Project IOECG Web App : Application Web de Gestion/Exécution d'Algorithmes de Deep Learning

Nom des étudiants : Naoufal BENAMAR, Alioune SECK, Elvarougou HAIBALLA, Baba SOW, Kaouthar BOUROUIS, Lucas ZHENG

Encadrants : M. Thierry HAMON, Mme. Sophie TOULOUSE

Client : M. Ahmad FALL

Année universitaire : 2024/2025

Contents

Introduction

Le principal objectif de ce projet est le développement d'un module de gestion et d'exécution des algorithmes de réseaux de neurones (modèles) à l'aide d'un orchestrateur de calculs HPC. Le module fournit un catalogue détaillé, intuitif, ergonomique et user friendly permettant à l'utilisateur d'explorer les modèles prédéveloppés et d'en sélectionner pour exécution sur des données ECG. Les données ECG sont des matrices de (12x5000) points. L'utilisateur pourra téléverser des ECG sur la plateforme pour exécution ou en sélectionner depuis une base de données contenant des ECG et mise à disposition pour ce projet.

Le module devra s'intégrer dans l'architecture préexistante et s'interfacer avec les services de communication (consul) et d'authentification centralisée (OAuth2/OpenID). La gestion du projet suivra la méthode Agile SCRUM.

Chapter 1

Authentification

1.1 Objectifs

Permettre aux utilisateurs répertoriés dans un annuaire LDAP d'accéder au service de l'application web.

1.2 Exigences Fonctionnelles

- L'application web doit rediriger les utilisateurs vers la page de connexion du système d'authentification centralisé et prêt à l'emploi dans l'environnement cloud UMMISCO, en utilisant le service OAuth 2.0 et OpenID Connect (OIDC).
- L'application web doit attribuer les permissions nécessaires aux utilisateurs en échangeant des informations stockées dans un annuaire LDAP avec le serveur d'authentification via des jetons d'accès (Access Token).
- L'échange entre l'application web et le serveur d'authentification doit se faire par le biais de requêtes et réponses d'API au format JSON.

1.3 Technologies Utilisées

- Frontend : React.js
- Backend : Java (Spring Boot)
- L'implémentation des requêtes et des réponses d'API sera effectuée avec la librairie Axios, permettant d'interagir et d'échanger des ressources entre les différents acteurs du projet, comme le serveur d'authentification centralisé et le planificateur HPC du module M.

Chapter 2

Développement des Fonctionnalités CRUD et Gestion des Projets

2.1 CRUD Projet

2.1.1 Objectif

Permettre aux utilisateurs de créer, consulter, modifier et supprimer des projets sur la plateforme.

2.1.2 Description

Un projet est un environnement de travail étanche dans lequel des expériences ou analyses peuvent être réalisées. Les projets peuvent inclure des données sensibles (comme des données patients/ECG), qui doivent être disponibles de manière sécurisée pour toutes les fonctionnalités métiers. Un projet appartient à son créateur, mais il est possible de l'ouvrir à des collaborateurs avec des rôles spécifiques définis par le créateur ou un administrateur.

2.1.3 Fonctionnalités

- Création de projet avec saisie des détails (nom, description, type de projet).
- Consultation d'une liste de projets dans la section "Mes Projets".
- Modification des détails du projet par l'utilisateur habilité.
- Suppression de projet avec confirmation et vérification de sécurité.

2.1.4 Technologies

- Frontend : React.js
- Backend : Java (Spring Boot)
- Base de données : PostgreSQL

2.1.5 Contraintes de Sécurité

- Authentification sécurisée pour l'accès à la création et à la gestion des projets.
- Permissions vérifiées pour les opérations CRUD.

2.2 CRUD Type Projet

2.2.1 Objectif

Permettre la gestion des types de projets, avec des opérations CRUD pour définir des types de projets paramétrables.

2.2.2 Description

Les types de projets influencent les opérations disponibles et la structure des analyses possibles dans un projet. Chaque type de projet peut être personnalisé en sélectionnant des paramètres, des opérations spécifiques, et en définissant des rôles associés.

2.2.3 Fonctionnalités

- Création de types de projets avec choix de paramètres et d'opérations autorisées.
- Consultation de la liste des types de projets existants.
- Mise à jour des types de projets pour ajouter des paramètres ou modifier des opérations.
- Suppression de types de projets avec vérification de l'impact sur les projets existants.

2.2.4 Contraintes

- Vérification que les types de projets ne peuvent être supprimés s'ils sont utilisés par des projets actifs.

2.3 Partager un Projet et Habilitations

2.3.1 Objectif

Permettre au créateur d'un projet de le partager avec d'autres utilisateurs en leur attribuant des habilitations spécifiques.

2.3.2 Description

Les habilitations sont définies de manière spécifique à chaque projet. Par exemple, un utilisateur peut avoir l'habilitation de supprimer un projet dans un projet A mais pas dans un projet B. Le créateur du projet peut gérer les habilitations des collaborateurs pour contrôler les actions qu'ils peuvent effectuer (ex : lecture, écriture, suppression).

2.3.3 Fonctionnalités

- Ajout de collaborateurs à un projet avec attribution de rôles et d'habilitations spécifiques.
- Visualisation des habilitations des collaborateurs d'un projet.
- Modification des habilitations des collaborateurs.

2.3.4 Technologies

- Frontend : React.js
- Backend : Python (Flask/Django) ou Java (Spring Boot)
- Base de données : PostgreSQL

2.3.5 Sécurité

- Authentification et vérification des permissions avant d'ajouter, modifier, ou consulter les habilitations.

2.4 CRUD Analyse/Expérience

2.4.1 Objectif

Permettre aux utilisateurs habilités de créer, consulter, modifier et supprimer des analyses ou expériences dans un projet.

2.4.2 Description

Une analyse ou expérience représente une action ou un ensemble de tests effectués dans le cadre d'un projet. Les utilisateurs doivent pouvoir gérer ces analyses de manière indépendante tout en étant soumis aux mêmes principes de sécurité que les projets.

2.4.3 Fonctionnalités

- Création d'une analyse avec des détails tels que le nom, la description et les paramètres spécifiques.
- Consultation des analyses existantes dans un projet.
- Modification des analyses par les utilisateurs ayant les droits appropriés.
- Suppression des analyses avec confirmation et vérification de sécurité.

2.4.4 Technologies

- Frontend : React.js
- Backend : Python (Flask/Django) ou Java (Spring Boot)
- Base de données : PostgreSQL

2.4.5 Sécurité

- Vérification des droits avant chaque opération CRUD sur les analyses.
- Logs d'audit pour suivre les modifications et suppressions des analyses.

Chapter 3

Catalogue/Bibliothèque de modèles IA : Vue et exploration de la bibliothèque de modèles IA

3.1 Objectifs

Améliorer la bibliothèque existante de modèles IA pour offrir une vue ergonomique permettant l'exploration des modèles, l'affichage de leurs métadonnées, et l'historique des modifications et opérations. Implémenter des filtres et des mécanismes de recherche pour faciliter la navigation et l'accès aux modèles. Mettre en place des habilitations pour restreindre l'accès aux modèles en fonction des projets et analyses auxquels ils sont rattachés.

3.2 Fonctionnalités

3.2.1 Exploration des modèles IA :

- Fournir une interface utilisateur conviviale qui permet de parcourir l'ensemble des modèles disponibles.
- Chaque modèle doit être présenté avec ses métadonnées (nom, version, auteur, type de modèle, date de création, etc.) et l'historique des modifications et des opérations (ex : mises à jour, utilisations).

3.2.2 Filtres et mécanismes de recherche :

- Implémentation des filtres pour trier et sélectionner les modèles selon divers critères tels que le type de modèle, la version, la date de création, l'auteur, etc.
- Ajout d'une barre de recherche permettant de trouver des modèles en utilisant des mots-clés

3.2.3 Historique des modifications et opérations :

- Affichage de l'historique détaillé des modifications pour chaque modèle, incluant les changements de version, les modifications des métadonnées et les opérations réalisées

sur le modèle (ex : entraînement, validation).

3.2.4 Rattachement des modèles aux analyses et projets :

- Les modèles doivent être rattachés à des analyses spécifiques, elles-mêmes associées à des projets. Cela permet de mieux organiser les modèles et de comprendre leur contexte d'utilisation.

3.3 Gestion des habilitations et des droits d'accès

3.3.1 Contrôle des accès :

- Implémentation d'un système de gestion des habilitations pour restreindre l'accès aux modèles selon les droits attribués aux utilisateurs. Ces habilitations doivent pouvoir être configurées au niveau des projets et des analyses.
- Exemples de permissions : lecture seule, modification, suppression, exécution du modèle.

3.3.2 Rattachement des modèles aux analyses et projets :

- Permettre aux administrateurs de projet ou aux créateurs d'analyse de gérer les permissions d'accès aux modèles pour chaque utilisateur participant au projet.

3.4 Technologies et intégration

3.4.1 Technologies existantes:

- La bibliothèque utilise des modèles au format ONNX, majoritairement implémentés avec Apache MXNet.
- Les métadonnées et l'historique des modèles sont stockés dans la base de données MongoDB.

3.4.2 Intégration frontend :

- une interface en **React.js** pour offrir une navigation fluide et intuitive dans la bibliothèque.

3.4.3 Intégration backend :

- le backend (en Java avec **Spring Boot**) pour gérer les requêtes de recherche, l'affichage des métadonnées et la vérification des habilitations.

3.4.4 Sécurité et authentification :

- Utilisation du **JWT** ou un autre mécanisme d'authentification pour vérifier l'identité des utilisateurs et leurs permissions.

3.5 Vue ergonomique et fonctionnalités avancées

3.5.1 Tableau de bord de la bibliothèque :

- Présentation des modèles sous forme de liste ou de tuiles, avec un aperçu rapide des métadonnées principales.
- Ajout des boutons ou des icônes permettant d'accéder directement aux détails du modèle, de consulter l'historique ou de gérer les permissions.

3.5.2 Affichage des détails et historique :

- Lorsqu'un utilisateur clique sur un modèle, une page de détails doit s'ouvrir avec toutes les métadonnées et l'historique des modifications

Chapter 4

Explication Conceptuelle de la Structure des Datasets

4.1 Qu'est-ce qu'un Dataset ?

Un dataset représente un ensemble de données filtrées, regroupant des ECGs et des informations sur les patients. Les utilisateurs peuvent appliquer des filtres spécifiques pour extraire uniquement les données nécessaires (par exemple, par visite, date, ou données cliniques) et créer un dataset nommé. Chaque dataset est associé à un projet et peut être réutilisé dans d'autres projets si les permissions le permettent.

4.2 Lien entre les Datasets et les Projets

- Un projet peut contenir plusieurs datasets.
- Les datasets peuvent être partagés entre plusieurs projets, selon les permissions définies.

4.3 Exemple de Dataset en JSON

```
{
  "id": "dataset-123",
  "name": "Dataset ECG Projet A",
  "projectId": "projet-001",
  "createdAt": "2024-11-10",
  "updatedAt": "2024-11-10",
  "filters": {
    "visit": "première_consultation",
    "date": "2024-11-09",
    "timeRange": ["10:00", "12:00"],
    "clinicalData": {
      "age": [30, 50],
      "diagnosis": "Arythmie"
    }
  },
```

```
  "patients": [
    {
      "id": "patient-001",
      "name": "John Doe",
      "ecgs": [
        {
          "id": "ecg-001",
          "date": "2024-11-09",
          "time": "10:30",
          "data": "signal_données_ici"
        }
      ]
    }
  ],
  "permissions": {
    "read": ["utilisateur1", "utilisateur2"],
    "write": ["admin"]
  }
}
```

4.4 Opérations CRUD pour les Datasets via une API

4.4.1 Créer un Dataset (POST)

- Endpoint : ‘/datasets‘
- Description : Permet de créer un dataset en définissant ses attributs et ses filtres.

4.4.2 Lire un Dataset (GET)

- Endpoint : ‘/datasets/id‘
- Description : Récupère les détails d’un dataset à partir de son ID.

4.4.3 Mettre à Jour un Dataset (PUT)

- Endpoint : ‘/datasets/id‘
- Description : Permet de modifier les informations ou les filtres d’un dataset existant.

4.4.4 Supprimer un Dataset (DELETE)

- Endpoint : ‘/datasets/id‘
- Description : Supprime un dataset identifié par son ID.

Chapter 5

Exécution de Modèles sur des Données de Signaux ECG

5.1 Objectif

Développer une fonctionnalité permettant l'exécution d'un ou de plusieurs modèles sur des données de signaux ECG. Cette exécution sera orchestrée via le HPC Scheduler (M8) pour garantir une allocation efficace des ressources de calcul sur un cluster de haute performance. L'objectif est de fournir un pipeline d'exécution automatisé qui gère la sélection des modèles, le choix des données, et le suivi de l'exécution.

5.2 Fonctionnalités

5.2.1 Pré-requis : Sélection de Modèles et Données

- Catalogue des modèles (M7 - 41&42) : L'utilisateur peut choisir un ou plusieurs modèles de machine learning dans le catalogue.
- Intégrer une interface permettant de sélectionner un ou plusieurs modèles prédéfinis pour l'inférence.
- Permettre aux utilisateurs de sélectionner explicitement des datasets ECG ou de rechercher des fichiers ECG spécifiques.

5.2.2 Génération et Transmission de la Définition de Pipeline

- Après la sélection des modèles et des données, le système génère automatiquement une définition de pipeline au format JSON et la transmet au HPC Scheduler.
- Structure du document JSON :
  - Modèles : Liste des modèles choisis, avec leurs métadonnées (ID, version, type d'entrée, sortie attendue).
  - Données : Liste des datasets sélectionnés, incluant les chemins ou identifiants des données ECG.
  - Paramètres d'exécution : Options pour ajuster l'allocation des ressources (ex : CPU, GPU).

- S'assurer que les informations de configuration et de ressources respectent les spécifications requises pour chaque modèle et jeu de données.
- Transmission du document JSON au HPC Scheduler pour l'ordonnancement et l'exécution des tâches sur le cluster de calcul.

5.2.3 Orchestration de l'Exécution via le HPC Scheduler (M8)

- Le HPC Scheduler est chargé de gérer l'ordonnancement des jobs et l'affectation des ressources (CPU, GPU, mémoire) nécessaires pour chaque modèle.
- Chaque tâche dans le pipeline doit être exécutée en parallèle ou en séquence selon les dépendances définies dans le document JSON.
- Les jobs doivent s'exécuter dans un environnement sécurisé et isolé, avec optimisation de la consommation des ressources.

5.2.4 Suivi de l'Exécution et Statut des Pipelines

- Fournir une interface de suivi en temps réel de l'état de chaque pipeline (en cours, terminé, en échec).
- Le suivi inclut des informations sur la consommation des ressources (CPU, GPU, mémoire) pour chaque tâche, récupérées via l'API du HPC Scheduler.
- Envoyer des notifications en cas d'échec ou d'interruption de l'exécution pour permettre une reprise ou un diagnostic rapide.
- Intégrer des commandes permettant d'interrompre ou de relancer un pipeline sans affecter les autres tâches en cours.

5.3 Technologies

- HPC Scheduler (M8) : Orchestrateur principal des tâches d'exécution, permettant la gestion des ressources et l'optimisation des calculs.
- API REST : Interface de soumission des modèles, des données, et de suivi des exécutions.
- Formats JSON : Utilisés pour la transmission de la configuration du pipeline à l'orchestrateur.
- Interface Utilisateur : Permettant la sélection des modèles, des données, et le suivi en temps réel des exécutions.

5.4 Contraintes et Exigences

- Scalabilité : Le système doit être capable de gérer plusieurs pipelines en parallèle, en fonction des ressources disponibles sur le cluster HPC.

- Sécurité : Chaque tâche doit s'exécuter dans un environnement isolé pour protéger les données et les ressources.
- Robustesse : Garantir que l'orchestrateur gère les interruptions et permet une reprise rapide des pipelines.
- Compatibilité : Les modèles doivent être compatibles avec le format JSON défini pour le HPC Scheduler et l'infrastructure de calcul.

Chapter 6

Génération du Rapport Final

6.1 Objectifs

- Fournir un rapport consolidé des résultats des analyses effectuées sur des données ECG en utilisant des modèles de deep learning.
- Mettre à disposition des médecins un rapport structuré, intégrant des visualisations et des tableaux récapitulatifs pour une interprétation rapide et efficace et une prise de décision facile.
- Permettre l'export du rapport au format PDF pour archivage, partage et collaboration.

6.2 Exigences Fonctionnelles

6.2.1 Service de Génération de Rapports

L'application doit permettre de générer des rapports basés sur une ou plusieurs analyses spécifiques, sur un projet complet, ou sur un pipeline d'exécution. Chaque rapport inclura un résumé des données avec des interprétations, ainsi que des visualisations sous forme de graphiques et de tableaux pour faciliter la compréhension. Il mettra en avant les éléments clés issus des analyses réalisées par les modèles de deep learning, permettant ainsi aux utilisateurs de disposer d'une vue d'ensemble claire et synthétique des résultats obtenus.

6.2.2 Export PDF

Le rapport généré sera exporté au format PDF, permettant aux utilisateurs de le télécharger directement depuis l'interface utilisateur pour un accès facile et une consultation hors ligne.

6.3 Contraintes Techniques

6.3.1 Backend

- Spring Boot pour développer l'API **REST** permettant la récupération des données nécessaires (signaux ECG, résultats des modèles IA, etc.), la génération et l'export des rapports.

- Intégration des bibliothèques telles que **iText** pour la génération et la personnalisation des fichiers PDF.

6.3.2 Frontend

- React pour l'interface utilisateur, permettant aux utilisateurs de déclencher la génération du rapport et de télécharger le **PDF**.
- Intégration d'**Axios** pour les requêtes **HTTP** vers l'**API** backend.