Interopérabilité avancée en santé

Travaux dirigés

Leo DEVIN (leo.devin@epita.fr)
Phu-Hung DANG(phu-hung.dang@epita.fr)

Un cabinet médical situé à Paris est constitué de deux médecins cardiologues et un secrétariat[1]*.*

Le secrétariat accède au dossier administratif du patient et à la partie de son dossier médical qui contient les comptes rendus.

Un patient, envoyé par son médecin traitant (hors périmètre), doit porter sur lui un dispositif médical (DM) connecté qui transmet à intervalles réguliers des données de tension artérielle, de rythme cardiaque et d'oxymétrie au cabinet médical.

Un service (hors périmètre) compile les données en un compte rendu quotidien, et peut émettre des alertes qui, suivant les paramètres choisis par le médecin, lui envoient un mail et éventuellement un SMS aux urgences (hors périmètre) et à un proche identifié dans le dossier patient.

Nous sommes dans un monde idéal où tous les standards d'interopérabilité sont appliqués.

Nous allons ébaucher le Logiciel de Gestion de Cabinet (LGC) du médecin, avec ses services d'interopérabilité, et simuler la transmission des données du DM.

NB : c'est un TD et non un projet industriel. Vous avez le droit d'utiliser des briques *open source* sans vergogne, en citant les sources et les modifications qui ont été faites…

[1] Pour éviter toute polémique, j'emploie systématiquement le genre neutre qui, en français, ne se distingue pas du masculin.

TD I *Créer un socle de description conforme aux standards d'interopérabilité.*

Le Modèle des Objets de Santé (MOS) a pour ambition de décrire les objets de l'environnement de santé indépendamment des technologies, sous forme de diagrammes UML. On peut décrire ainsi un professionnel de santé, ses caractéristiques professionnelles et ses activités, ses ressources, ses lieux d'activités et leur structuration, son agenda…
On peut aussi décrire un patient, avec tout son environnement, et son dossier.

La Nomenclature des Objets de Santé (NOS) permet de codifier les éléments du NOS, par exemple les types d'établissement ou d'intervention.

MOS et NOS permettent donc de décrire le fonctionnement d'une structure de santé, mais pas le contenu des activités. Vous n'y trouverez pas la description d'une tension artérielle, ni les médicaments qui traitent l'hypertension, mais l'hypertension artérielle a l'identifiant 1.2.250.1.213.3.3.11/520 dans la description des actes et la prise de tension en continu en ambulatoire est l'acte 1.2.250.1.213.3.3.11/934 en complément d'un bilan radiologique d'hypertension artérielle 1.2.250.1.213.3.3.11/1143.

Cela permet donc d'avoir un socle commun pour un logiciel devant s'intégrer à un SI de santé (hôpital ou clinique, cabinet, laboratoire…).

On va aussi pouvoir, grâce à des matrices d'habilitations, savoir qui accède à quoi, sujet très sensible !

L'objectif de ce TD est d'implémenter un sous-ensemble du MOS et du NOS avec un système d'authentification/autorisation permettant de gérer les habilitations.

Pour ce projet complexe de gestion de cabinet médical, voici comment nous avons abordé le problème et organisé le développement :

1. **Approche et Définition du Problème**
Nous avons commencé par analyser les exigences du projet, en nous concentrant sur les éléments essentiels : l'interopérabilité des données médicales, la gestion sécurisée des utilisateurs, la synchronisation avec les dispositifs médicaux connectés, et l'authentification des utilisateurs. Nous avons identifié les composants clés de l'application, à savoir : la gestion des patients, des professionnels de santé, des dispositifs médicaux, ainsi que l'intégration des normes MOS/NOS pour garantir la conformité.

2. **Méthodologie Adoptée**

Planification : Nous avons défini un plan de développement en plusieurs étapes, structurant l'avancement du projet :

- Mise en place des bases de données et de Docker. (Leo)
- Configuration du backend avec Node.js, Express, et MongoDB (avec une possibilité d'intégration de PostgreSQL pour Keycloak). (Leo)
- Développement du frontend avec React et intégration de Vite pour un flux de développement rapide. (Phu-Hung)
- Sécurisation de l'application avec Keycloak pour l'authentification et la gestion des rôles. (Leo)
- Tests des fonctionnalités et ajustements de l'interface. (Phu-Hung)

3. **Ressources Externes Utilisées**

- Documentation Officielle :

- MongoDB Documentation pour la création de schémas et les opérations CRUD.
- Keycloak Documentation pour la configuration des rôles et l'authentification sécurisée.
- Docker Documentation pour l'orchestration des services.

- Tutoriels et Articles :

- Tutoriels spécifiques pour l'intégration de Docker avec des applications Node.js/React.
- Articles sur les meilleures pratiques d'interopérabilité médicale et les normes MOS/NOS.

- Support Communautaire :

- Stack Overflow et forums MongoDB pour résoudre des problèmes techniques spécifiques, ainsi que les outils d'iA comme ChatGPT et ClaudeiA.

4. **Travail Collaboratif**

Partage de Code : Nous avons utilisé Git et GitHub pour gérer le versionnage du code et effectuer des revues régulières des commits.

5. Implémentation de la Solution

- **Backend (API)** : Le backend est construit avec Node.js et Express. Chaque entité (patients, professionnels, dispositifs médicaux) a son propre schéma

dans MongoDB. J'ai mis en place des routes pour gérer les requêtes des utilisateurs avec vérification des autorisations grâce à Keycloak.

- **Frontend** : L'interface utilisateur est développée avec React et Vite. Des formulaires de saisie permettent aux professionnels de santé de gérer les patients, tandis que les composants affichent les données sous forme de listes ou de graphiques.
- **Docker** : Chaque service (frontend, backend, MongoDB, Keycloak, PostgreSQL) est conteneurisé pour faciliter le déploiement et les tests. Docker Compose orchestre le tout.
- **Sécurité** : Keycloak gère l'authentification et l'autorisation, avec des rôles assignés aux différents utilisateurs (médecins, secrétaires) pour un accès contrôlé aux données.

1) Préparation

Quels sont les objets du MOS dont nous aurons besoin ?

Les principaux objets du Modèle des Objets de Santé (MOS) pour ce cas incluraient :

- **Professionnels de santé** : Les médecins cardiologues et le personnel du secrétariat, avec leurs rôles et les informations de contact.

- **Patient** : Informations de base sur le patient, son historique médical, et son dossier contenant des éléments comme les comptes rendus médicaux.

- **Dispositifs médicaux (DM)** : Inclure le dispositif connecté porté par le patient, avec la capacité de suivre la tension artérielle, le rythme cardiaque, et l'oxymétrie.
- **Service externe de compilation de données** : Bien que hors périmètre, il est pertinent d'inclure une description pour la réception de comptes rendus et d'alertes par email/SMS.

- **Dossier médical** : Partie structurée contenant les comptes rendus et les alertes, accessibles selon les habilitations.

- **Actes médicaux** : Référencer les actes en lien avec le suivi des mesures, comme la prise de tension continue.

Quelles sont les nomenclatures du NOS qui seront utiles ?

Les nomenclatures importantes issues de la Nomenclature des Objets de Santé (NOS) vont inclure :

- **Types d'établissements de santé** : Par exemple, le type de cabinet médical.

- **Actes médicaux et interventions** : Notamment l'identifiant de l'hypertension artérielle (1.2.250.1.213.3.3.11/520), de la prise de tension continue (1.2.250.1.213.3.3.11/934), et d'un bilan radiologique (1.2.250.1.213.3.3.11/1143).

- **Dispositifs médicaux** : Code pour identifier le type de dispositif de surveillance connectée du patient.

- **Rôles et permissions** : Les nomenclatures pour spécifier les rôles (médecins, secrétariat) et les niveaux d'accès.

Quelle matrice d'habilitations pouvons-nous construire (en attendant d'avoir quelque chose de plus rigoureux…) ?

Roles	Dossier administratif	Dossier médical (résume)	Dossier complet	Alertes
Secretariat	Accès complet	Accès aux comptes rendus	Aucun accès	Aucun
Medecin	Accès complet	Accès complet	Accès complet	Accès
Service externe	Aucun	Accès aux données brutes	Accès limité (sous supervision)	Envoi
Patient	Aucun accès	Accès limité (read-only)	Aucun accès	Non

Cette matrice indique les accès par rôle, avec un système qui permettrait au médecin d'avoir un accès complet, tandis que le secrétariat aurait un accès limité aux informations pertinentes pour le suivi administratif.

2) Implémentation

Créer la base de données et les mécanismes de saisie et de visualisation sécurisés associés, ainsi qu'un jeu de données de tests.

Dans MongoDB, chaque type d'entité sera stocké dans une collection dédiée, et chaque document dans ces collections contiendra les informations spécifiques aux patients, professionnels de santé, dispositifs médicaux et comptes rendus.

a) Créer la base de données avec votre SGBD favori.

Nous avons donc implémenté une base de donnée MongoDB avec les objets MOS définis ci-dessus :

- Une table **Patients** pour stocker les informations patient.
- Une table **Professionnels** de santé pour les médecins et secrétaires.
- Une table **Dispositifs Médicaux** pour les DM connectés, avec une colonne pour chaque type de mesure..

- Une table **Comptes rendus** pour stocker les rapports quotidiens générés (la data des devices).

b) Mettre en place un système d'authentification/autorisation mettant en œuvre les règles d'habilitation (par exemple Keycloak).
Pour l'implémentation de Keycloak, nous avons configuré un **realm** nommé `"medical-cabinet"` pour gérer les rôles et les utilisateurs spécifiques à cette application médicale pour rester conforme à la matrice d'habilitation. Voici les principaux points :

- **Rôles** : Nous avons défini trois rôles principaux :
  - `"secretary"` : destiné au personnel de secrétariat médical, avec accès aux données administratives.
  - `"doctor"` : destiné aux cardiologues, avec accès aux données médicales des patients.
  - `"patient"` : pour les utilisateurs patients, avec accès restreint aux informations personnelles.

c) Développer un front qui permettra de d'alimenter et consulter la base de données (gestionnaire de formulaires).

Nous avons conçu pour faciliter l'alimentation et la consultation de la base de données via une interface de gestion de formulaires intuitive. Développé en React avec Vite, il permet aux utilisateurs, en fonction de leur rôle, d'ajouter des patients, de consulter et de mettre à jour leurs informations, et de gérer les professionnels médicaux.

L'application utilise l'interface **Keycloak** pour gérer l'authentification et l'autorisation en fonction des rôles (secrétaire, docteur, patient), garantissant un accès sécurisé. Les formulaires dynamiques permettent de saisir et de modifier les données des patients et des professionnels, connectant le frontend au backend via une API.

Le frontend est structuré pour offrir une navigation fluide, avec des formulaires adaptés aux différents types d'entrées, tout en assurant la communication avec le backend sur le port 5000 grâce à un proxy dans la configuration de Vite.

Un schéma de l'implémentation est disponible sur la page du repo :
https://github.com/Heldeee/inta_td

Si le groupe le souhaite, il est possible de mutualiser le front, à condition que les contributions de chacun soient clairement décrites. Vous pouvez utiliser la technologie de votre choix, de préférence adaptée au prototypage.

L’appréciation dépendra :

- De la compréhension des problématiques
- De la qualité des explications
- De la qualité de l’implémentation
- De la manière d’exploiter les ressources disponibles