

Spécification détaillées

**Détection d’anomalies pulmonaires**

**(COVID et Pneumonies Virales)**

**Plan**

1. Objectif
2. Acteurs
3. Cas d’usage
4. Présentation de l’interface utilisateur - Frontend
5. Présentation des APIs – Backend
6. Spécifications techniques – Fonctionnement Backend
7. Règles de gestion - Monitoring de l’application

# Objectif

L’objectif du client est de fournir un service de détection d’anomalie pulmonaire type COVID ou Pneumonie Virale à ses internes en médecine afin de réaliser un premier diagnostic permettant de trier les patients non malades et les patients présentant une anomalie pulmonaire.

# Acteurs

* **Les utilisateurs :**

Internes : Ce sont des praticiens en cours de formation spécialisée qui exercent sous la responsabilité d’un médecin senior. Ils effectuent un premier diagnostic sur l’état de santé d’un patient avant de faire appel à des médecins

* **Les contributeurs :**

Ce sont des utilisateurs de profil radiologue (médecin senior) qui valident le diagnostic fourni par l’application aux internes. Leur expertise permet d’améliorer les modèles de prédiction au début de la mise en place de l’outil

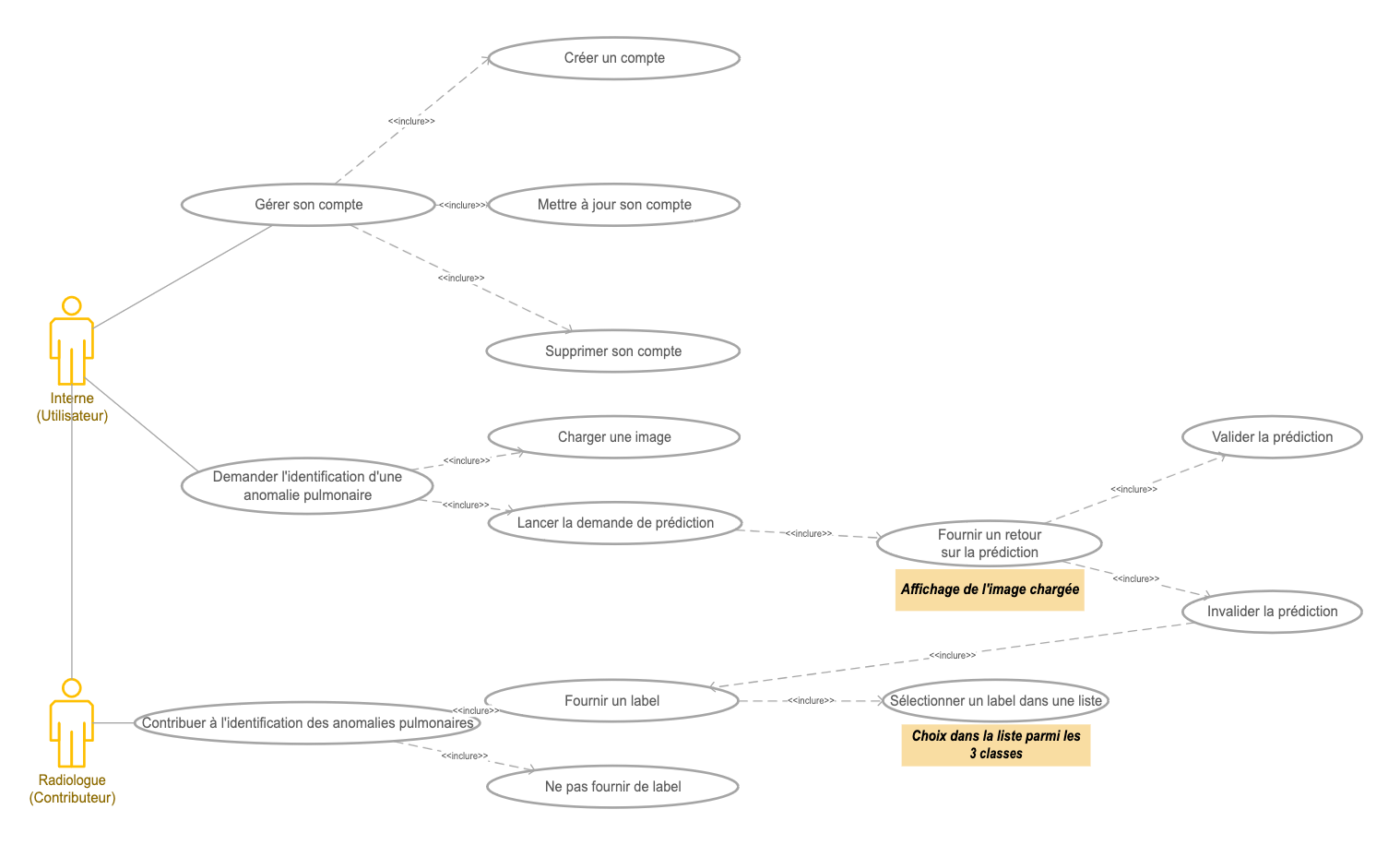
* **L’administrateur :**

Deux profils d’administrateur :

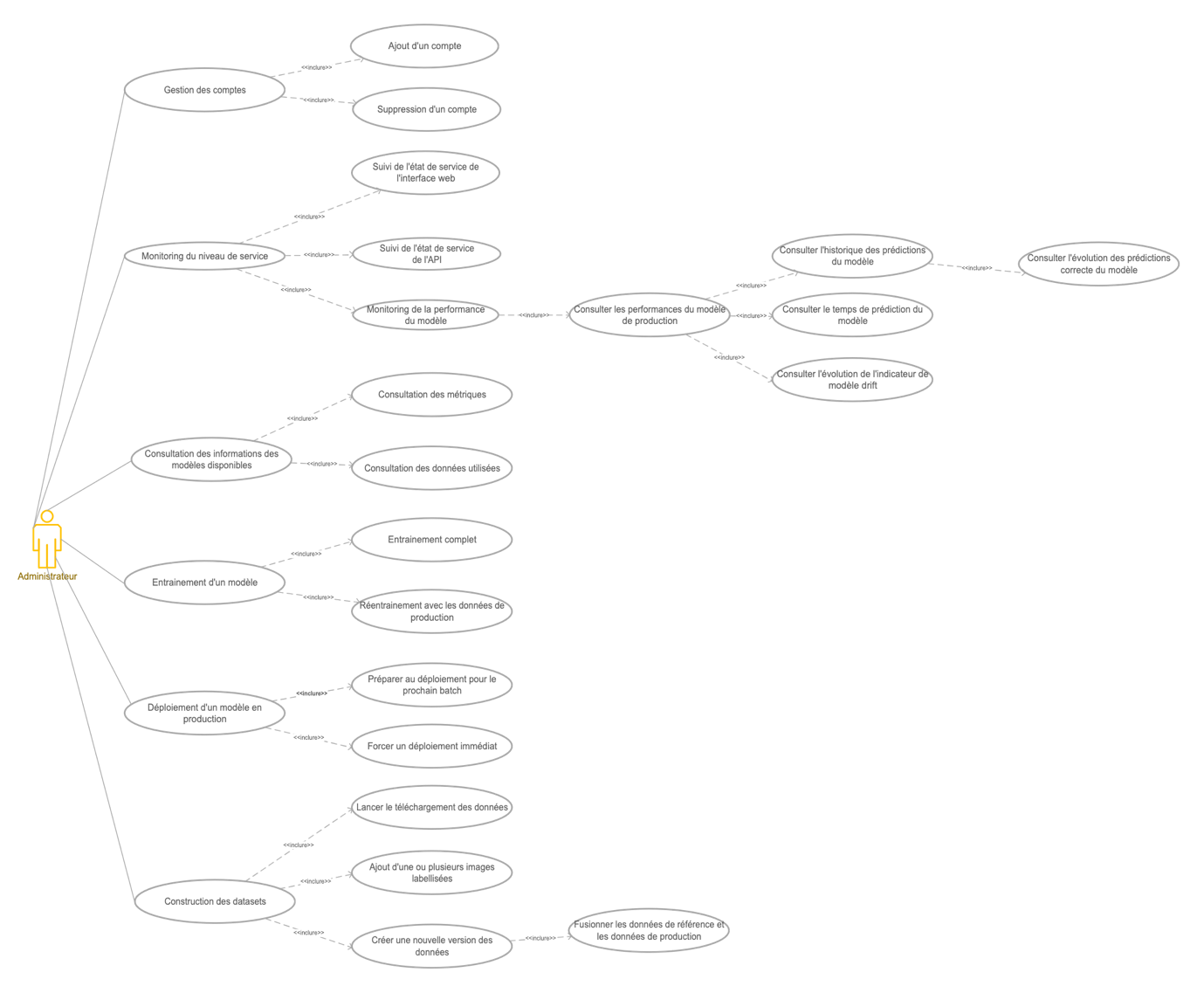
* **Datascientist :** Son rôle est de répondre aux besoins du client en matière de performance du modèle. Ainsi, il choisit les données les plus pertinentes, construit l’architecture IA adaptée et définit les métriques pour mesurer les performances après déploiement. Il effectue donc l’entrainement et l’évaluation d’un modèle au début du projet
* **MLOps Engineer :** Son rôle est de s’assurer du bon fonctionnement du service pour les utilisateurs en surveillant l’état de l’application mais aussi la performance du modèle. En cas de drift du modèle, il peut décider d’effectuer un ré-entrainement du modèle et de déployer le nouveau modèle si les performances sont conformes aux exigences du client.

# Les cas d’usage

**Vue générale des cas d’usage par acteur – Utilisateur/Contributeur**



**Vue générale des cas d’usage par acteur – Administrateur**



## Utilisateur

* + 1. **Gérer son compte**

Pour utiliser notre application, un utilisateur doit être authentifié et doit donc créer un compte, il doit aussi pouvoir le mettre à jour ou le supprimer.

* ***Créer un compte***

Pour la création d’un compte, un nom d’utilisateur et mot de passe fort seront demandés. *Pour des raisons de simplicité, il ne sera pas demandé de confirmer la création du compte par email.*

***Nom d’utilisateur : 12 caractères minimum***

***Mot de passe : 12 caractères minimum, Inclure des majuscules (A, B, C, ...) et des minuscules (a, b, c, ...). Utiliser des chiffres (0-9). Intégrer des symboles et caractères spéciaux (par exemple, @, #, $, %, &, \*).***

* ***Mettre à jour son compte***

L’utilisateur aura la possibilité de mettre à jour son mot de passe. Pour cela, il doit saisir son mot de passe actuel et son nouveau mot de passe à deux reprises.

***Nouveau Mot de passe : Différent du mot de passe actuel, 12 caractères minimum, Inclure des majuscules (A, B, C, ...) et des minuscules (a, b, c, ...). Utiliser des chiffres (0-9). Intégrer des symboles et caractères spéciaux (par exemple, @, #, $, %, &, \*).***

* ***Supprimer son compte***

L’utilisateur peut supprimer son compte à tout moment.

* + 1. **Demander l'identification d'une anomalie pulmonaire**
* ***Charger une image***

Afin d’identifier une anomalie pulmonaire, un utilisateur doit charger une image.

***Les formats acceptés : png, jpg***

***Taille maximale acceptée : 500 Kb***

***Nombre maximal d’images nécessaires pour une prédiction : 1***

L’utilisateur a la possibilité de supprimer l’image et d’en recharger une nouvelle avant de lancer une demande de prédiction.

* ***Lancer une demande de prédiction***

Lorsque l’utilisateur lance une demande de prédiction, celle-ci est réalisée en Backend et sont fournis à l’utilisateur la classe et la **photo** chargée par l’utilisateur, ainsi que l’**indice de confiance**.

* + Fournir un retour sur la prédiction :
    - Valider la prédiction : A l’aide de la classe et l’affichage de l’image de radiographie fournie
    - Invalider la prédiction et fournir une classe : Choix d’une classe dans la liste
    - Invalider la prédiction sans fournir de label : Lorsque l’utilisateur ne parvient pas à identifier l’anomalie

## Contributeur (Hors MVP)

* + 1. **Contribuer à l'identification des anomalies pulmonaires**
* ***Fournir un label***

Seule l’image non labellisée est affichée

Lorsque le contributeur peut établir un diagnostic, il peut *sélectionner un label dans une liste*

* ***Ne pas fournir de label***

Lorsque l’utilisateur ne parvient pas à identifier l’anomalie, il n’effectue aucune action.

***Règle de validation définitive d’un label : 1 validation par le contributeur***

## Administrateur

* + 1. **Gestion des comptes**
       1. **Ajout d’un compte**
       2. **Suppression d’un compte**
    2. **Monitoring du niveau de service**

L’Administrateur dont le rôle est de s’assurer du bon fonctionnement des services proposés, doit avoir à disposition des outils de monitoring de service.

* + - 1. Suivi de l'état de service de l'interface web et des APIs

Deux points de suivi de l’état de service :

* Via l’interface Web, dans un menu dédié au suivi de l’état de service à un instant t

**Techniquement :** Lors de l’accès à l’application, un Health Check est envoyé à chaque service, et un status est affiché (UP si le service est démarré - status http 200, DOWN si non)

* Via une alerte lorsqu’un service n’est pas démarré.

**Techniquement :** Un service de batch est exécuté toutes les minutes afin de vérifier l’état de service de toutes les applications. Si au moins un des services ne fonctionne pas, une notification est envoyée à l’administrateur

* + - 1. Monitoring de la performance du modèle
* Consulter les performances du modèle de production

L’évaluation des performances du modèle de production sont basées sur plusieurs critères :

* + Evolution de la précision des prédictions : Basée sur les retours utilisateurs
  + Evolution du temps des prédictions : Afin de se conformer au niveau de service attendu par le client pour ses utilisateurs
  + Détection de drift du modèle en production

Pour cela, l’administrateur a la possibilité de consulter l'historique des prédictions du modèle

Les informations affichées sont collectées en temps réel :

* + - Version du modèle
    - Prédiction
    - Indice de confiance
    - Durée d’exécution de la prédiction
    - Retour utilisateur

Et de consulter l'évolution de l'indicateur de modèle drift

Le drift est basé sur la variation des performances entre les données de référence et les données de productions labellisées par les utilisateurs (on ignore ainsi les données non labellisées)

Deux seuils sont configurés pour détecter un drift de modèle:

* Différentiel des moyennes de prédiction : Seuil par défaut à 0,01
* Différentiel de l’écart type : Seuil par défaut à 0,05

Un drift est détecté lorsqu’un des deux seuils est dépassé.

* + 1. **Consultation des informations des modèles disponibles**
       1. Consultation des métriques

Pour chaque modèle, l’administrateur aura la possibilité de consulter les métriques d’entrainement, ainsi que les données qui ont été utilisées pour l’entrainement

* + 1. **Entrainement d'un modèle**
       1. Entrainement complet

L’administrateur a la possibilité de lancer l’entrainement d’un modèle, ce qui consiste avec notre application à :

* Sélectionner les données à utiliser pour l’entrainement.
* Choisir ou non d’équilibrer la répartition des données. Si c’est le cas, le nombre d’image sélectionné pour chaque classe est le même, et basé sur le nombre d’images disponibles pour la classe minimale.

Lorsque cette option est choisie, l’administrateur a également accès à la liste des images qui ont été sélectionné pour le modèle

* + - 1. Réentrainement avec les données de production

Cette option est celle utilisée en cas de drift d’un modèle.

Alors qu’elle sera automatiquement réalisée dès la détection d’un drift sur le modèle en production, l’administrateur a la possibilité de l’appliquer à des modèles antérieurs.

* + 1. **Déploiement d'un modèle en production**

L’administrateur peut accéder via l’application à l’historique des informations d’entrainement des modèles, et sélectionner ainsi la version candidate.

* + - 1. Préparer au déploiement pour le prochain batch

Lorsque l’administrateur aura identifié une version candidate d’un modèle, il a la possibilité de la préparer pour mise en production

**Techniquement :** Un tag est ajouté au modèle et pourra ainsi être déployé au prochain batch dédié aux fenêtres de maintenance de l’application.

* + - 1. Forcer un déploiement immédiat

L’administrateur a également la possibilité de déployer un modèle immédiatement, ce qui peut être utile dans le cas où un drift du modèle est très important et que le nouveau modèle semble corriger cette dérive

* + 1. **Construction des datasets**

Afin de gérer l’évolution des données, l’administrateur a également la possibilité de forcer la récupération des données ou la fusion des données de production avec les données de référence

* + - 1. Lancer le téléchargement des données

Les données Kaggle ne sont pas mise à jour régulièrement et un batch de téléchargement automatisé n’est pas pertinent si la période de raffraichissement des données n’est pas prévisible.

L’administrateur a donc la possibilité de lancer manuellement le téléchargement des données KAGGLE.

* + - 1. Créer une nouvelle version des données
* Fusionner les données de référence et les données de production, ou kaggle

**Techniquement :** L’intégration de nouvelle données est basée sur le hash md5 des images. Ainsi, seuls les fichiers avec un hash md5 absent de la base de données seront ajoutés.

* + 1. **Traitement des alertes reçues**

L’administrateur reçoit des alertes par mail sur le fonctionnement de l’application. Le type de message reçu :

Dysfonctionnement de l’API (Retour erreur 503) – Date et Heure

* ***Monitoring du niveau de service***
  + Suivre l’état de service de l’interface web

Dysfonctionnement de l’application web (Retour erreur 503) – Date et Heure

* + Suivre l’état de service de l’API

Dysfonctionnement de l’API (Retour erreur 503) – Date et Heure

* ***Monitoring de la performance du modèle***

Une alerte est reçue par l’administrateur lorsqu’un « drift » est détecté, ce qui est un indicateur de l’altération de la performance d’un modèle. Celui-ci est basé sur la précision des prédictions demandées par les utilisateurs et par la validation des prédictions par les utilisateurs et les contributeurs.

Règles de calcul du drift et seuil

* + 1. **Gestion des performances du modèle**
  + Réentrainement du modèle

L’action de réentraînement d’un modèle est manuelle. La décision de relancer un réentrainement dépendra de l’analyse par l’administrateur :

* De la performance et la non régression du nouveau modèle.
* De l’intégration d’une nouvelle espèce.
* Du volume de nouvelles images labellisées (pour les espèces existantes) afin d’améliorer les performances du modèle

Toutes ces décisions de réentrainement du modèle dépendent du volume de données.

* + Déployer un nouveau modèle

Le déploiement d’un nouveau modèle dépendra de la performance cible souhaitée.

# Présentation de l’interface utilisateur – Frontend

***« Streamlit »*** sera utilisé pour développer une interface utilisateur simple et intuitive.

Seront proposés les cas d’usage des utilisateurs et contributeurs, mais également à l’administrateur

Une API sert de passerelle entre les actions demandées par le métier et le Back End.

La page d’accueil invite l’utilisateur à s’authentifier.

Lorsque la connexion est établie, cela activera des sections selon le profil de l’utilisateur.

Pour un utilisateur/contributeur :

* Accueil (Connexion/Déconnexion)
* Page de prédiction
* Mon compte

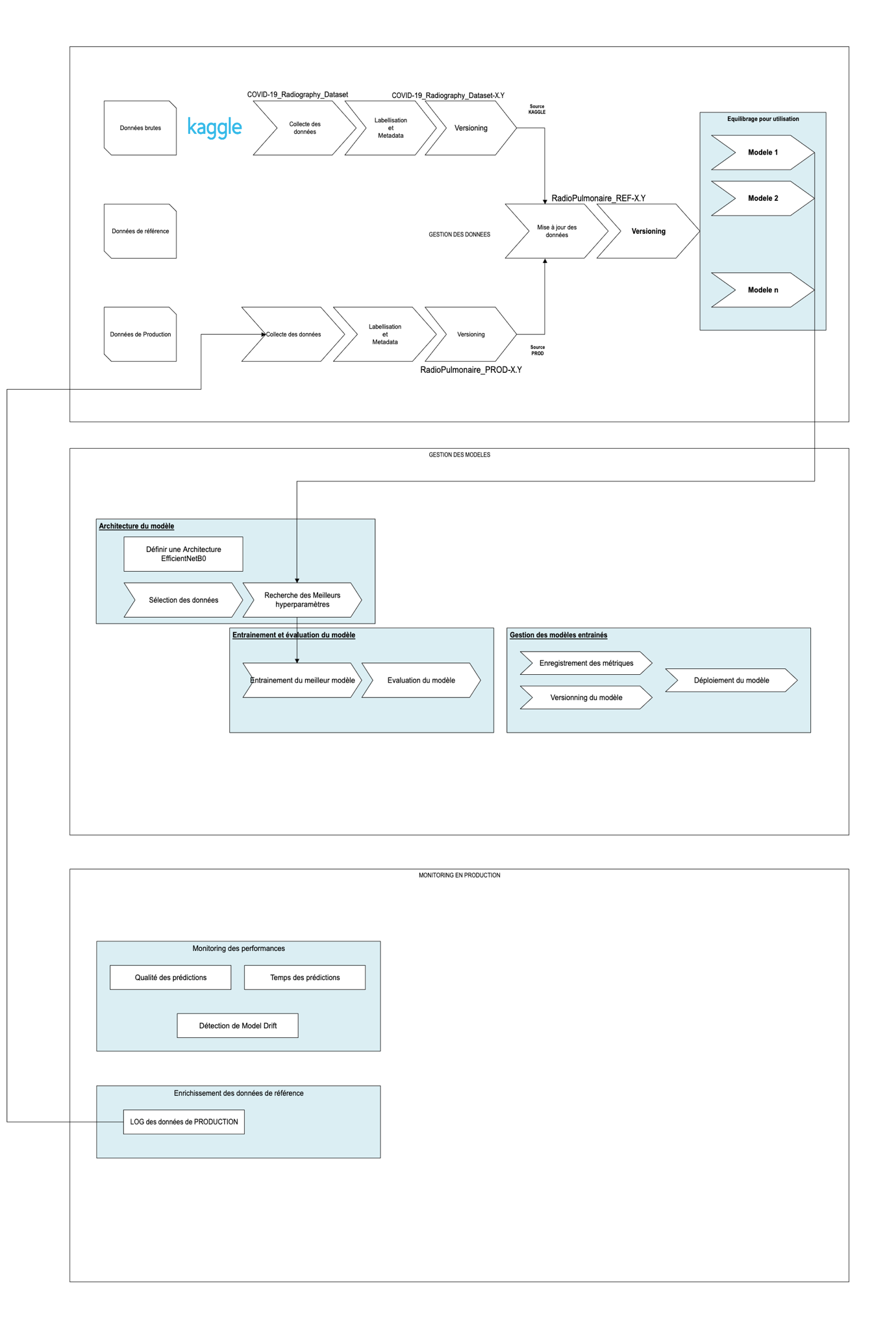
Pour un administrateur, les sections ci-dessus seront complétées par les sections suivantes :

* Etat des services
* Performance du modèle en production
* Evaluation
* Entrainement (de modèle)
* Déploiement (de modèle)
* Datasets (Bases de données)

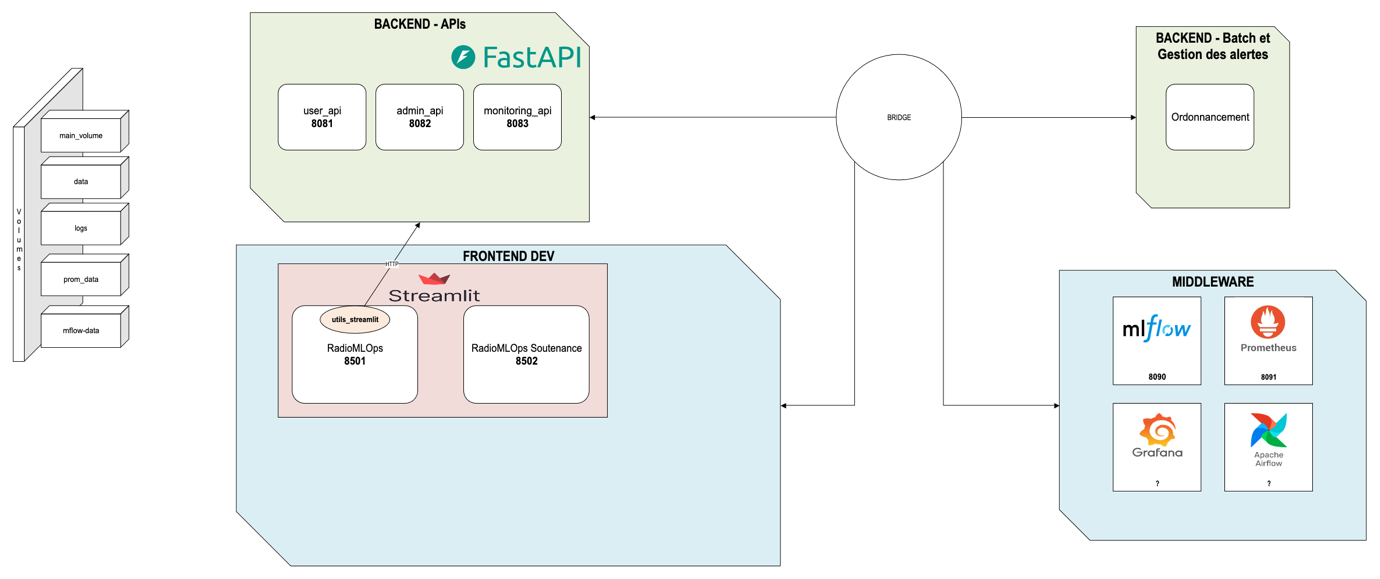
# Règles de gestion - Monitoring de l’application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type d’alerte** | **Exécution** | **Critère d’alerte** | **Action en cas d’alerte** | **Fréquence** |
| Etat des service | Health check des APIs et des applications | Status != 200 | Notification | Toutes les minutes |
| Model Drift | Calcul de drift | Seuil moyenne ou écart type dépassé | Réentrainement  Notification | quotidien |
| Suivi d’une demande d’entrainement | A la demande de l’administrateur depuis l’interface Streamlit | Fin de l’exécution | Notification avec le lien vers l’experiment MLFlow correspondant à l’entrainement | A la demande |
| Suivi d’une demande de calcul de drift | A la demande de l’administrateur depuis l’interface Streamlit | Fin de l’exécution | Notification avec le lien vers l’experiment MLFlow correspondant à l’entrainement | A la demande |
| Suivi des déploiements d’un modèle prêt | A la demande de l’administrateur depuis l’interface Streamlit | Fin de l’exécution | Notification avec le lien vers l’experiment MLFlow correspondant à l’entrainement | A la demande |
|  |  |  |  |  |
|  |  |  |  |  |

# Workflow Global



# Architecture technique Docker



* **Backend APIs :** Ensemble de services exposés pour les différents profils : Utilisateur, Administrateur et Batch. Développés avec FastAPI
* **Backend Batch et gestion des alertes** : Ensemble de scripts exécutés en Crontab
* **Front End DEV :** Interface web développées pour les utilisateurs (et pour la présentation du projet)
* **Middleware :** Ensemble d’outils configurés pour fournir des services web et API complémentaires aux applications : Appel via un navigateurs ou via le code applicatif
* **Volumes partagés entre les différents containers**