**1. Fonctionnalités principales**

**Réception, modification et émission de fichiers :**

* Formats supportés : HL7, HPRIM XML, CSV.
* Modifications effectuées selon des règles personnalisables.

**Modes de transmission :**

* Réception et émission via TCP/IP.
* Réception et émission via SFTP.

**Interface utilisateur :**

* Application web avec interface graphique développée en **Flet**.
* Authentification utilisateur.

**Fonctionnement :**

* Fonctionne comme un service sur un serveur Windows.

**2. Architecture et arborescence des fichiers**

**Racine du projet**

bash

Copier le code

EAI\_Medical/

├── app.py # Point d'entrée principal de l'application.

├── config/

│ ├── settings.py # Configuration des paramètres (TCP/IP, SFTP, etc.).

│ └── logging.conf # Configuration des logs.

├── core/

│ ├── hl7\_processor.py # Gestion et traitement des fichiers HL7.

│ ├── hprim\_processor.py # Gestion et traitement des fichiers HPRIM XML.

│ ├── csv\_processor.py # Gestion et traitement des fichiers CSV.

│ ├── file\_manager.py # Gestion des fichiers reçus/envoyés.

│ └── transmission.py # Gestion des transmissions TCP/IP et SFTP.

├── services/

│ ├── service\_manager.py # Gestion du service Windows.

│

graphql

Copier le code

├── user\_auth/

│ ├── auth.py # Gestion de l'authentification des utilisateurs.

│ ├── models.py # Modèles pour les utilisateurs (base de données).

│ └── routes.py # Routes liées à l'authentification (login, logout).

├── web/

│ ├── interface.py # Interface graphique en Flet.

│ └── templates/ # Templates pour l'affichage web.

├── utils/

│ ├── hl7\_parser.py # Fonctions utilitaires pour traiter les fichiers HL7.

│ ├── xml\_utils.py # Fonctions utilitaires pour traiter XML (HPRIM).

│ ├── csv\_utils.py # Fonctions utilitaires pour le CSV.

│ ├── tcp\_utils.py # Fonctions utilitaires pour TCP/IP.

│ └── sftp\_utils.py # Fonctions utilitaires pour SFTP.

├── tests/

│ ├── test\_hl7.py # Tests pour les fichiers HL7.

│ ├── test\_hprim.py # Tests pour les fichiers HPRIM XML.

│ ├── test\_csv.py # Tests pour les fichiers CSV.

│ ├── test\_auth.py # Tests pour l'authentification.

│ └── test\_service.py # Tests pour les services (TCP/IP, SFTP).

└── requirements.txt # Liste des dépendances Python.

**3. Détails des composants**

**1. Transmission et réception**

* **TCP/IP** :
  + Implémentation avec le module socket de Python.
  + Écoute sur un port défini (paramétrable dans settings.py).
* **SFTP** :
  + Utilisation de la bibliothèque paramiko.

**2. Traitement des fichiers**

* **HL7** :
  + Utilisation de bibliothèques comme hl7 ou hl7apy.
* **HPRIM XML** :
  + Manipulation avec xml.etree.ElementTree ou lxml.
* **CSV** :
  + Gestion via le module natif csv.

**3. Interface utilisateur**

* Interface en **Flet** :
  + Dashboard pour visualiser les fichiers reçus/envoyés.
  + Section pour configurer les paramètres réseau (TCP/IP, SFTP).
  + Section pour authentification (login/logout).

**4. Authentification**

* Implémentation avec des **tokens JWT**.
* Gestion des utilisateurs via une base de données SQLite ou PostgreSQL.

**5. Service Windows**

* Création d’un service avec la bibliothèque pywin32.

**4. Exemple de requirements.txt**

plaintext

Copier le code

flask

flet

hl7

lxml

paramiko

pywin32

pyjwt

pytest

**5. Étapes principales**

1. **Configuration** :
   * Créez settings.py pour stocker les paramètres comme les ports et les identifiants SFTP.
2. **Interface graphique** :
   * Construisez une interface utilisateur intuitive avec Flet.
3. **Traitement des fichiers** :
   * Implémentez des fonctions robustes pour lire, modifier et valider les formats HL7, HPRIM XML et CSV.
4. **Transmission** :
   * Développez des classes pour gérer les connexions TCP/IP et SFTP.
5. **Authentification** :
   * Ajoutez des fonctionnalités de login/logout et des protections JWT.
6. **Déploiement** :
   * Configurez l'application pour tourner comme un service sous Windows.

Si vous souhaitez que je développe des parties spécifiques ou fournisse des exemples de code, faites-le-moi savoir !

**Phase 1 : Analyse et planification (1 semaine)**

**Objectifs :**

* Définir clairement les besoins fonctionnels et techniques.
* Identifier les ressources nécessaires (bibliothèques, outils, compétences).
* Valider l'architecture logicielle et l'arborescence des fichiers.

**Tâches :**

1. Analyser les formats HL7, HPRIM XML, et CSV pour identifier les règles de traitement spécifiques.
2. Décider des détails d'implémentation des protocoles TCP/IP et SFTP.
3. Rédiger des spécifications techniques pour les modules :
   * Gestion des fichiers.
   * Transmission.
   * Authentification.
   * Interface utilisateur.
4. Créer une maquette de l'interface utilisateur avec Flet.

**Phase 2 : Mise en place des bases de l'application (2 semaines)**

**Objectifs :**

* Préparer l'infrastructure logicielle de l'application.
* Implémenter les fonctionnalités fondamentales.

**Tâches :**

1. **Initialisation du projet :**
   * Configurer le dépôt Git (GitHub, GitLab, etc.).
   * Créer la structure de fichiers et installer les dépendances via requirements.txt.
2. **Configuration :**
   * Implémenter les paramètres réseau et les configurations dans settings.py.
   * Ajouter un fichier de configuration pour les logs.
3. **Backend de traitement des fichiers :**
   * Implémenter les modules hl7\_processor.py, hprim\_processor.py, et csv\_processor.py.
   * Créer des fonctions de test unitaire pour chaque module.
4. **Transmission :**
   * Développer et tester l'envoi/réception de données en TCP/IP.
   * Configurer un environnement SFTP pour tester la transmission des fichiers.
5. **Authentification :**
   * Ajouter l'authentification avec JWT dans auth.py et tester les routes d'authentification.

**Phase 3 : Développement avancé (3 semaines)**

**Objectifs :**

* Ajouter des fonctionnalités secondaires.
* Développer et tester l'interface utilisateur.

**Tâches :**

1. **Interface graphique :**
   * Créer une interface utilisateur interactive avec Flet (interface.py).
   * Ajouter des fonctionnalités pour :
     + Visualiser les fichiers reçus.
     + Modifier les configurations réseau (port TCP, SFTP, etc.).
2. **Service Windows :**
   * Implémenter le service Windows avec pywin32 (service\_manager.py).
   * Tester le démarrage, l'arrêt et la gestion du service.
3. **Traitement des erreurs :**
   * Ajouter une gestion robuste des erreurs pour les modules de traitement de fichiers.
   * Loguer les erreurs critiques (HL7 mal formé, échec de connexion TCP/SFTP).
4. **Tests :**
   * Écrire des tests pour tous les modules dans le dossier tests/.
   * Tester les fonctionnalités de bout en bout (reception -> modification -> émission).

**Phase 4 : Validation et déploiement (2 semaines)**

**Objectifs :**

* Tester l'application dans un environnement réel.
* Préparer la documentation et les guides d’utilisation.

**Tâches :**

1. **Tests finaux :**
   * Tester l'application sur un serveur Windows.
   * Valider les transmissions TCP/IP et SFTP avec des données réelles.
   * Effectuer des tests de charge pour évaluer les performances.
2. **Documentation :**
   * Documenter le code (docstrings, commentaires).
   * Rédiger un guide utilisateur pour l'interface.
   * Créer un guide d'installation et de configuration.
3. **Déploiement :**
   * Packager l'application pour un déploiement facile (fichier EXE ou script d’installation).
   * Configurer le service sur le serveur Windows de production.

**Phase 5 : Maintenance et améliorations continues (ongoing)**

**Objectifs :**

* Ajouter des fonctionnalités basées sur les retours d'utilisateurs.
* Corriger les bogues et optimiser l'application.

**Tâches :**

1. Mettre en place un système de tickets pour le suivi des bugs et des améliorations.
2. Ajouter des fonctionnalités supplémentaires, comme :
   * Envoi automatique des notifications en cas d'échec de transmission.
   * Intégration avec des bases de données pour stocker les logs.
   * Export des fichiers traités dans d'autres formats.
3. Planifier des mises à jour régulières pour suivre l’évolution des formats HL7 et HPRIM.

**Timeline globale**

| **Phase** | **Durée** | **Échéance** |
| --- | --- | --- |
| Analyse et planification | 1 semaine | Semaine 1 |
| Mise en place des bases | 2 semaines | Semaine 3 |
| Développement avancé | 3 semaines | Semaine 6 |
| Validation et déploiement | 2 semaines | Semaine 8 |
| Maintenance | Continu | À partir de la semaine 9 |

Cette roadmap peut être ajustée en fonction des priorités et des ressources disponibles. Si vous avez des questions ou souhaitez développer un point spécifique, je suis à votre disposition.

Créer une interface "no code" pour permettre aux utilisateurs d'appliquer des modifications sur les messages (HL7, HPRIM, CSV) avant de les renvoyer est une excellente idée. Voici une solution détaillée pour intégrer cette fonctionnalité dans votre application.

**1. Fonctionnalités de l'interface "no code"**

**Objectifs :**

1. **Création de règles personnalisées** :
   * Ajouter des règles pour modifier, ajouter ou supprimer des champs spécifiques.
   * Appliquer des transformations (par exemple, convertir une date au format ISO, ajouter un préfixe à un champ).
2. **Visualisation des messages** :
   * Afficher un aperçu des messages avant et après application des règles.
3. **Enregistrement des règles** :
   * Permettre aux utilisateurs de sauvegarder les règles sous forme de modèles réutilisables.
4. **Validation** :
   * Vérifier que les modifications respectent le format du fichier (HL7, XML, CSV).

**2. Workflow utilisateur**

1. **Chargement des fichiers** :
   * L'utilisateur charge un fichier HL7, XML ou CSV.
2. **Création ou sélection des règles** :
   * L'utilisateur crée de nouvelles règles ou sélectionne un modèle existant.
3. **Application des règles** :
   * Les modifications sont appliquées sur un aperçu du message.
4. **Validation et enregistrement** :
   * L'utilisateur valide les modifications et renvoie le message modifié.

**3. Structure technique**

**Modules nécessaires :**

1. **Gestion des règles** :
   * Ajouter un module core/rule\_engine.py pour gérer les règles.
   * Sauvegarder les règles sous forme de fichiers JSON ou dans une base de données.
2. **Interface utilisateur** :
   * Utiliser Flet pour créer une interface visuelle pour la configuration des règles.
3. **Application des règles** :
   * Modifier les modules hl7\_processor.py, hprim\_processor.py, et csv\_processor.py pour intégrer les règles.

**Améliorations possibles**

1. **Règles conditionnelles** :
   * Ajouter des conditions pour appliquer les règles (exemple : "Si le champ X contient Y, alors modifier Z").
2. **Sauvegarde des configurations utilisateur** :
   * Associer des règles spécifiques à des utilisateurs via l'authentification.
3. **Aperçu dynamique** :
   * Ajouter une vue en temps réel des modifications dans l'interface Flet.
4. **Validation des règles** :
   * Vérifier les conflits entre les règles (exemple : suppression et modification du même champ).