Mise en situation 3 (E4)

Réaliser une application intégrant un service d'intelligence artificielle

Projet AniMOV : Surveillance et Analyse Comportementale des Chèvres avec l'IA

Formation Développeur en Intelligence Artificielle
RNCP 37827
Promotion 2023-2024

Manuel CALDEIRA

- Table des matières

1. INTRODUCTION

Dans le cadre de ma formation de Développeur en Intelligence Artificielle, j'ai eu l'opportunité de participer à un projet innovant intitulé AniMOV : Surveillance et Analyse Comportementale des Chèvres avec l'IA. Ce projet s'inscrit dans un contexte où l'intelligence artificielle joue un rôle de plus en plus prépondérant dans l'amélioration des pratiques agricoles, en particulier dans le suivi et la gestion du bien-être animal.

L'objectif principal du projet AniMOV est de développer une application capable de surveiller en temps réel le comportement des chèvres à l'aide de caméras de surveillance et de modèles de deep learning. En utilisant des réseaux de neurones convolutifs (CNN), l'application analyse les vidéos pour détecter des comportements spécifiques, identifier des anomalies, et fournir aux éleveurs des informations exploitables pour optimiser la gestion de leur troupeau.

Ce rapport détaille les différentes étapes du développement de l'application, depuis l'analyse des besoins jusqu'à la conception et la mise en œuvre technique. Il met en lumière les méthodes et outils employés pour créer une solution qui non seulement répond aux attentes des éleveurs et des scientifiques, mais qui a également le potentiel d'être adaptée à d'autres contextes, comme la surveillance d'animaux en zoo.

2. ANALYSE DES BESOINS

2.1. PARTIES PRENANTES ET DES OBJECTIFS

Les parties prenantes du projet AniMOV, incluent plusieurs groupes principaux impliqués dans le développement, la mise en œuvre et l'utilisation de l'application de surveillance comportementale des chèvres.

1. **Éleveurs** : Les utilisateurs finaux de l'application, qui bénéficieront directement de la surveillance et des analyses comportementales pour améliorer le bien-être des chèvres et optimiser les pratiques d'élevage.
2. **Développeurs et Ingénieurs Logiciels** : Ceux qui conçoivent et développent l'application, y compris les experts en deep learning et en analyse de données.
3. **Chercheurs et Scientifiques** : Les spécialistes en comportement animal et en bien-être animal et en IA qui utilisent les données pour effectuer des recherches et améliorer les algorithmes de détection et d'analyse comportementale.
4. **Fournisseurs de Technologie** : Les entreprises et les fournisseurs qui fournissent les équipements nécessaires, tels que les caméras de surveillance, les serveurs, et les autres matériels technologiques.

Ces parties prenantes collaborent pour assurer le succès du projet, de la conception initiale à la mise en œuvre et à l'utilisation pratique de l'application.

2.1.1. Objectifs :

Les objectifs du projet AniMOV, sont les suivants :

1. **Surveillance Précise du Comportement des Chèvres :**
   - Utiliser des caméras de surveillance pour collecter des enregistrements vidéo continus des chèvres dans leur environnement.
   - Analyser ces vidéos à l'aide de modèles de deep learning, notamment des réseaux de neurones convolutifs (CNN), pour reconnaître divers comportements animaux.
2. **Détection et Suivi des Cycles d'Activité Quotidienne :**
   - Identifier et suivre les cycles d'activité des chèvres tout au long de la journée.
3. **Identification des Comportements Anormaux :**
   - Détecter les comportements qui dévient de la norme pour intervenir rapidement.
4. **Reconnaissance des Signes de Stress ou de Maladie :**
   - Repérer les signes précoces de stress ou de maladie pour permettre une intervention précoce et ainsi améliorer le bien-être animal.
5. **Amélioration des Pratiques d'Élevage :**
   - Fournir aux éleveurs des insights précis et exploitables sur le comportement des chèvres, facilitant une gestion optimisée de l'élevage.
6. **Développement d'une Application Intuitive :**
   - Créer une application qui transforme les données complexes extraites par les algorithmes de deep learning en visualisations intuitives et facilement compréhensibles pour les éleveurs.
7. **Utilisation de la Technologie de l'Intelligence Artificielle :**
   - Exploiter les avancées en intelligence artificielle et en deep learning pour transformer les données brutes en informations utiles et exploitables en temps réel.
8. **Possibilité d'Adaptation à d'Autres Animaux :**
   - Explorer le potentiel d'adaptation de cette technologie de surveillance comportementale à d'autres animaux, comme ceux en zoo, pour une application plus large.

2.2. COLLECTE ET ANALYSE DES BESOINS

Les méthodes de collecte et d'analyse des besoins pour le projet AniMOV, telles qu'elles peuvent être déduites du document, comprennent plusieurs approches pour assurer que l'application développée répond adéquatement aux attentes et exigences des différentes parties prenantes. Voici les principales méthodes :

2.2.1. Collecte des Besoins

L'étude du corpus documentaire mis à disposition pour le projet, comprenant les contenus de réunions, les éléments de communication tels que les articles de journaux et les plaquettes, ainsi que les vidéos, a pour objectif de cerner en profondeur les défis quotidiens auxquels les éleveurs sont confrontés. Il s'agit notamment de leurs besoins spécifiques en matière de surveillance et de gestion des chèvres. Parallèlement, cette étude vise à comprendre les attentes des scientifiques pour mieux aligner les actions du projet avec leurs exigences. Enfin, l'objectif ultime est de structurer ces connaissances de manière à séduire les investisseurs, en leur présentant un projet à fort potentiel et bien ancré dans les réalités du terrain.

2.2.2. Analyse des Pratiques Existantes :

L'analyse des pratiques existantes vise à examiner en détail les méthodes actuelles de surveillance et de gestion utilisées par les éleveurs. Cette analyse se concentrera sur l'évaluation des outils et technologies déjà en place, ainsi que sur les procédures suivies par les éleveurs et les éthologues pour la gestion des chèvres. L'objectif est de comprendre l'efficacité des pratiques actuelles et d'identifier les domaines où des améliorations ou des innovations pourraient être introduites.

2.2.3. Prototypage :

Le prototypage a pour objectif de valider les besoins et les fonctionnalités envisagées en les concrétisant à travers un outil pratique. Pour ce faire, des prototypes de l'application seront développés, permettant ainsi d'identifier les éventuelles lacunes et d'ajuster les fonctionnalités en conséquence. Cette démarche itérative vise à s'assurer que l'application répond de manière optimale aux attentes des utilisateurs finaux tout en étant fonctionnelle et efficace.

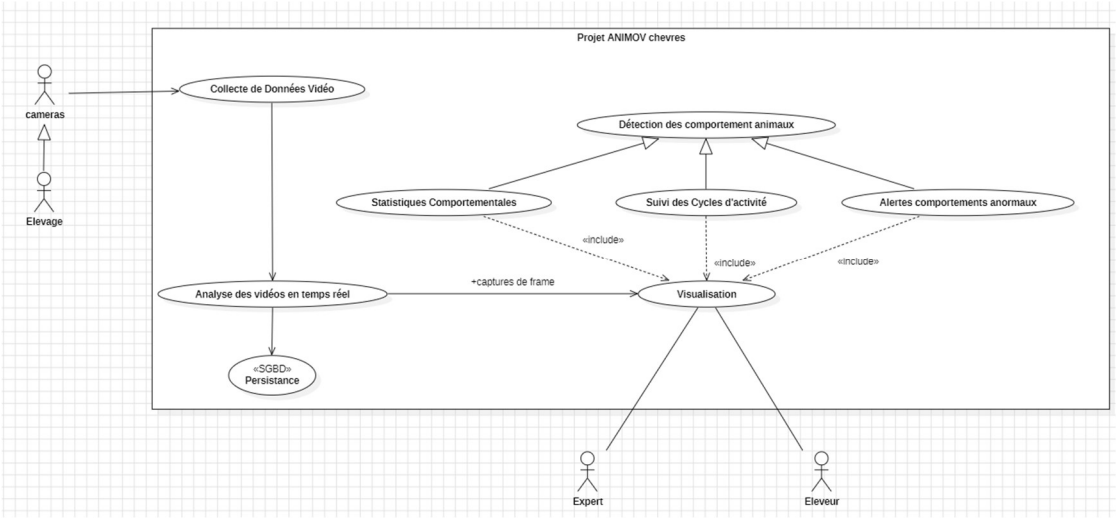
2.2.4. Analyse des Données Collectées :

L'analyse des données collectées a pour objectif d'examiner les données comportementales des chèvres afin d'identifier les tendances et les anomalies éventuelles. Pour ce faire, des modèles de deep learning ainsi que des techniques avancées d'analyse de données seront utilisés pour traiter les enregistrements vidéo. Ces approches permettront d'extraire des insights pertinents sur le comportement des chèvres, contribuant ainsi à une meilleure compréhension et à une gestion plus efficace des troupeaux.

En combinant ces méthodes de collecte et d'analyse des besoins, le projet AniMOV vise à développer une application qui répond précisément aux attentes des éleveurs et des scientifiques, tout en intégrant les avancées technologiques pour une surveillance efficace et une gestion optimale des chèvres.

2.3. SPECIFICATIONS FONCTIONNELLES

Les spécificités fonctionnelles du projet AniMOV, incluent diverses fonctionnalités essentielles conçues pour répondre aux besoins et optimiser la gestion des élevages. Voici les principales spécificités fonctionnelles :



2.3.1. Collecte de Données Vidéo :

La collecte de données vidéo consiste à utiliser des caméras de surveillance pour enregistrer et suivre en continu les chèvres dans leur environnement. Cette surveillance visuelle est ensuite intégrée à des modules de deep learning, permettant de traiter et d'analyser les vidéos en temps réel. Grâce à cette intégration, il est possible de suivre la position des animaux et de surveiller leur comportement de manière précise et instantanée.

2.3.2. Détection des Comportements Anormaux :

La détection des comportements anormaux repose sur l'utilisation d'algorithmes de deep learning capables d'identifier des comportements inhabituels chez les chèvres, potentiellement indicatifs de stress ou de maladies. Ce système permet une identification et une alerte en temps réel des comportements anormaux, facilitant ainsi une intervention rapide. Un système d'alerte est mis en place pour notifier les éleveurs en cas de détection de tels comportements, avec des notifications en temps réel pour permettre une réaction proactive.

Parallèlement, un suivi continu des cycles d'activité des chèvres est assuré, incluant les périodes de repos et d'activité. L'analyse des données comportementales recueillies permet de fournir des insights détaillés sur les habitudes quotidiennes des chèvres.

Enfin, des statistiques comportementales sont calculées, incluant les moyennes, les maximums, les minimums, et les écarts-types des comportements observés. Ces données sont agrégées sous forme de pourcentages et de suites temporelles, offrant ainsi une vision globale et détaillée du comportement des animaux au fil du temps.

2.3.3. Interface Utilisateur Intuitive :

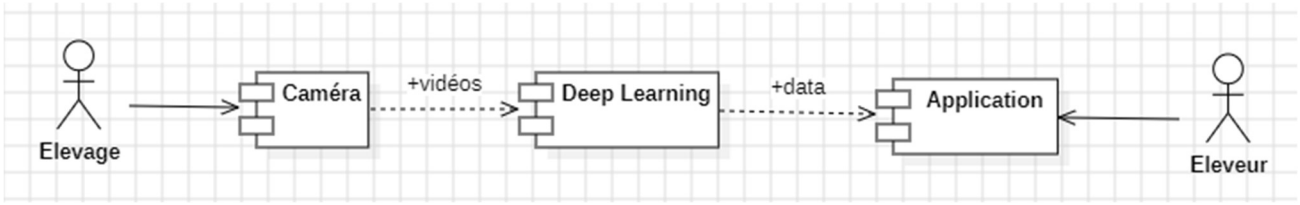
L'interface utilisateur intuitive est conçue pour offrir aux éleveurs une visualisation simple et compréhensible des données collectées. Cette interface graphique permet une navigation aisée à travers les différentes sections de l'application, notamment la surveillance en direct et l'analyse rétrospective. Parmi les fonctionnalités clés, une option de vidéo-surveillance en direct est

disponible, permettant un monitoring en temps réel des chèvres. De plus, l'application propose une analyse des enregistrements vidéo passés, facilitant l'identification des tendances et des anomalies comportementales survenues au fil du temps.

2.3.4. Gestion des Données

La gestion des données comprend le stockage des données collectées dans une base de données dédiée, où sont centralisés les enregistrements vidéo, les statistiques comportementales, ainsi que les métadonnées associées. Ces données sont structurées et accessibles via des flux de données en format JSON, assurant une interopérabilité et une facilité d'intégration avec d'autres systèmes ou outils d'analyse. Ces spécificités fonctionnelles permettent de créer une application robuste et intuitive, capable de transformer les données brutes collectées en informations précieuses pour les éleveurs, contribuant ainsi à améliorer le bien-être animal et à optimiser les pratiques d'élevage.

2.4. MODELISATION



Pour atteindre ces objectifs, on utilise des caméras de surveillance pour collecter des enregistrements vidéo en continus des chèvres dans leur environnement. Les données visuelles collectées sont ensuite traitées et analysées en utilisant des modèles de deep learning, notamment des réseaux de neurones convolutifs (CNN), qui sont formés pour reconnaître divers comportements animaux. Ces modèles sont entraînés sur un vaste ensemble de données annotées. Une attention particulière est accordée à la collecte de données diversifiées pour assurer la robustesse et la généralisation des modèles d'apprentissage profond. Une fois formés, les algorithmes sont capables de surveiller et d'analyser en continu le comportement des chèvres, fournissant aux éleveurs des informations précieuses pour la prise de décision en temps réel.

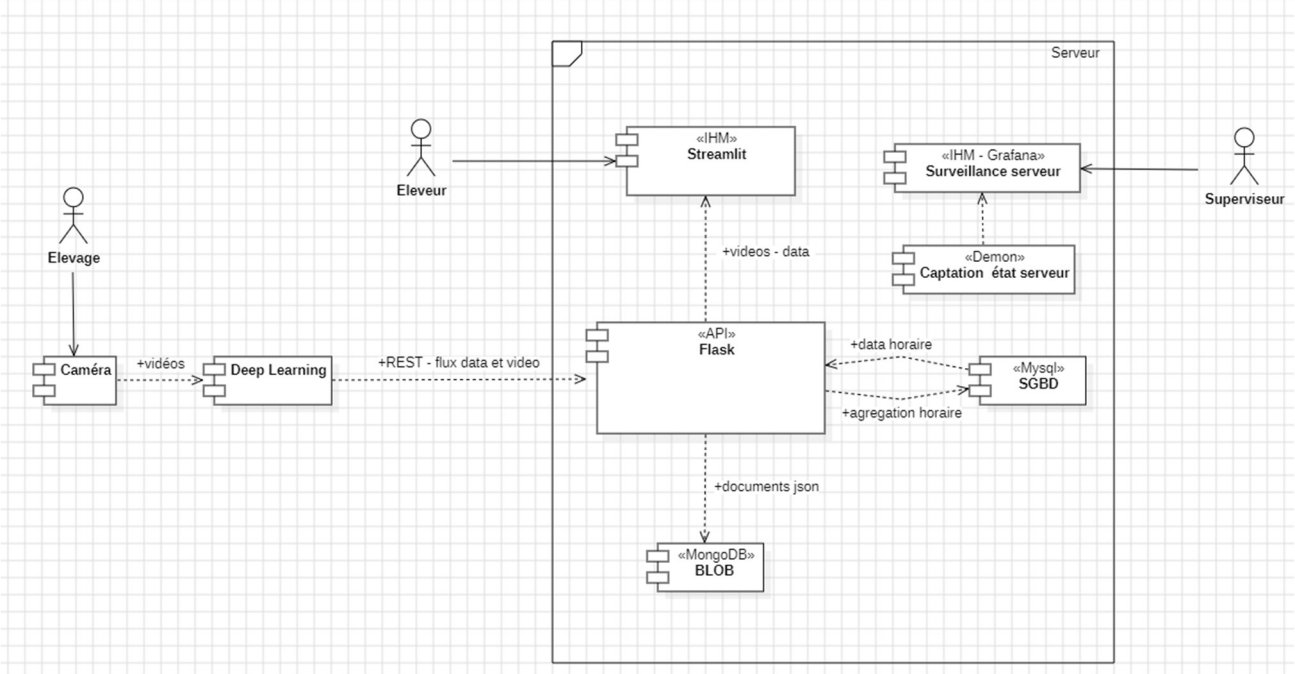
2.4.1. Notre apport au projet :

Notre travail se concentre sur le développement d'une application spécialisée qui recueille des données issues de l'analyse de deep learning. Cette application a pour but de transformer les caractéristiques complexes extraites par les algorithmes de deep learning en graphiques décisionnels intuitifs. Ces visualisations permettent aux éleveurs de comprendre facilement l'état de bien-être de leurs chèvres, d'identifier rapidement les comportements anormaux et de prendre des décisions éclairées pour la gestion de l'élevage. Notre contribution majeure réside dans la simplification de l'accès aux insights avancés que le deep learning procure, rendant ainsi la technologie IA directement applicable et bénéfique au quotidien des éleveurs.

3. CONCEPTION DU CADRE TECHNIQUE

3.1. 1. ARCHITECTURE TECHNIQUE ET APPLICATIVE

3.1.1. Architecture :



L'architecture du système est structurée selon un modèle "trois tiers", qui répartit les responsabilités entre la présentation, la logique métier, et le stockage des données, assurant ainsi une meilleure modularité, scalabilité, et maintenance.

Le **tiers de présentation** est consacré à l'interface utilisateur et à la visualisation des données. Il inclut deux composants principaux : **Streamlit** et **Grafana**. Streamlit est une application web interactive permettant aux éleveurs de visualiser les données traitées, de recevoir des notifications, et de consulter les résultats des analyses de Deep Learning. Grafana, quant à lui, est utilisé pour surveiller les performances du système en temps réel à travers des tableaux de bord personnalisés, offrant une vue d'ensemble des données de performance et de l'état du serveur.

Le **tiers logique**, ou backend, est responsable du traitement des données, de l'application des règles métiers, et de la gestion des communications entre le frontend et le stockage des données. Ce tiers comprend **Flask**, qui agit comme un serveur d'API recevant les requêtes HTTP du frontend, traitant les données, et communiquant avec les bases de données. Flask joue également un rôle crucial dans le traitement des données en mode edge computing, ce qui permet de réduire la latence et les coûts de transmission. En parallèle, un module de Deep Learning, fourni par l'INRA, reçoit les flux vidéo des caméras, effectue des inférences à l'aide de modèles d'apprentissage profond, et transmet les résultats à Flask pour un traitement ultérieur.

Le **tiers de données** se charge du stockage et de la gestion des données. Il repose sur deux bases de données principales : **MySQL** et **MongoDB**. MySQL est une base de données relationnelle utilisée pour stocker les données structurées,

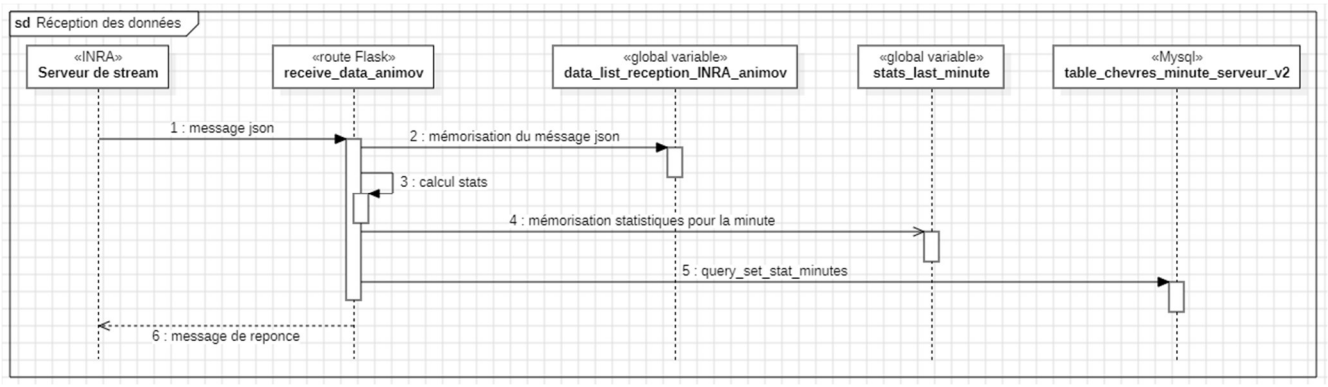
notamment celles agrégées sur une base horaire. MongoDB, une base de données NoSQL orientée documents, est utilisée pour stocker les fichiers JSON et potentiellement d'autres types de données BLOB. De plus, un démon de captation surveille en arrière-plan l'état du serveur et transmet ces informations à Grafana pour la surveillance en temps réel.

En termes de fonctionnement, l'utilisateur (l'éleveur) interagit avec Streamlit pour visualiser les données vidéo et les résultats des analyses. Le superviseur utilise Grafana pour surveiller les performances du système. Flask, au cœur du tiers logique, reçoit les requêtes du frontend, traite les données, calcule les statistiques, capte le flux de stream issu des inférences Deep Learning, et envoie les réponses appropriées. MySQL et MongoDB assurent le stockage respectif des données structurées et non structurées, tandis que le démon de captation veille à la surveillance continue du serveur.

Cette architecture "**trois tiers**" organise les responsabilités de manière distincte et efficace, facilitant ainsi l'évolution, l'entretien, et la mise à l'échelle du système tout en assurant une gestion optimale des données et des performances.

3.1.2. Concepts dynamiques

- **Réception des données :**

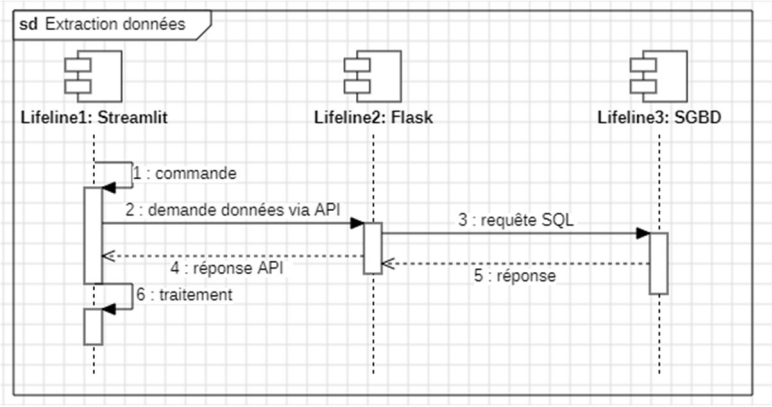


Le diagramme de séquence UML décrit l'interaction entre différents composants lors de la réception et du traitement des données, en illustrant le flux d'informations entre les acteurs et les lignes de vie suivante : le Serveur de stream (INRA), la route Flask `receive_data_animov`, les variables globales `data_list_reception_INRA_animov` et `stats_last_minute`, ainsi que la base de données MySQL `table_chevres_minute_serveur_v2`.

Le processus commence par l'envoi d'un message JSON par le Serveur de stream (INRA) à la route Flask `receive_data_animov`. Cette route mémorise ensuite le message reçu dans la variable globale `data_list_reception_INRA_animov`. Sur la base des données reçues, la route Flask effectue le calcul des statistiques via un processus d'Edge computing, et les résultats sont mémorisés dans la variable globale `stats_last_minute`. Par la suite, Flask envoie une requête pour mettre à jour ces statistiques dans la base de données MySQL, plus précisément dans la table `table_chevres_minute_serveur_v2`. Pour conclure, Flask répond au Serveur de stream en envoyant un message indiquant soit un traitement correct avec un code 200, soit un code d’erreur en cas de problème.

Ce diagramme met en évidence le flux d'informations critique entre les différents composants, démontrant comment les données JSON sont reçues, traitées, et stockées de manière efficace, assurant ainsi une gestion fiable des données tout au long du processus.

- **Extraction de données agrégées :**



Ce diagramme séquentiel illustre le processus d'extraction des données en impliquant trois composants principaux : Streamlit, Flask, et un SGBD (Système de Gestion de Base de Données).

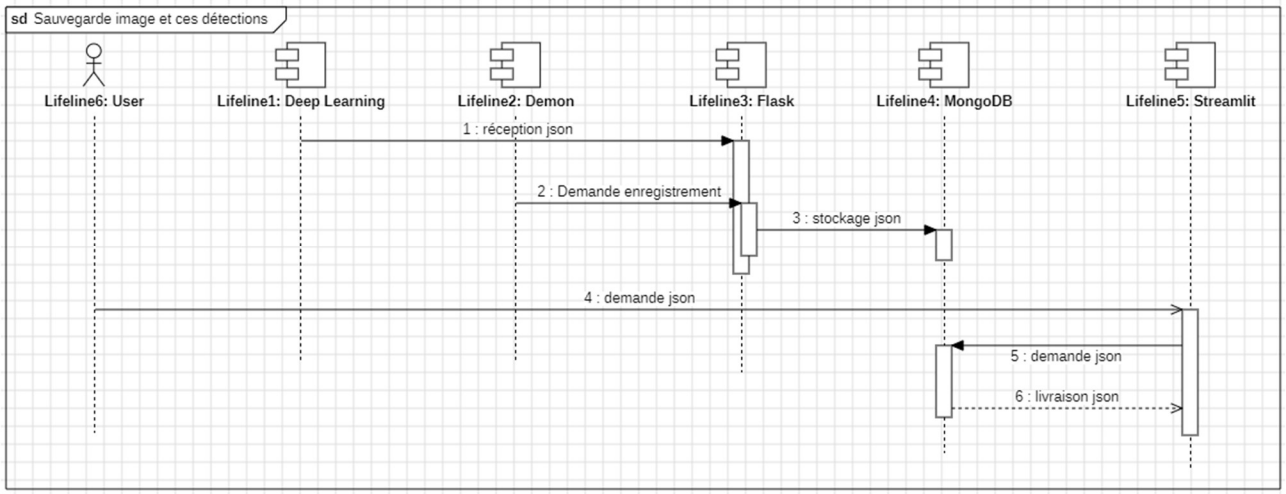
Le processus débute lorsque l'utilisateur interagit avec l'application Streamlit, envoyant une commande via une interface homme-machine (IHM). En réponse, Streamlit envoie une demande de données à l'API Flask, sous la forme d'une requête HTTP, qu'elle soit de type GET ou POST. Après réception de cette demande, Flask génère une requête SQL afin d'interroger la base de données et de récupérer les informations demandées. Le SGBD traite cette requête SQL et renvoie les données correspondantes à Flask. Ensuite, Flask formate ces données en JSON et les renvoie à Streamlit via l'API. Enfin, Streamlit reçoit les données, les traite, et les affiche à l'utilisateur.

Ce diagramme met en évidence le flux de données et les interactions entre les composants pour réaliser une tâche d'extraction de données de manière efficace.

- **Processus de stockage BLOB des images et détections Deep Learning :**

Ce diagramme de séquence décrit l'interaction entre différents composants d'un système, depuis la réception de données JSON par un modèle de Deep Learning jusqu'à leur stockage dans une base de données MongoDB via une application Flask, ainsi que la récupération de ces données par un utilisateur via une interface Streamlit.

Dans un premier temps, le modèle de Deep Learning envoie régulièrement des données d'inférence, incluant des détections et des images, sous forme de stream JSON. Un démon, fonctionnant en arrière-plan, envoie toutes les 15 minutes une demande d’enregistrement à l'API Flask. Flask reçoit cette demande, traite les données reçues du stream, puis les envoie à MongoDB pour stockage.



Par la suite, un utilisateur, via l'application Streamlit, peut initier une commande pour demander ces données JSON stockées dans MongoDB. Streamlit envoie alors une requête à MongoDB, qui traite la demande et renvoie les données JSON nécessaires à Streamlit, permettant ainsi à l'utilisateur d'accéder aux informations demandées. Ce processus global assure une gestion fluide des données, depuis leur génération et stockage jusqu'à leur récupération et utilisation par l'utilisateur final.

3.2. SPECIFICATION DES TECHNOLOGIES ET OUTILS

3.2.1. Langages et Framework Utilisés

Les choix technologiques et les outils pour ce projet sont soigneusement sélectionnés pour garantir une efficacité et une facilité d'utilisation maximales. Le tiers de présentation (Frontend) est assuré par **Streamlit**, un outil interactif développé en Python, et **Grafana**, qui permet de concevoir des tableaux de bord en temps réel. Le tiers logique (Médiation/Backend) repose sur **Flask**, également en Python, pour développer des applications web performantes et exposer des API RESTful, facilitant ainsi une communication fluide entre le frontend et le backend.

Pour le tiers de données (Stockage), **MySQL** est utilisé pour les données structurées avec SQL comme langage, tandis que **MongoDB** prend en charge les données non structurées, utilisant JavaScript pour les requêtes. Un démon en Python surveille l'état des serveurs, assurant ainsi la captation et la gestion des données en temps réel.

Edge Computing est adopté pour traiter les données localement, réduisant ainsi la latence et les coûts de transmission. Le format **JSON** est utilisé pour l'échange de données, garantissant une lecture et une écriture aisées pour les humains et les machines.

En termes d'outils de développement, **Visual Studio Code** est utilisé pour l'édition de code, et **Spyder** sert d'environnement de développement intégré (IDE) pour les scientifiques et les analystes de données. **Git** et **Fork** sont les outils de choix pour le contrôle de version, tandis que **DBeaver** et **HeidiSQL** sont utilisés pour la gestion des bases de données. **MongoDB Compass** permet une interaction intuitive avec MongoDB, et **Docker** facilite la

conteneurisation, permettant un déploiement et une portabilité simplifiés des applications.

D'autres outils comme **Bitvise SSH Client** assurent des connexions sécurisées à des serveurs distants, et **Cloud Scaleway** est le fournisseur de services cloud choisi pour l'hébergement. **Windows Subsystem for Linux (WSL)** permet aux développeurs d'utiliser les outils Linux dans un environnement Windows. Ces choix technologiques et outils sont conçus pour créer un environnement de développement robuste, sécurisé et hautement performant, répondant aux besoins spécifiques du projet.

3.3. METHODES DE DEVELOPPEMENT

3.3.1. Méthodologie

Nous avons adopté une méthodologie type Scrum à base de sprints de 3 semaines. Cette approche permet une planification itérative et incrémentale, facilitant ainsi l'adaptation aux changements et à l'amélioration continue.

3.3.2. Planification

Pour la planification, nous utilisons Excel pour découper le projet en tâches et sous-tâches, organisées dans un diagramme de Gantt. Cela offre une vue d'ensemble claire des différentes phases du projet et des dépendances entre les tâches.

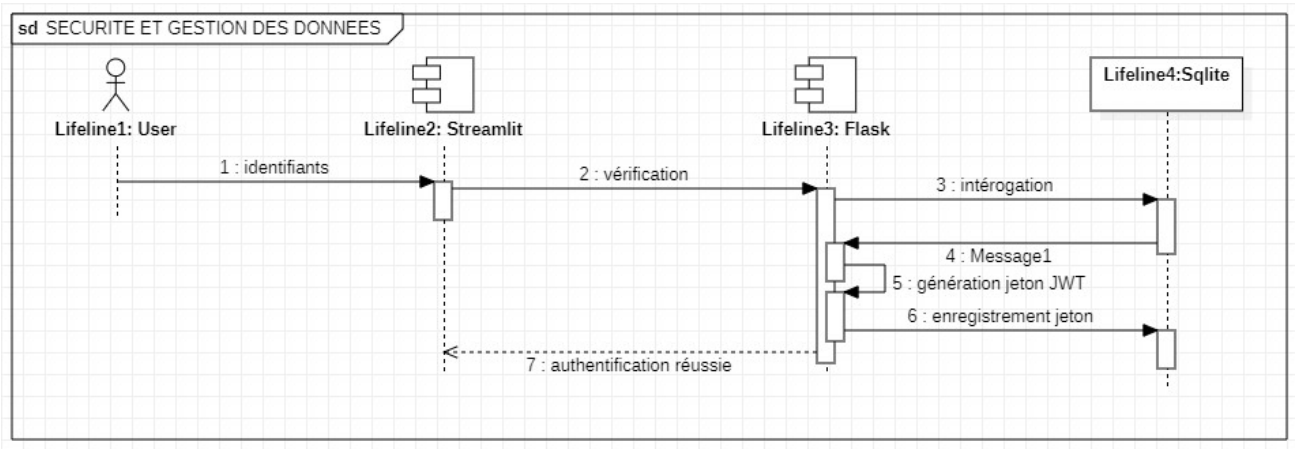
3.3.3. Réalité du Projet

Dans la pratique, j'ai davantage consacré mon temps à un projets de prospection en intelligence artificielle (IA) plutôt qu'à ce projet. Cela s'est fait sans suivre un planning strict, ce qui a nécessité une certaine flexibilité et une adaptation constante aux priorités changeantes. Cette approche a permis de concentrer les efforts sur les aspects innovants et exploratoires du projet IA, tout en répondant aux besoins et aux opportunités qui se présentaient pour ce projet.

3.4. SECURITE ET GESTION DES DONNEES

Dans ce projet, la sécurité et la gestion des données sont rigoureusement assurées par l'utilisation de JSON Web Tokens (JWT) pour l'authentification et la sécurisation des accès aux fonctionnalités de l'API. Lorsqu'un utilisateur souhaite interagir avec l'API, il doit d'abord s'authentifier en envoyant ses identifiants via un point d'entrée dédié. Si l'authentification est réussie, l'API génère un jeton JWT qui est ensuite utilisé pour toutes les requêtes ultérieures, garantissant ainsi que chaque requête provient d'un utilisateur autorisé. Ce jeton est inclus dans les en-têtes de requêtes et permet de sécuriser l'accès aux ressources. En cas de problème, comme une requête mal formatée ou l'utilisation d'un jeton invalide, l'API renvoie des messages d'erreur explicites, avec des codes d'erreur appropriés, tels que `401 Unauthorized`, ce qui informe l'utilisateur de la nécessité de se réauthentifier. Ces mesures garantissent que seuls les utilisateurs autorisés peuvent accéder aux informations et fonctionnalités critiques de l'application.

3.4.1. Description du diagramme de séquence



1. **Utilisateur :**
   - L'utilisateur initie le processus en envoyant ses informations d'identification (nom d'utilisateur et mot de passe) via l'interface utilisateur (IHM).
2. **Interface utilisateur (IHM) :**
   - L'IHM reçoit les informations et les transmet à l'API pour authentification.
3. **API (Flask) :**
   - L'API reçoit les informations d'authentification et interagit avec la base de données SQLite pour vérifier les informations fournies.
4. **Base de données (SQLite) :**
   - La base de données stocke les informations d'identification (login et mot de passe) ainsi que les jetons JWT. L'API interroge SQLite pour valider les informations d'identification.
5. **JWT Service :**
   - Si les informations d'identification sont correctes, le service JWT génère un jeton JWT et le retourne à l'API.
6. **API (Flask) :**
   - L'API enregistre ce jeton dans la base de données SQLite et le retourne à l'utilisateur via l'IHM.
7. **Interface utilisateur (IHM):**
   - Une fois l'authentification réussie, l'utilisateur peut envoyer des requêtes pour accéder aux routes et obtenir ainsi les données.

4. COORDINATION DE LA REALISATION TECHNIQUE

4.1. PLANIFICATION ET GESTION DE PROJET

Le projet est organisé en plusieurs sprints, chacun ayant des objectifs spécifiques pour assurer une progression méthodique et structurée du développement.

Sprints 1 : Mise en place de l'environnement de développement (Octobre)

Le premier sprint est centré sur l'établissement de l'environnement de développement. Cela inclut la configuration des outils de développement, l'installation de WSL (Windows Subsystem for Linux), la création d'environnements virtuels via Anaconda, et la mise en place de serveurs locaux pour Flask et Streamlit. Docker est également configuré pour gérer les bases de données MySQL et MongoDB. En parallèle, une étude des données .csv est menée, comprenant le chargement des données dans la base de données via un script Python, une analyse descriptive des données, et la rédaction d'un rapport descriptif. Ce sprint inclut également la conception de tables agrégées à partir des données .csv, la création d'une API Flask pour interagir avec la base de données, et la conception d'une application Streamlit pour visualiser les données agrégées.

Sprints 2 : Mise en place du serveur cloud (Novembre)

Le deuxième sprint se concentre sur la configuration du serveur cloud. Cela commence par la mise en place d'un environnement virtuel et l'implémentation de Flask. Ensuite, Docker est déployé avec Docker Compose pour gérer les bases de données MySQL et MongoDB ainsi que Grafana. L'installation de Bitvise SSH est effectuée pour des connexions sécurisées. Ce sprint inclut également le développement d'un démon de surveillance du serveur, la création de tableaux de bord dans Grafana, et le développement d'une route Flask pour la gestion des incidents et la résolution (arrêt et démarrage automatique de la base de données). De plus, un système de notification par email est mis en place via Flask pour les événements notables, et la journalisation des événements est intégrée. Enfin, tout est déployé sur le cloud.

Sprints 3 : Détection d'éléphants (Décembre)

Le troisième sprint porte sur l'extraction et l'analyse d'images à partir de vidéos bruitées. Une structure de fichiers est mise en place pour effectuer des expériences de deep learning, et un modèle YoloV8 est entraîné sur les images en utilisant une base de données COCO existante.

Sprints 4 : Extraction du fond des images et construction d'une base COCO (Janvier)

Ce sprint est dédié à la production d'un pipeline pour l'extraction du fond des images et à la création d'une nouvelle base de données COCO à partir des images traitées. Cela permet de raffiner les données pour des expériences de deep learning plus précises.

Sprints 5 : Implémentation dans Flask de la réception du flux en stream (Février)

Le cinquième sprint se concentre sur l'intégration du flux en stream dans Flask, notamment par la conception d'une route spécifique pour la réception du flux du modèle de détection de chèvres. Il inclut également la persistance des données dans la SGBD, la conception de requêtes pour alimenter Flask, la création de routes sécurisées dans Flask (avec mot de passe et jeton), et le développement d'une application Streamlit pour le monitoring en temps réel.

Sprints 6 : Détection d’éléphants dans des vidéos de bonne qualité (Mars)

Le sixième sprint implique une veille technologique sur la détection des postures animales, l'extraction d'images à partir de vidéos de haute qualité, l'annotation de ces images à l'aide du logiciel labelImg, la création d'une base COCO, et l'entraînement de trois modèles YoloV8.

Sprints 7 : Détection couché/debout (Avril)

Ce sprint se concentre sur l'installation de Postgres et PostGIS, l'extraction des interférences de tous les frames de deux vidéos et le stockage des géométries des détections dans des tables PostGIS. Des requêtes géométriques sont conçues pour calculer le taux de recouvrement des détections. Enfin, une application Streamlit est développée pour comparer et analyser les modèles sur le critère debout/couché, suivie de la rédaction d'un rapport d'analyse des modèles.

Sprints 8 : Suivi de la boîte de détection des éléphants pour déterminer s'ils dorment (Mai)

Ce sprint porte sur la conception de requêtes géométriques pour l'appariement des boîtes de détection entre les frames. L'application Streamlit est complétée par un formulaire d'analyse de suivi de boîtes de détection. De plus, un serveur MLFlow est installé sur le cloud, et les modèles sont alimentés via un script Python. Enfin, une route est implémentée dans Flask pour charger le modèle sur le cloud.

Sprints 9 : Analyse et Veille Technologique : Évaluation des Puces IA Embarquées et Étude Comparative d'un Chatbot Concurrent (Juin)

Lors du sprint 9, nous avons mené deux activités distinctes : une veille sur les puces électroniques dédiées à l'intelligence artificielle embarquée et une étude approfondie d'un chatbot concurrent. La veille technologique visait à identifier et analyser les avancées récentes en matière de puces électroniques optimisées pour l'IA, en se concentrant sur les performances, la consommation énergétique et les possibilités d'intégration dans divers dispositifs embarqués. Cette recherche a permis de compiler des informations précieuses sur les tendances actuelles et les perspectives futures de ce secteur.

Parallèlement, l'étude du chatbot concurrent s'est déroulée à travers une série d'échanges dynamiques entre deux instances de ChatGPT. Le premier chatbot, incarnant un persona défini en début d'interview, a répondu aux questions posées par le second chatbot, qui jouait le rôle d'évaluateur des risques. Ce dernier, en fonction d'un contexte évolutif, adaptait ses questions pour approfondir l'analyse du comportement et des réponses du chatbot étudié. Les réponses fournies par le premier chatbot étaient ensuite analysées par le second, qui formulait de nouvelles questions en conséquence, tout en produisant régulièrement des évaluations des risques. Cette approche itérative a permis de tester la robustesse et la pertinence des réponses du chatbot concurrent, ainsi que de mieux comprendre les éventuels risques associés à son utilisation dans divers contextes. Les résultats obtenus seront utilisés pour affiner notre propre solution et anticiper les défis potentiels.

Sprints 10 : Prospection CIR et Thèse CIFRE, Analyse Nutritionnelle des Poussins et Préparation du Document E3 (Juillet)

Lors du sprint 10, trois activités principales ont été menées : une prospection sur le Crédit Impôt Recherche (CIR) et les thèses CIFRE, une étude exploratoire des données (EDA) à partir d'un fichier CSV concernant l'élevage de poussins, ainsi que l'ébauche du document E3. La prospection sur le CIR et les thèses CIFRE a permis de recueillir des informations essentielles pour optimiser le financement des projets de recherche et développement au sein de l'entreprise, en identifiant les opportunités et les conditions requises pour bénéficier de ces dispositifs.

Simultanément, une étude approfondie des données nutritionnelles liées à l'élevage de poussins a été réalisée, basée sur un fichier CSV contenant les informations pertinentes. Cette analyse a conduit à la création d'une application Streamlit dédiée à l'analyse des données, permettant de visualiser et d'interpréter facilement les tendances nutritionnelles et leur impact sur la croissance des poussins. L'application a été conçue pour offrir une interface utilisateur intuitive, facilitant ainsi l'exploration des données pour des utilisateurs non techniques.

Enfin, une première version du document E3 a été élaborée.

Sprints 11 : Automatisation via GitHub Actions et Finalisation des Livrables E1 à E5 (Aout)

Lors du sprint 11, deux tâches cruciales ont été réalisées : la mise en place de GitHub Actions et la complétion des livrables E1 à E5. L'implémentation de GitHub Actions a été un point fort de ce sprint, permettant d'automatiser divers processus de développement, tels que les tests et le déploiement continu.

4.2. COLLABORATION ET COMMUNICATION

La collaboration et la communication au sein de l'équipe sont des éléments cruciaux pour garantir la fluidité du projet et la coordination des efforts. Pour atteindre ces objectifs, des réunions quotidiennes sous forme de stand-ups sont organisées avec mon tuteur. Ces réunions brèves et efficaces permettent de faire un point sur l'avancement des tâches, de partager les éventuels obstacles rencontrés, et de synchroniser les prochaines étapes. Ce format favorise la transparence et assure un alignement constant sur les objectifs quotidiens.

En complément, des outils de communication tels que Slack et Microsoft Teams jouent un rôle indispensable pour maintenir une communication instantanée et efficace au sein du consortium ANIMOV ou de l'équipe de l'entreprise. Ces plateformes permettent de partager en temps réel des informations cruciales, de résoudre rapidement les problèmes et de coordonner les tâches de manière fluide. Grâce à ces outils, les membres de l'équipe peuvent collaborer de façon harmonieuse et productive, même à distance, garantissant ainsi une continuité dans le travail.

De plus, ces plateformes facilitent l'organisation de réunions virtuelles et le suivi des progrès en temps réel, ce qui contribue à une prise de décision plus éclairée.

5. Pratiques MLOps

Le MLOps, ou Machine Learning Operations, combine les pratiques du DevOps avec la gestion des modèles de machine learning pour optimiser leur cycle de vie, de la conception à la production. En facilitant la collaboration entre les équipes de développement et opérationnelles, le MLOps accélère le déploiement des modèles tout en assurant leur fiabilité, reproductibilité et scalabilité.

Automatisation des workflows IA : Dans le cadre de notre modèle SARIMAX, l'utilisation de MLflow est essentielle pour suivre et gérer ce modèle. MLflow permet d'enregistrer et de suivre les versions du modèle SARIMAX, ses paramètres associés, ainsi que les métriques de performance telles que le RMSE. Il facilite également la gestion du cycle de vie du modèle, incluant son déploiement, ses mises à jour, et la récupération des versions les plus récentes pour des prédictions optimisées.

Surveillance et gestion du modèle SARIMAX : Les pratiques de déploiement continu et de monitoring des performances sont cruciales pour assurer la qualité du modèle SARIMAX en production. Le déploiement continu permet de mettre à jour ce modèle sans interruption, grâce à une intégration fluide avec des outils comme MLflow et Flask. Le suivi des performances du modèle SARIMAX est réalisé à travers des calculs de métriques, tels que le RMSE, qui permettent d'évaluer et d'ajuster le modèle en fonction des résultats obtenus. Des tests automatisés sont également mis en place pour valider le bon fonctionnement du modèle dans l'environnement de production.

6. Git Action

GitHub Actions est un outil d'intégration et de déploiement continu (CI/CD) intégré à GitHub. Il permet d'automatiser des workflows tels que la construction, les tests et le déploiement de votre code directement depuis votre dépôt GitHub.

Le workflow s'exécute sur une machine virtuelle Ubuntu (`ubuntu-latest`). Il intègre un service MySQL 5.7, nécessaire pour votre projet. Les variables d'environnement pour la configuration de MySQL, telles que le mot de passe root et le nom de la base de données, sont gérées via les secrets GitHub.

Les étapes du workflow débutent par le checkout du code à l'aide de l'action `actions/checkout@v4`. Ensuite, on configure l'environnement Python en spécifiant la version 3.8, ce qui correspond à la version de Python utilisée dans le projet. On met à jour `pip` et on installe les dépendances listées dans `requirements.txt`, assurant ainsi que l'environnement est correctement préparé pour l'exécution des tests.

Avant d'aller plus loin, on introduit une pause de 15 secondes pour permettre à MySQL de s'initialiser correctement. Ensuite, on exécute le script Python (`migrations/mysql_setup.py`) pour configurer la base de données, en utilisant une URI de connexion générée à partir des secrets GitHub.

Enfin, on lance les tests avec `pytest`, en limitant le nombre de tests échoués à 5 avant de s'arrêter. L'ensemble de ce workflow est conçu pour assurer que le projet, qui dépend de MySQL et utilise Python, est correctement testé à chaque modification.

```yaml
name: CI

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    services:
      mysql:
        image: mysql:5.7
        env:
          MYSQL_ROOT_PASSWORD: ${{ secrets.DB_PASSWORD }}
          MYSQL_DATABASE: ${{ secrets.DB_NAME }}
        ports:
          - 3306:3306
        options: >-
          --health-cmd="mysqladmin ping --silent"
          --health-interval=10s
          --health-timeout=5s
          --health-retries=3

    steps:
    - name: Check out code
      uses: actions/checkout@v4

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.8'

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: Wait for MySQL to be ready
      run: sleep 15  # Attendre 15 secondes pour que MySQL soit prêt

    - name: Run MySQL setup
      env:
        MYSQL_DATABASE_URI: mysql+pymysql://${{ secrets.DB_USER }}:${{ secrets.DB_PASSWORD }}@${{
secrets.DB_HOST }}:3306/${{ secrets.DB_NAME }}
      run: |
        python migrations/mysql_setup.py

    - name: Run tests with pytest
      run: |

        pytest -v --maxfail=5 --disable-warnings
```

7. CONCLUSION

Le développement de l'application s'est appuyé sur une méthodologie rigoureuse, intégrant des pratiques MLOps qui assurent la fiabilité et l'efficacité des modèles déployés. La structuration en architecture "trois tiers" garantit la modularité et la scalabilité du système, tandis que l'intégration de technologies comme Flask, Streamlit, et Grafana permet une gestion fluide des données et une expérience utilisateur optimisée.

En répondant aux besoins des éleveurs et des scientifiques tout en assurant une gestion sécurisée des données, le projet AniMOV se positionne comme un outil indispensable pour une gestion moderne et éclairée des élevages. Ce projet démontre également le potentiel de l'IA pour transformer les pratiques agricoles, ouvrant la voie à des applications similaires pour d'autres espèces animales et dans divers environnements. L'implication dans ce projet m'a permis d'approfondir mes compétences

en développement d'applications IA tout en contribuant à une cause d'importance sociale et économique, en ligne avec les objectifs de ma formation en Intelligence Artificielle.