intégrant un modèle de **Deep Learning**. Développée selon une approche **agile Scrum**, cette solution vise à accélérer la prise de décision, améliorer la traçabilité et réduire les coûts liés au contrôle qualité, tout en offrant une alternative innovante et opérationnelle au processus traditionnel.

Chapitre 2

Spécification des besoins

2.1 Introduction

Avant d’entamer la conception et le développement de toute solution technologique, il est indispensable d’identifier, de formaliser et de hiérarchiser l’ensemble des besoins auxquels cette solution devra répondre.

Dans le cadre du **projet ADOM**, cette démarche vise à doter les équipes d’achat de **Roquette** d’un outil mobile capable de détecter rapidement et fiablement l’aflatoxine dans les grains de maïs sur le terrain. Elle intègre :

- les contraintes opérationnelles (conditions environnementales, connectivité, formation des utilisateurs) ;
- les exigences techniques (temps de réponse, précision des analyses, compatibilité avec les équipements) ;
- la sécurité et la confidentialité des données ;
- l’intégration fluide avec les processus existants et l’architecture SI de Roquette.

Ce chapitre définit un **référentiel** commun : besoins **fonctionnels**, **non-fonctionnels**, contraintes et hypothèses. Il sert d’appui à la conception, à la recette et au déploiement.

2.2 Spécification des besoins fonctionnels

Détection en temps réel

- Détection/quantification de l'aflatoxine à partir d'une photo terrain.
- Résultats rapides (idéalement en moins de 3 minutes).

Catégorisation des échantillons

- Classement automatique en bandes : 0–30 ppb, 31–50 ppb, 51–75 ppb, 76–100 ppb, > 101 ppb.

Prise de décision immédiate

- Recommandation : *Conforme*, *À surveiller*, *Non conforme*.

Utilisation sur appareils mobiles

- Installation Android & iOS, expérience fluide sur terminaux modestes.

Fonctionnement offline / online

- Utilisable hors-ligne ; synchronisation automatique quand la connexion revient.

Saisie et gestion des informations

- Saisie fournisseur, origine, lot/véhicule, géolocalisation, etc.

Enregistrement et traçabilité

- Enregistrement local ; transfert vers serveur sécurisé pour archivage/traçabilité.

Sécurité et gestion des accès

- Authentification et contrôle d'accès ; respect du RGPD et des lois locales.

Ergonomie

- Interface intuitive, guidage étape par étape.

Gestion de la montée en charge

- Support des pics (récoltes) et d'un nombre d'utilisateurs croissant.

Reporting

- Rapports par lot/site/période, personnalisables et exportables.

Formation et support

- Guides, tutoriels vidéo, support technique pour une prise en main rapide.

2.3 Spécification des besoins non-fonctionnels

Performance

- Traitement d'un échantillon en ≤ 3 minutes.
- Comportement fluide et stable, y compris lors de pics multi-utilisateurs.

Précision et fiabilité

- Précision du modèle IA $> 80\%$; robustesse à des prises de vue variées.

Disponibilité et tolérance aux pannes

- Mode *offline* avec resynchronisation automatique.
- Haute disponibilité pendant les campagnes d'achat.

Compatibilité et portabilité

- Android/iOS, multiples tailles d'écran et versions ; compatible avec des terminaux d'entrée de gamme.

Sécurité et confidentialité

- Chiffrement des données en local et en transit, accès sécurisés, conformité RGPD et lois locales.

Scalabilité

- Montée en charge (utilisateurs/volumes) et ajout de modules sans refonte.

Maintenance et évolutivité

- Mises à jour aisées (modèle IA & fonctionnalités), idéalement à distance ; documentation à jour.

Ergonomie et accessibilité

- Interface claire, simple, multilingue ; peu de formation requise.

Conformité réglementaire

- Respect des normes (sécurité, qualité des données, traçabilité) et alignement RSE Roquette.

2.4 Présentation des cas d'utilisation

Cette section présente les différents cas d'utilisation liés au projet **ADOM**. Le diagramme UML ci-dessous illustre les interactions principales entre les acteurs (utilisateur et administrateur) et le système.

Diagramme de cas d'utilisation (Use Case Diagram)

Acteurs principaux :

- **Utilisateur** : capture et envoie des images de grains de maïs via l'application mobile.
- **Administrateur** : gère l'entraînement du modèle et la mise à jour de l'API.

Cas d'utilisation :

- S'inscrire / Se connecter via Firebase ;
- Uploader une image ;
- Lancer une prédiction ;
- Recevoir le résultat (classe + niveau de confiance) ;
- Mettre à jour et entraîner le modèle CNN (côté Administrateur).

images/use_case_adom.png

Figure 2.1 – Diagramme de cas d'utilisation du projet ADOM

Chapitre 3

Conception du système

3.1 Introduction

La conception constitue une étape clé entre la spécification des besoins et l'implémentation. Elle permet de représenter, sous forme de modèles, la structure et le comportement du système afin de garantir sa cohérence, sa robustesse et sa maintenabilité.

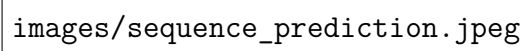
Dans le cadre du projet **ADOM**, la conception se décline en deux volets complémentaires :

- la **modélisation dynamique** décrivant les comportements, scénarios et flux de données (diagrammes de séquence, d'activités et de cas d'utilisation) ;
- la **modélisation statique** mettant en avant la structure du système (diagramme de classes, dictionnaire et modèle relationnel) ainsi que les architectures logicielle et matérielle de déploiement.

3.2 Modélisation dynamique

3.2.1 Diagramme de séquence

Le diagramme de séquence suivant représente le scénario de prédiction, depuis l'authentification jusqu'à l'affichage du résultat.



images/sequence_prediction.jpeg

Figure 3.1 – Diagramme de séquence — Flux de prédiction

3.2.2 Diagramme d'activités

Ce diagramme décrit le flux d'activités de la prédiction d'aflatoxine.

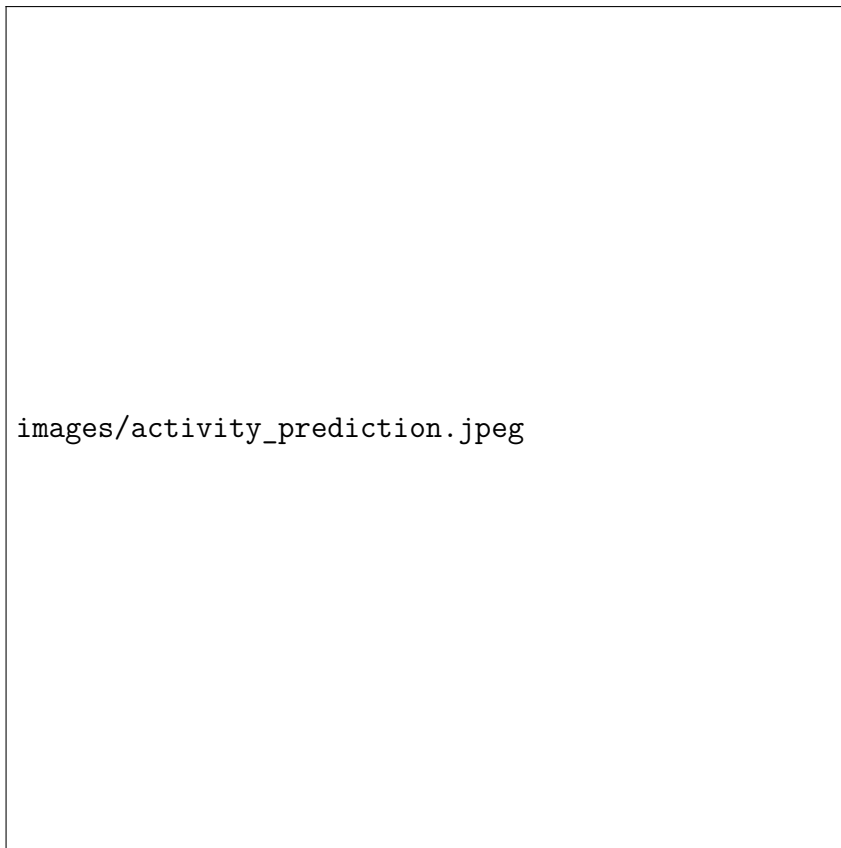


Figure 3.2 – Diagramme d’activités — Processus de prédiction

3.3 Modélisation statique

3.3.1 Diagramme de classes

Le diagramme de classes présente les principales entités de l'application (mobile, API, modèle IA et stockage) et leurs relations.



Figure 3.3 – Diagramme de classes — Application ADOM

3.3.2 Modèle relationnel

Le modèle relationnel dérivé est représenté ci-dessous :



Figure 3.4 – Modèle relationnel de la base de données

3.4 Architecture du système

3.4.1 Architecture logicielle

L'architecture logicielle repose sur une approche modulaire intégrant :

- **Application mobile (Flutter + Firebase)** : authentification, capture et envoi d'images ;
- **API Flask + TFLite** : traitement et exécution des prédictions ;
- **Module IA (CNN + Azure ML)** : entraînement, évaluation et export du modèle ;
- **CI/CD (Azure DevOps)** : automatisation des déploiements ;
- **Stockage (Azure Blob Storage)** : gestion des datasets et modèles.

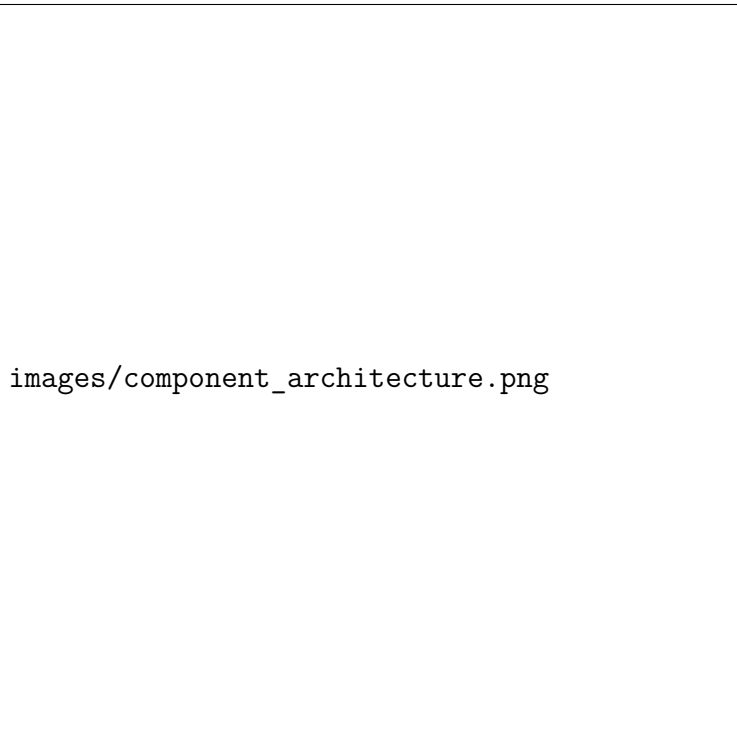


Figure 3.5 — Architecture logicielle — composants principaux

Vue de déploiement — Azure

Le déploiement cible s'appuie sur des services managés Azure : *Azure Container Apps* pour l'API, *Container Registry* pour les images, *Azure ML* pour l'entraînement et *Blob Storage* pour les données.

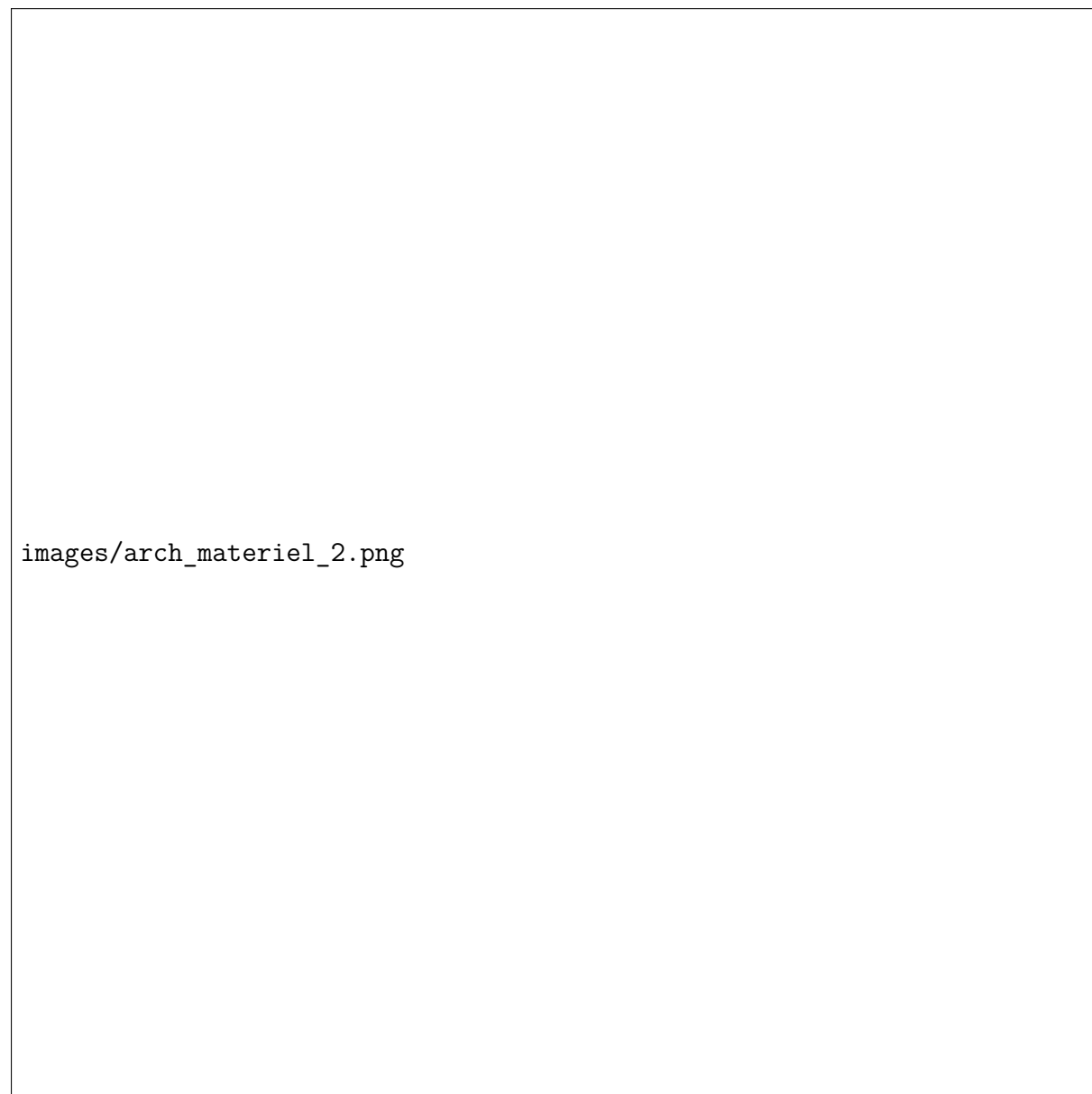


Figure 3.6 — Architecture matérielle — Déploiement sur Azure
(entraînement, registre de modèles, images Docker,
déploiement API)

3.5 Outils et technologies utilisés

Les choix technologiques du projet ADOM sont variés en raison de sa nature hybride (mobile, cloud, intelligence artificielle). Chaque outil a été sélectionné pour répondre à un besoin précis et garantir la performance, la sécurité et la maintenabilité de la solution.

3.5.1 Python

Python est un langage de programmation polyvalent largement utilisé en data science et apprentissage automatique. Dans ADOM, il a servi à écrire les scripts de prétraitement, l'entraînement du modèle d'IA et la logique serveur via l'API.

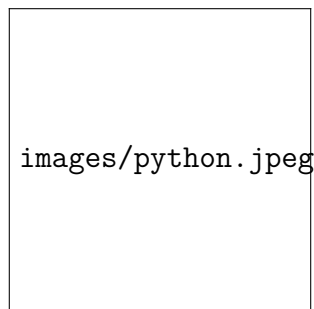


Figure 3.7 – Logo de Python

3.5.2 TensorFlow / Keras

TensorFlow est une bibliothèque de Deep Learning open source développée par Google. Keras, son API haut niveau, simplifie la création et l'entraînement de réseaux de neurones. Dans ADOM, TensorFlow/Keras ont servi à construire et exporter le modèle de détection d'aflatoxine.

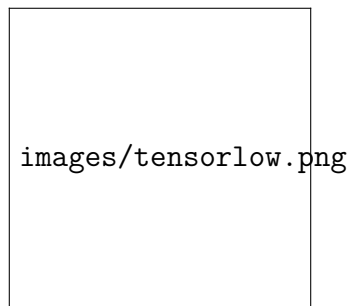


Figure 3.8 – Logo de TensorFlow

3.5.3 Flask

Flask est un micro-framework web Python utilisé pour créer des API REST légères. Il a permis d'exposer l'API de classification, qui reçoit une image et retourne la prédiction du modèle.

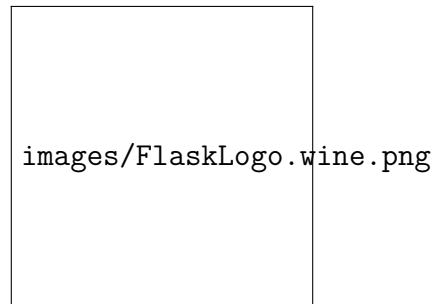


Figure 3.9 – Logo de Flask

3.5.4 Flutter et Dart

Flutter est un framework multiplateforme open-source développé par Google pour le développement mobile, utilisant Dart comme langage de programmation. Dans ADOM, Flutter a servi à créer l'application `adomApp` pour Android et iOS, offrant une interface utilisateur fluide et moderne.

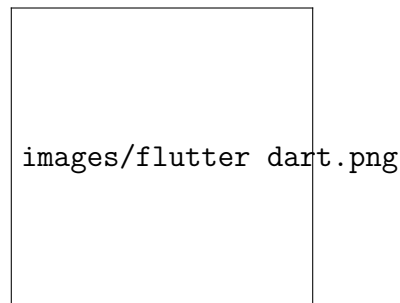
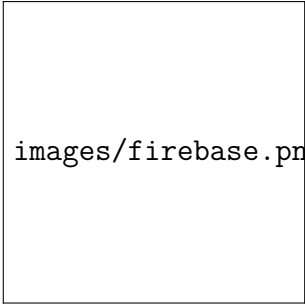


Figure 3.10 – Logo de Flutter (utilise Dart comme langage)

3.5.5 Firebase Authentication

Firebase est une plateforme de Google offrant des services backend prêts à l'emploi. Dans ADOM, *Firebase Auth* a été utilisé pour gérer l'authentification sécurisée des utilisateurs.



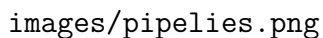
images/firebase.png

Figure 3.11 – Logo de Firebase

3.5.6 Pipelines CI/CD

Le cycle de développement et de déploiement continu a été automatisé avec **Azure DevOps** :

- **Pipeline de build** : compilation du code, exécution des tests unitaires, création d'une image Docker de l'API.
- **Pipeline de release** : push de l'image vers Azure Container Registry puis déploiement automatique sur Azure Container Apps.
- **Suivi** : journaux centralisés et monitoring via Azure Monitor pour garantir disponibilité et fiabilité.



images/pipelies.png

Figure 3.12 – Exemple de pipeline CI/CD sur Azure DevOps pour l'API ADOM

3.5.7 Azure Machine Learning (Azure ML)

Azure ML est un service cloud de Microsoft permettant de gérer tout le cycle de vie d'un modèle IA (entraînement, suivi, versioning et déploiement). Dans ADOM, il a servi à entraîner le modèle, suivre les métriques et stocker les artefacts.

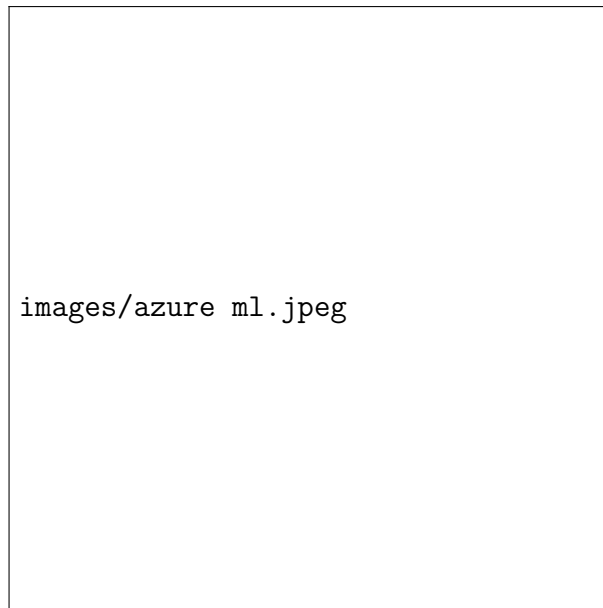


Figure 3.13 – Logo d’Azure Machine Learning

3.5.8 Azure Active Directory (Azure AD)

Azure AD est le service d’identité de Microsoft, assurant la gestion des accès et la sécurité. Il a été utilisé pour sécuriser l’accès aux ressources cloud et à l’API ADOM (Managed Identity).

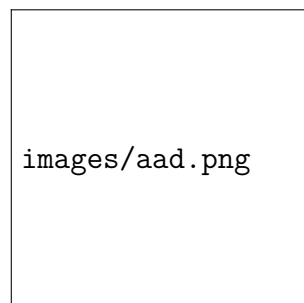


Figure 3.14 – Logo d’Azure Active Directory

3.5.9 Azure DevOps

Azure DevOps est une plateforme d’intégration et déploiement continu (CI/CD). Elle a permis d’automatiser les pipelines, la construction d’images Docker et le déploiement dans Azure Container Apps.




images/devops.png

Figure 3.15 – Logo d’Azure DevOps

3.5.10 Docker

Docker est une plateforme de conteneurisation qui permet d’emballer une application avec toutes ses dépendances dans un environnement portable. Dans ADOM, Docker a été utilisé pour exécuter l’API et préparer les images pour le déploiement dans Azure.



images/docker.jpeg

Figure 3.16 – Logo de Docker

3.5.11 Visual Studio Code (VS Code)

VS Code est un IDE léger et extensible utilisé pour écrire le code Python, Flutter/Dart, et configurer les pipelines.



images/vscode.png

Figure 3.17 – Logo de Visual Studio Code

3.5.12 Jira

Jira est un outil de gestion de projet agile qui permet de suivre les tâches, bugs et sprints. Dans ADOM, Jira a servi à gérer le backlog et à organiser le suivi des développements.

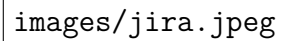
The image is a placeholder for the Jira logo, represented by the text 'images/jira.jpeg' inside a square frame.

Figure 3.18 – Logo de Jira

3.5.13 Microsoft Teams

Microsoft Teams est un outil de communication et de collaboration. Dans ADOM, il a servi aux réunions régulières, à la communication entre les équipes et au partage de documents.

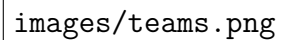
The image is a placeholder for the Microsoft Teams logo, represented by the text 'images/teams.png' inside a square frame.

Figure 3.19 – Logo de Microsoft Teams

3.6 Benchmarking fonctionnel

Le benchmarking fonctionnel est une étape essentielle avant de retenir une pile technologique. L'objectif n'est pas de copier des solutions existantes, mais d'identifier les bonnes pratiques et de choisir les outils les plus adaptés aux besoins du projet ADOM. Dans ce cadre, une étude comparative a été menée sur plusieurs technologies de développement mobile, backend, IA et cloud.

Les critères étudiés incluent :

- **Compatibilité multi-plateformes** (Android/iOS/Web pour le mobile, support cloud pour l'IA) ;
- **Facilité d'intégration** avec les pipelines DevOps et les services de sécurité ;
- **Performance et scalabilité** (entraîner et déployer des modèles IA rapidement) ;
- **Coût et disponibilité** (open-source vs solutions propriétaires) ;
- **Documentation et communauté** (soutien en cas de problème, rapidité de montée en compétence).

Après analyse des alternatives, les choix finaux ont été guidés par la nécessité d'avoir une solution complète, interopérable et alignée avec l'écosystème Microsoft déjà présent chez Roquette.

Domaine	Choix retenu	Alternative	Justification
Dév. mobile	Flutter/Dart	React Native	Flutter offre un rendu natif homogène et une base de code unique Android/iOS.
Framework IA	TensorFlow/Keras	PyTorch	Export et déploiement plus simples, intégration native avec Azure ML.
API Backend	Flask (Python)	FastAPI	Simplicité et légèreté pour une API REST ; suffisante pour nos besoins.
Authentification	Firebase Auth	Azure AD B2C	Mise en place rapide sur mobile, SDK Flutter officiel.
CI/CD	Azure DevOps	GitHub Actions	Alignement avec l'écosystème Azure + service connections intégrées.
Cloud IA	Azure ML	AWS SageMaker	Standard interne chez Roquette, intégration directe avec DevOps et Blob Storage.
Conteneurisation	Docker	Podman	Solution la plus mature et largement supportée dans les pipelines CI/CD.
IDE	VS Code	PyCharm	Léger, multiplateforme, support natif Python et Flutter.
Collaboration	Microsoft Teams	Slack	Déjà utilisé par l'entreprise, intégration facile avec Azure et DevOps.
Gestion projet	Jira	Trello	Plus adapté au suivi agile (sprints, backlog) et au travail en équipe.

Table 3.1 – Tableau comparatif des technologies

Chapitre 4

Implémentation et résultats

4.1 Introduction

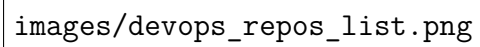
Ce chapitre détaille la mise en œuvre concrète du projet **ADOM**, depuis la préparation des données jusqu'au déploiement de l'application mobile, en passant par l'entraînement du modèle d'intelligence artificielle et la création de l'API. Chaque étape est illustrée par des captures d'écran et des extraits de code pertinents, mettant en lumière les choix techniques et les résultats obtenus.

4.2 Préparation de l'environnement et des données

4.2.1 Configuration de l'environnement de développement

La première étape a consisté à mettre en place un environnement de développement robuste et reproductible.

- **Azure DevOps** : Création d'un nouveau projet pour centraliser la gestion du code, des pipelines et des artefacts.
- **Dépôts Git** : Initialisation de deux dépôts distincts : `adom-api` pour le backend et le modèle IA, et `adom-app` pour l'application mobile Flutter.
- **IDE** : Utilisation de Visual Studio Code avec des extensions spécifiques pour Python, Docker et Flutter.

The image is a screenshot of the Azure DevOps interface, specifically the 'Repos' section. It displays a list of repositories. The first repository is 'my-repo', which is a public repository with a green checkmark icon. Below it, there are several other repositories, some of which are private (indicated by a lock icon) and some are public. The list includes repository names, their visibility status, and the number of files. The interface is clean and modern, with a white background and blue accents.

images/devops_repos_list.png

Figure 4.1 – Liste des dépôts sur Azure DevOps

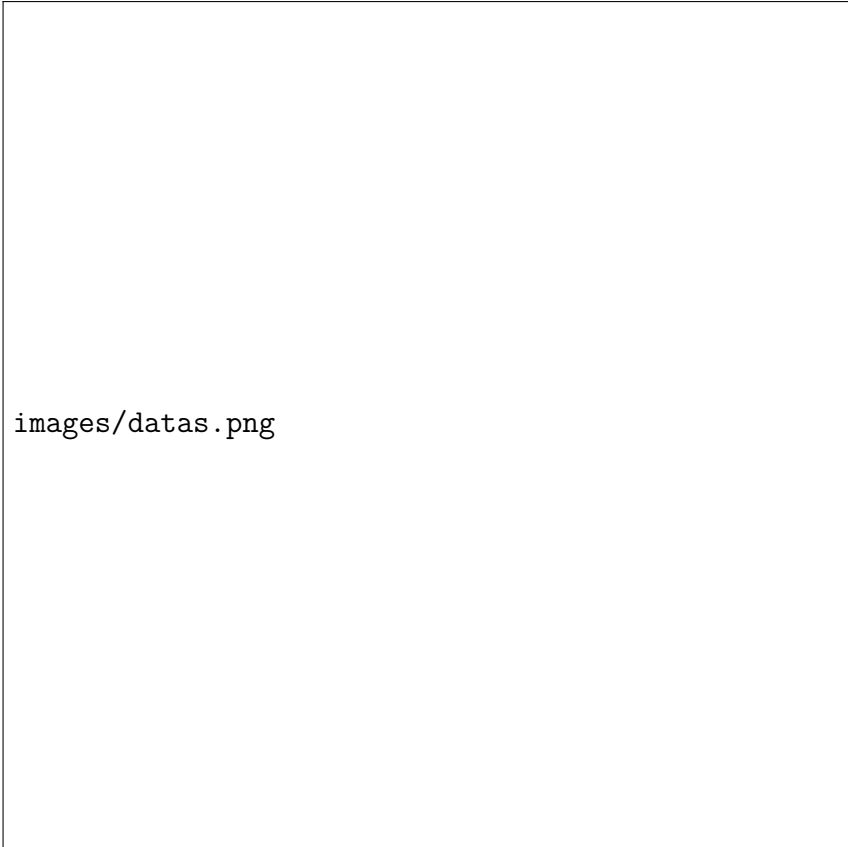


Figure 4.2 – Arborescence du dépôt adom-api

4.2.2 Collecte et préparation des données

Le succès d'un modèle de Deep Learning dépend fortement de la qualité et de la quantité des données d'entraînement.

- **Source** : Les images de grains de maïs ont été fournies par les équipes de Roquette en Inde.
- **Volume** : Le dataset initial comprenait plusieurs milliers d'images, chacune associée à une étiquette correspondant à une plage de concentration d'aflatoxine (en ppb).
- **Nettoyage et augmentation** : Les données ont été nettoyées pour supprimer les images non pertinentes. Des techniques d'augmentation de données (rotation, zoom, retournement) ont été appliquées pour enrichir le dataset et améliorer la robustesse du modèle.



images/datas.png

Figure 4.3 – Exemples d’images du dataset

4.3 Développement et entraînement du modèle d’IA

4.3.1 Architecture du modèle

Un modèle de **Réseau de Neurones Convolutif (CNN)** a été choisi pour sa performance reconnue en classification d’images. L’architecture retenue est une version adaptée de modèles classiques comme VGG ou ResNet, optimisée pour un bon compromis entre précision et temps d’inférence.

4.3.2 Entraînement sur Azure Machine Learning

L’entraînement a été réalisé sur **Azure Machine Learning** pour bénéficier de sa puissance de calcul et de ses outils de suivi.

- **Script d’entraînement** : Un script Python utilisant TensorFlow/Keras a été développé pour charger les données, construire le modèle et lancer l’entraînement.
- **Suivi des métriques** : Azure ML a permis de suivre en temps réel des métriques clés comme la précision (*accuracy*) et la perte (*loss*) sur les ensembles d’entraînement et de validation.

- **Versioning des modèles** : Chaque version du modèle entraîné a été enregistrée dans le registre de modèles d’Azure ML, facilitant la traçabilité et le déploiement de la meilleure version.

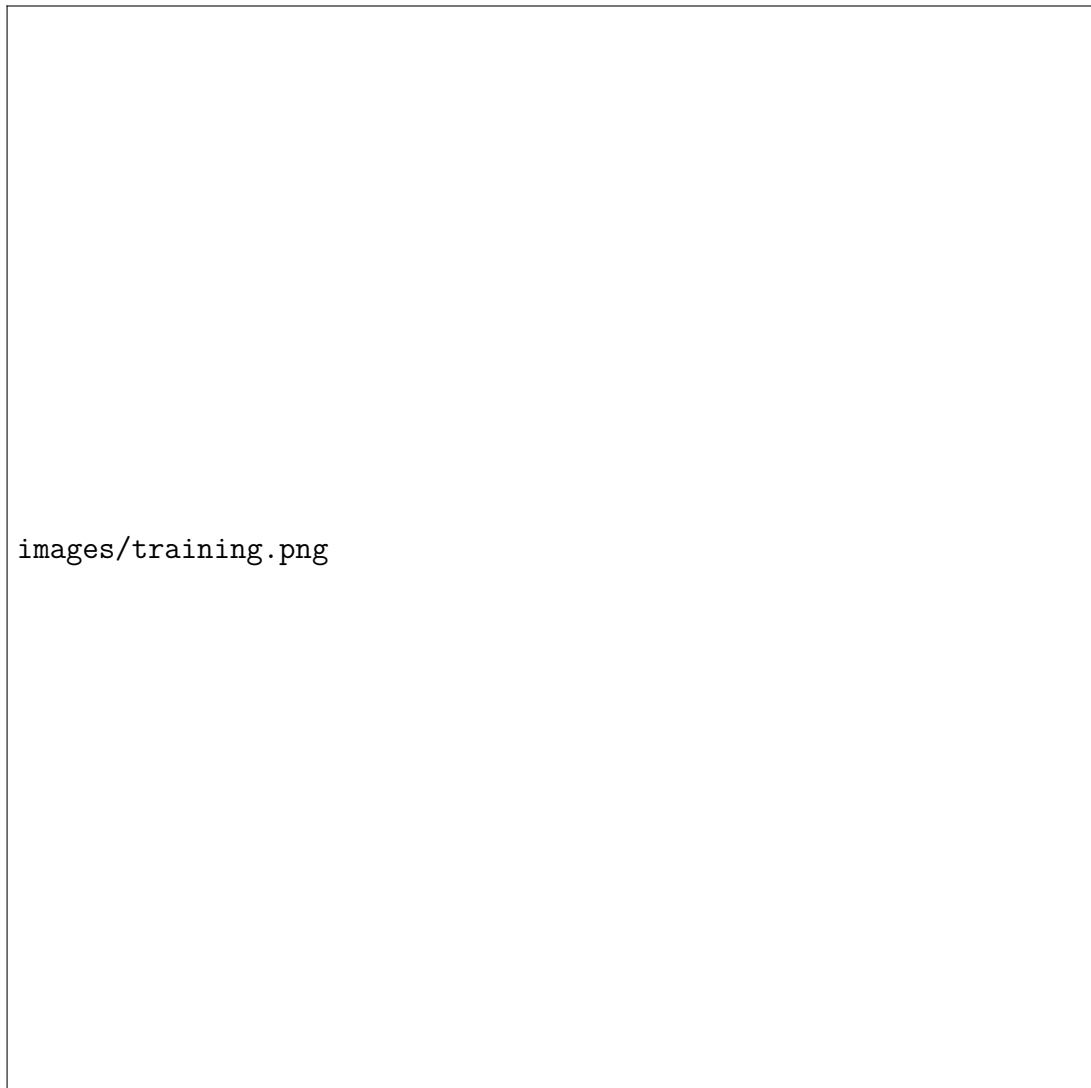
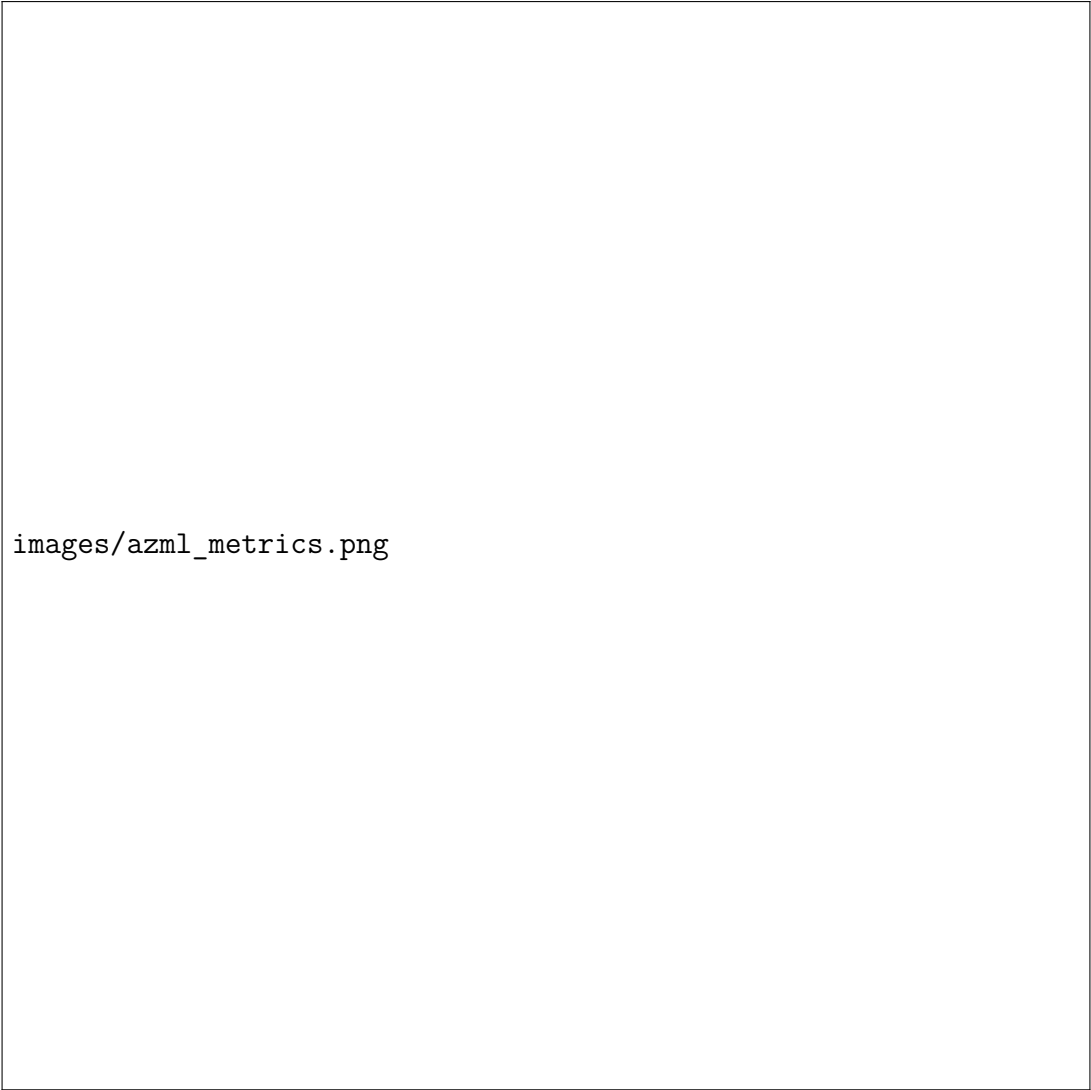



Figure 4.4 – Courbes d’entraînement (précision et perte) sur Azure ML



images/azml_metrics.png

Figure 4.5 – Suivi des métriques sur Azure ML



images/azml_models.png

Figure 4.6 – Registre des modèles sur Azure ML

4.4 Développement de l'API et de l'application mobile

4.4.1 Création de l'API Flask

Une API REST a été développée avec Flask pour servir de pont entre l'application mobile et le modèle d'IA.

- **Endpoint de prédiction** : Un endpoint `/predict` a été créé. Il accepte une requête POST contenant une image, la prétraite, et la passe au modèle TFLite pour obtenir une prédiction.
- **Conteneurisation Docker** : L'API a été encapsulée dans une image Docker pour garantir un déploiement cohérent et reproductible.

4.4.2 Développement de l'application mobile avec Flutter

L'application `adomApp` a été développée avec Flutter pour offrir une expérience utilisateur moderne et multiplateforme.

- **Authentification** : Intégration de Firebase Authentication pour une connexion sécurisée.
- **Interface utilisateur** : Conception d'écrans clairs et intuitifs pour la connexion, la prise de photo, et l'affichage des résultats.
- **Communication avec l'API** : Implémentation de la logique pour envoyer l'image capturée à l'API Flask et afficher la prédiction retournée.



Figure 4.7 – Écran de connexion



Figure 4.8 – Écran de connexion rempli

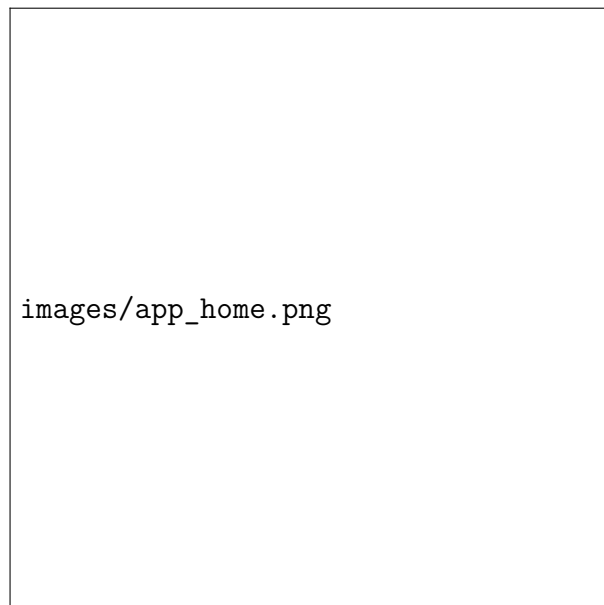
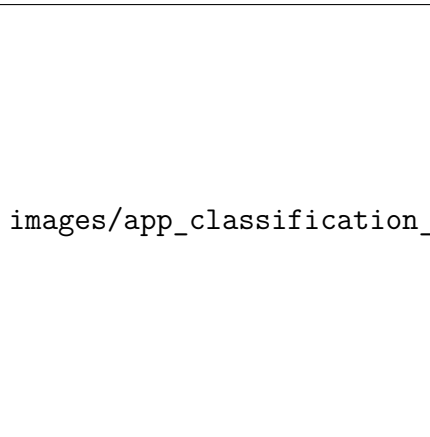
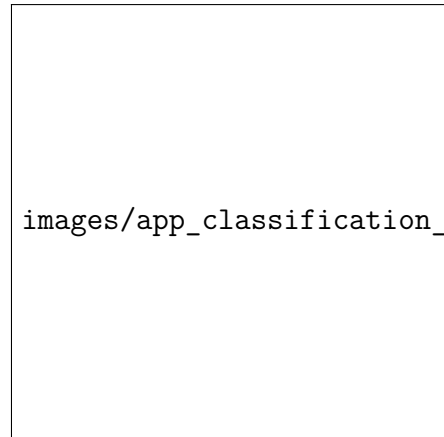


Figure 4.9 – Page d'accueil de l'application



images/app_classification_empty.png

Figure 4.10 – Écran de classification avant sélection



images/app_classification_selected.png

Figure 4.11 – Écran de classification après sélection



images/app_result.png

Figure 4.12 – Affichage du résultat de la prédiction



images/app_result2.png

Figure 4.13 – Autre exemple de résultat

4.5 Déploiement et intégration continue (CI/CD)

4.5.1 Pipelines sur Azure DevOps

L'automatisation a été au cœur du processus de déploiement.

— **Pipeline de Build (CI)** : À chaque push sur la branche principale du dépôt `adom-api`, un pipeline se déclenche automatiquement pour :

1. Construire l'image Docker de l'API.
2. Pousser l'image vers Azure Container Registry.

- **Pipeline de Release (CD)** : Une fois l'image poussée, un pipeline de déploiement prend le relais pour mettre à jour l'instance Azure Container Apps avec la nouvelle version de l'image.

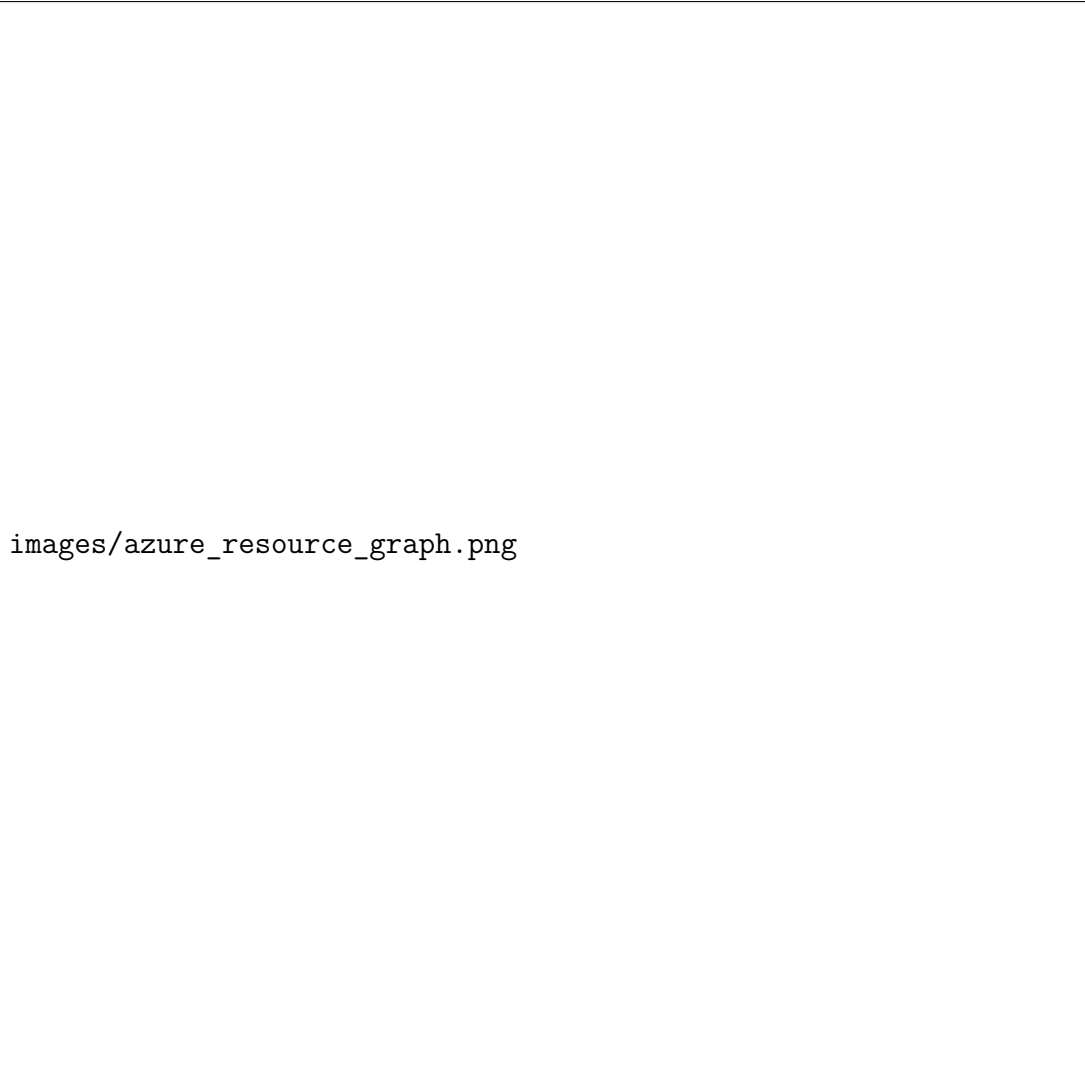


Figure 4.14 – Exemple de configuration de pipeline YAML sur Azure DevOps

4.5.2 Déploiement sur Azure

La solution est entièrement hébergée sur des services managés Azure.

- **Azure Container Apps** : Héberge l'API conteneurisée, offrant une scalabilité automatique et une gestion simplifiée.
- **Azure Container Registry** : Stocke les images Docker de l'API.
- **Azure Monitor** : Utilisé pour la surveillance des logs et des performances de l'API en production.



images/azure_resource_graph.png

Figure 4.15 – Graphe des ressources Azure utilisées pour le projet

4.6 Conclusion

Ce chapitre a détaillé le parcours complet de l'implémentation du projet ADOM. En partant d'un besoin métier clair, nous avons mis en place une solution technologique complète, alliant intelligence artificielle, développement mobile et infrastructure cloud. L'approche agile et l'automatisation via les pipelines CI/CD ont permis de livrer une solution fonctionnelle, robuste et prête à être testée en conditions réelles. Les résultats obtenus, tant en termes de performance du modèle que de fluidité de l'application, sont prometteurs et ouvrent la voie à une transformation significative du processus de contrôle qualité chez Roquette.

Conclusion générale

Le projet de fin d'études, mené au sein de l'entreprise **Roquette Frères**, a permis de répondre à une problématique métier concrète : la détection rapide et fiable du niveau d'aflatoxine dans les grains de maïs. En partant d'un processus manuel, long et coûteux, nous avons conçu et développé une solution innovante, **ADOM**, qui s'appuie sur les technologies les plus récentes en matière d'intelligence artificielle, de développement mobile et de cloud computing.

Sur le plan technique, ce projet a été l'occasion de maîtriser un écosystème technologique riche et varié. L'utilisation de **Python** et **TensorFlow** pour la création du modèle de Deep Learning, de **Flask** pour le développement de l'API, et de **Flutter** pour l'application mobile a permis de construire une solution modulaire et performante. L'hébergement sur **Microsoft Azure** et l'automatisation des déploiements via les pipelines **CI/CD** d'Azure DevOps ont garanti la robustesse, la scalabilité et la maintenabilité du système.

Au-delà des aspects techniques, ce projet a été une expérience humaine et professionnelle enrichissante. L'immersion au sein de l'équipe Data de Roquette et l'application de la méthodologie agile **Scrum** ont permis de développer des compétences clés en gestion de projet, en communication et en travail d'équipe. Les échanges réguliers avec les tuteurs et les experts métiers ont été essentiels pour aligner la solution sur les besoins réels des utilisateurs finaux.

Les résultats obtenus sont très encourageants. L'application mobile est fonctionnelle et le modèle d'IA atteint une précision prometteuse, ouvrant la voie à des tests sur le terrain à plus grande échelle. Le projet ADOM a non seulement atteint ses objectifs initiaux, mais il a également posé les bases d'une plateforme évolutive qui pourra, à l'avenir, intégrer de nouvelles fonctionnalités ou être adaptée à d'autres cas d'usage au sein de l'entreprise.

En perspective, plusieurs pistes d'amélioration peuvent être envisagées pour enrichir la solution :

- **Amélioration continue du modèle** : Intégrer davantage de données et explorer des architectures de modèles plus complexes pour augmenter encore la précision.
- **Déploiement on-device** : Embarquer le modèle TFLite directement dans l'application mobile pour un fonctionnement 100% hors-ligne.

- **Tableau de bord de suivi** : Développer une interface web pour visualiser les statistiques d'utilisation et les résultats des prédictions à l'échelle de l'entreprise.

Ce projet représente ainsi un socle opérationnel pour une plateforme de détection d'aflatoxine performante, évolutive et alignée avec les enjeux actuels de la sécurité alimentaire, tout en ouvrant la voie vers des solutions intelligentes intégrant IA, mobilité et objets connectés.

Annexes

Annexe A : Code source de l'API Flask

```
from flask import Flask, request, jsonify
import tensorflow as tf
import numpy as np
from PIL import Image
import io

app = Flask(__name__)

# Charger le modèle TFLite
interpreter = tf.lite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Définir les classes
classes = ['0-30', '31-50', '51-75', '76-100', '>101']

def preprocess_image(image_bytes):
    img = Image.open(io.BytesIO(image_bytes)).convert('RGB')
    img = img.resize((224, 224))
    img_array = np.array(img, dtype=np.float32)
    img_array = np.expand_dims(img_array, axis=0)
    return img_array

@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['file']
```

```

if file.filename == '':
    return jsonify({'error': 'No selected file'}), 400

try:
    img_bytes = file.read()
    input_data = preprocess_image(img_bytes)

    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()

    output_data = interpreter.get_tensor(output_details[0]['index'])
    predicted_class_index = np.argmax(output_data)
    predicted_class = classes[predicted_class_index]
    confidence = float(np.max(output_data))

    return jsonify({
        'prediction': predicted_class,
        'confidence': confidence
    })
except Exception as e:
    return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)

```

Annexe B : Code source du widget de classification (Flutter)

```

import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

class ClassificationScreen extends StatefulWidget {
  @override
  _ClassificationScreenState createState() => _ClassificationScreenState();
}

```

```

class _ClassificationScreenState extends State<ClassificationScreen> {
  File? _image;
  String _prediction = '';
  double _confidence = 0.0;
  bool _isLoading = false;

  Future<void> _pickImage(ImageSource source) async {
    final pickedFile = await ImagePicker().pickImage(source: source);
    if (pickedFile != null) {
      setState(() {
        _image = File(pickedFile.path);
        _prediction = '';
        _confidence = 0.0;
      });
    }
  }

  Future<void> _classifyImage() async {
    if (_image == null) return;

    setState(() {
      _isLoading = true;
    });

    var request = http.MultipartRequest(
      'POST',
      Uri.parse('YOUR_API_ENDPOINT/predict'), // Remplacez par votre URL
    );
    request.files.add(await http.MultipartFile.fromPath('file', _image!.path));

    try {
      var response = await request.send();
      if (response.statusCode == 200) {
        var responseData = await response.stream.bytesToString();
        var decodedData = json.decode(responseData);
        setState(() {
          _prediction = decodedData['prediction'];
          _confidence = decodedData['confidence'];
        });
      }
    }
  }
}

```



```

        });
    } else {
        setState(() {
            _prediction = 'Erreur de classification';
        });
    }
} catch (e) {
    setState(() {
        _prediction = 'Erreur de connexion';
    });
} finally {
    setState(() {
        _isLoading = false;
    });
}
}

@override
Widget build(BuildContext context) {
    // ... UI Code ...
}
}

```

Annexe C : Configuration du pipeline de build (YAML)

```

trigger:
- main

pool:
    vmImage: 'ubuntu-latest'

variables:
    dockerRegistryServiceConnection: 'YOUR_SERVICE_CONNECTION'
    imageRepository: 'adom-api'
    containerRegistry: 'YOUR_CONTAINER_REGISTRY.azurecr.io'
    tag: '$(Build.BuildId)'

steps:
- task: Docker@2

```

```
displayName: Build and push an image to container registry
inputs:
  command: buildAndPush
  repository: $(imageRepository)
  dockerfile: '$(Build.SourcesDirectory)/Dockerfile'
  containerRegistry: $(dockerRegistryServiceConnection)
  tags: |
    $(tag)
```

Annexe D : Bibliographie et sitographie

- **Ouvrages de référence :**
 - “Deep Learning with Python” par François Chollet
 - “Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow” par Aurélien Géron
- **Documentation officielle :**
 - Documentation de TensorFlow : <https://www.tensorflow.org>
 - Documentation de Flutter : <https://flutter.dev>
 - Documentation de Microsoft Azure : <https://docs.microsoft.com/azure>
- **Articles et publications :**
 - “Aflatoxin contamination in maize : a review” - Journal of Food Science