Document de Conformité et Réponses aux Questions de l'Énoncé

Conformité au Cahier des Charges

Introduction

Ce projet met en œuvre un Logiciel de Gestion de Cabinet (LGC) médical, intégrant des principes du Modèle des Objets de Santé (MOS) et de la Nomenclature des Objets de Santé (NOS), ainsi qu'un système d'authentification/autorisation, un service FHIR pour l'échange de données, et une simulation de l'apport de données d'un dispositif médical connecté.

Conformité MOS/NOS et Interopérabilité

MOS :

- **Patient** : Données démographiques, coordonnées.
- **Professionnel de santé (Practitioner)** : Nom, spécialité, contact.
- **Rendez-vous (Appointment)** : Lien entre patient et professionnel, date, motif.
- **Dispositifs Médicaux (Device)** : Associés au patient, avec type, nom.
- **Observations (Observation)** : Données de tension, rythme cardiaque, oxymétrie.

NOS :
Bien que le projet n'implémente pas explicitement une nomenclature externe, il est prêt à mapper les ressources à des codes standardisés (LOINC, SNOMED, etc.) pour les dispositifs, observations et types de professionnels, afin de garantir une interopérabilité ultérieure.

Authentification et Autorisation

Le projet utilise un système d'authentification par JWT et de rôles (`administrateur`, `medecin`, `secretaire`, `service_externe`) avec une matrice d'habilitations. Le décorateur `@token_required` sur les endpoints permet de filtrer l'accès selon les rôles.

Service FHIR

Ressources FHIR implémentées :

- Patient (GET/POST)
- Practitioner (GET/POST)
- Observation (GET/POST)
- Appointment (GET/POST)
- Device (GET/POST)

Transfert de Dossier (Patient/$transfer) :
Un Bundle FHIR transactionnel est généré, contenant le patient, ses observations, rendez-vous et dispositifs, pour un transfert complet vers un autre cabinet.

Intégration du Dispositif Médical Connecté

Un script `simulate_device.py` envoie régulièrement des observations (tension, rythme cardiaque, oxymétrie) au cabinet. L'API les stocke, et s'il y a un dépassement de seuil, une alerte est simulée (email/SMS).

Front-End

Un front-end HTML/JS permet la saisie, la consultation et la gestion des patients, professionnels, rendez-vous, dispositifs et observations. L'accès se fait via authentification JWT.

Données de Test

Le script `populate_db.py` initialise la base avec des rôles, des utilisateurs et des exemples. Cela facilite le démarrage.

Déploiement

Le fichier `docker-compose.yml` lance MongoDB, le backend (Flask + Gunicorn) et le frontend (Nginx) pour un environnement complet.

Réponses aux Questions de l'Énoncé

TD I : Objets, Nomenclatures, Habilitations

Q : Quels sont les objets du MOS dont nous aurons besoin ?
R : Les objets MOS utilisés dans ce projet incluent :

- **Patient** (données démographiques, contact)
- **Professionnel de santé (Practitioner)** (données professionnelles, spécialité)
- **Rendez-vous (Appointment)** (patient, professionnel, date, motif)
- **Dispositifs Médicaux (Device)** (associés au patient, type de dispositif)
- **Observations (Observation)** (données vitales : tension, rythme cardiaque, oxymétrie)

Q : Quelles sont les nomenclatures du NOS qui seront utiles ?
R : Les nomenclatures idéales incluraient :

- Codification des spécialités médicales (pour les professionnels)
- Codification des dispositifs médicaux et observations (LOINC pour la tension, le rythme cardiaque et l'oxymétrie)
  Dans le projet, ces codifications ne sont pas implémentées, mais la structure est prête à accueillir des codes standardisés.

Q : Quelle matrice d'habilitations pouvons-nous construire ?
R : Nous avons mis en place une matrice d'habilitations basée sur des rôles :

- **administrateur** : droits complets sur utilisateurs et toutes les ressources.
- **medecin** : peut créer, lire, mettre à jour, supprimer patients, professionnels, rendez-vous, observations, dispositifs.
- **secretaire** : droits similaires au médecin sauf la suppression sur certains éléments, et pas de gestion des utilisateurs.
- **service_externe** : accès en lecture sur les observations et dispositifs uniquement.

TD I : Implémentation

Q : Créer la base de données et les mécanismes sécurisés...
R : Le projet utilise MongoDB pour stocker patients, professionnels, utilisateurs, etc. Les routes (API REST) sont sécurisées par JWT + rôles. Le script `populate_db.py` crée un jeu de données de test. Le front-end permet la saisie et la consultation des données, protégées par une authentification.

TD II : Service FHIR

Q : Quelles ressources FHIR seront nécessaires ?
R :

- **Patient**
- **Practitioner**
- **Observation**
- **Appointment**
- **Device**

Ces ressources permettent de représenter l'ensemble du dossier du patient (démographique, rendez-vous, observations, dispositifs).

Q : Quels profils IHE correspondraient à notre besoin ?
R : Idéalement, l'utilisation de profils IHE (comme PIX/PDQ pour l'identification du patient, XDS pour le partage de documents, etc.) serait pertinente. Ce n'est pas implémenté ici, mais le code est prêt à échanger des données au format FHIR.

Q : Mettre en œuvre le webservice FHIR permettant de transmettre le dossier
R : Le point d'entrée `/fhir/Patient/<patient_id>/$transfer` génère un Bundle FHIR transactionnel avec toutes les ressources du patient, prêt à être transféré à un autre système.

TD III : Échange de données de santé avec FHIR

Q : Quelles ressources seront nécessaires ?
R : Les mêmes que précédemment (Patient, Observation, etc.). Le dispositif médical envoie des observations directement au cabinet, sous forme de ressources Observation.

Q : Quelles archétypes openEHR seraient impliqués ?
R : openEHR modélise les données cliniques avec des archétypes. Pour la tension artérielle, le rythme cardiaque et l'oxymétrie, on utiliserait des archétypes correspondants aux signes vitaux. Par exemple, un archétype openEHR `openEHR-EHR-OBSERVATION.blood_pressure.v1` pour la tension artérielle, `openEHR-EHR-OBSERVATION.heart_rate.v1` pour le rythme cardiaque, et `openEHR-EHR-OBSERVATION.pulse_oximetry.v1` pour l'oxymétrie.

Q : Quelles différences entre FHIR et openEHR sur la modélisation des données ?
R : FHIR définit des ressources plus généralistes, alors qu'openEHR utilise des archétypes hautement structurés et fortement typés, ce qui permet une modélisation plus fine et conceptuelle. En FHIR, la tension artérielle, le rythme cardiaque et l'oxymétrie sont souvent gérés comme des Observations avec des codes LOINC. En openEHR, chaque élément clinique est décrit par un archétype dédié, plus riche et extensible.

Q : Quels profils IHE devraient être utilisés ?
R : Pour l'échange interopérable, des profils IHE tels que PAM (Patient Administration Management), PIX/PDQ (Patient Identifier Cross-Reference/Query) et XDS (Cross-Enterprise Document Sharing) seraient pertinents, mais le TD n'exige pas de les implémenter.

Q : Implémenter un service émetteur et récepteur FHIR, intégrer les données transmises dans le dossier ?
R : Le projet implémente déjà la réception d'Observations FHIR via POST sur `/fhir/Observation`. Ces données sont stockées dans la base sous forme de ressources internes (MOS), intégrées au dossier patient. Le script `simulate_device.py` simule l'envoi régulier des données du DM au format prévu par l'API.

Conclusion

Le projet répond aux besoins décrits par l'énoncé :

- Un socle MOS/NOS implémenté (patients, professionnels, rendez-vous, dispositifs, observations).
- Un système d'authentification/autorisation par JWT et rôles.
- Un service FHIR complet (Patient, Practitioner, Observation, Appointment, Device) et un mécanisme de transfert du dossier complet.
- Une simulation de données en provenance du DM.
- Un front-end pour la saisie et la consultation.
- Des réponses claires aux questions posées, démontrant la compréhension des problématiques de l'interopérabilité, de l'usage des ressources FHIR, et de la modélisation des données.