LabXpert - l'API REST via Spring Boot (Part 1)

Le laboratoire médical TechLab a besoin d'une application de gestion, LabXpert, pour optimiser ses opérations en améliorant l'efficacité et la précision dans le traitement des analyses médicales.

Ce référentiel contient des classes Java qui représentent les entités du système LabXpert, un laboratoire d'analyse médicale. Ci-dessous, vous trouverez une brève description de chaque classe fournie :

Analyse
La classe Analyse représente une analyse médicale effectuée par le laboratoire. Elle contient des informations telles que le technicien responsable, l'échantillon associé, le statut du résultat, le type d'analyse, les dates de début et de fin, ainsi que des commentaires.

Echontillon
La classe Echontillon représente un échantillon médical soumis au laboratoire pour analyse. Elle comprend des détails tels que le patient associé, la date de prélèvement, le statut de l'échantillon, et la liste des analyses effectuées.

Fournisseur
La classe Fournisseur représente un fournisseur de réactifs médicaux. Elle contient des informations telles que le nom complet du fournisseur, le nom de la société, et la liste des réactifs fournis.

Patient
La classe Patient hérite de la classe abstraite Person et représente un patient du laboratoire. Elle inclut des détails spécifiques au patient tels que l'âge et la liste des échantillons associés.

Person
La classe abstraite Person représente une entité générique avec des informations personnelles de base, telles que le nom, le prénom, l'adresse, le numéro de téléphone, la ville, le sexe et la date de naissance.

Planification
La classe Planification représente la planification d'une analyse. Elle contient des détails tels que l'analyse associée, le technicien en charge, ainsi que les dates de début et de fin de la planification.

RapportStatistique
La classe RapportStatistique représente un rapport statistique généré par le laboratoire. Elle inclut des informations telles que le type de rapport, la période statistique, les données statistiques et un graphique associé.

Reactif
La classe Reactif représente un réactif médical utilisé dans les analyses. Elle comprend des informations telles que le nom, la description, la quantité en stock, la date d'expiration, et le fournisseur associé.

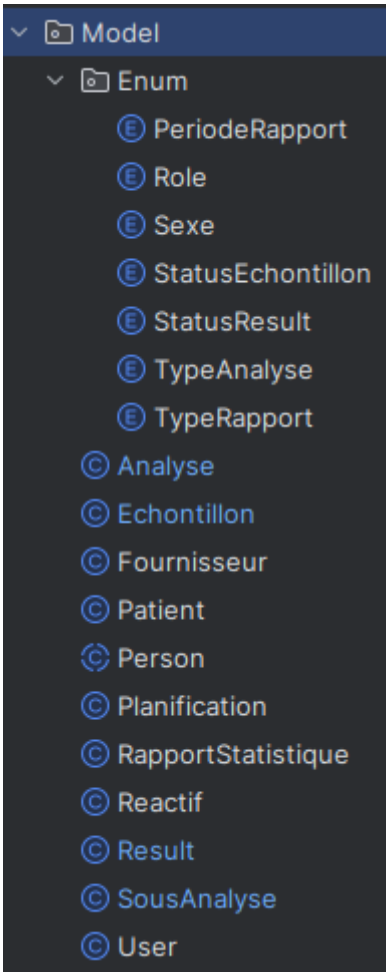
Result
La classe Result représente le résultat d'une sous-analyse. Elle inclut la valeur du résultat, l'unité de mesure, et la sous-analyse associée.

SousAnalyse
La classe SousAnalyse représente une sous-analyse effectuée dans le cadre d'une analyse principale. Elle contient des informations telles que le titre, les valeurs normales, l'analyse associée, le réactif utilisé, et le statut du résultat.

User
La classe User représente un utilisateur du système LabXpert. Elle hérite de la classe Person et inclut des détails spécifiques à l'utilisateur tels que l'e-mail, le mot de passe, le rôle, ainsi que les listes d'analyses et de planifications associées.

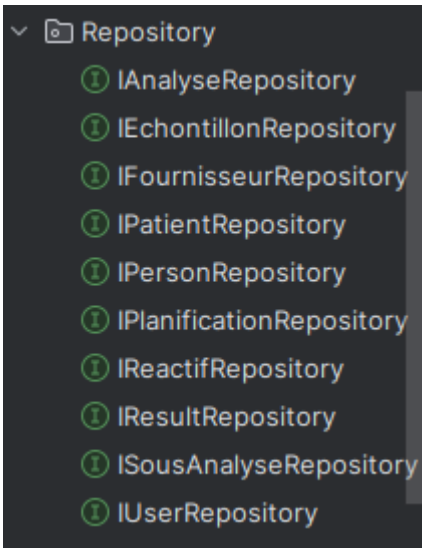
Ces classes fournissent une structure de base pour modéliser les entités du laboratoire d'analyse médicale dans le système LabXpert.

Package des Models



Package des Repository

Ces interfaces repository pour traiter les opérations classiques dans commandeLinner méthode



Package des Services

Ces méthodes dans les classes service implémente les méthodes classiques crud operations pour tester ces fonctionnalités dans commandeLinner

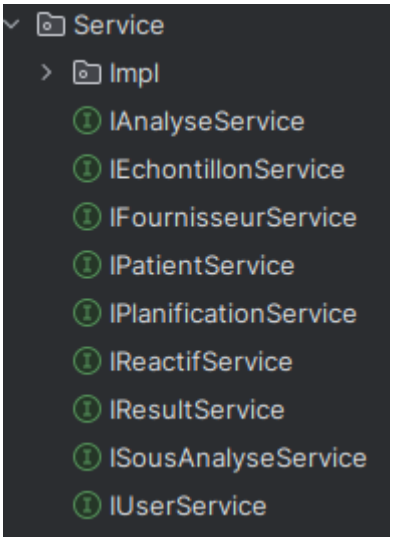


Diagramme de classe

Ce diagramme représente la conception et la structure statique d'un système en montrant les classes du LabXpert, les attributs de ces classes, les relations entre les classes, et les opérations ou méthodes que les classes peuvent exécuter

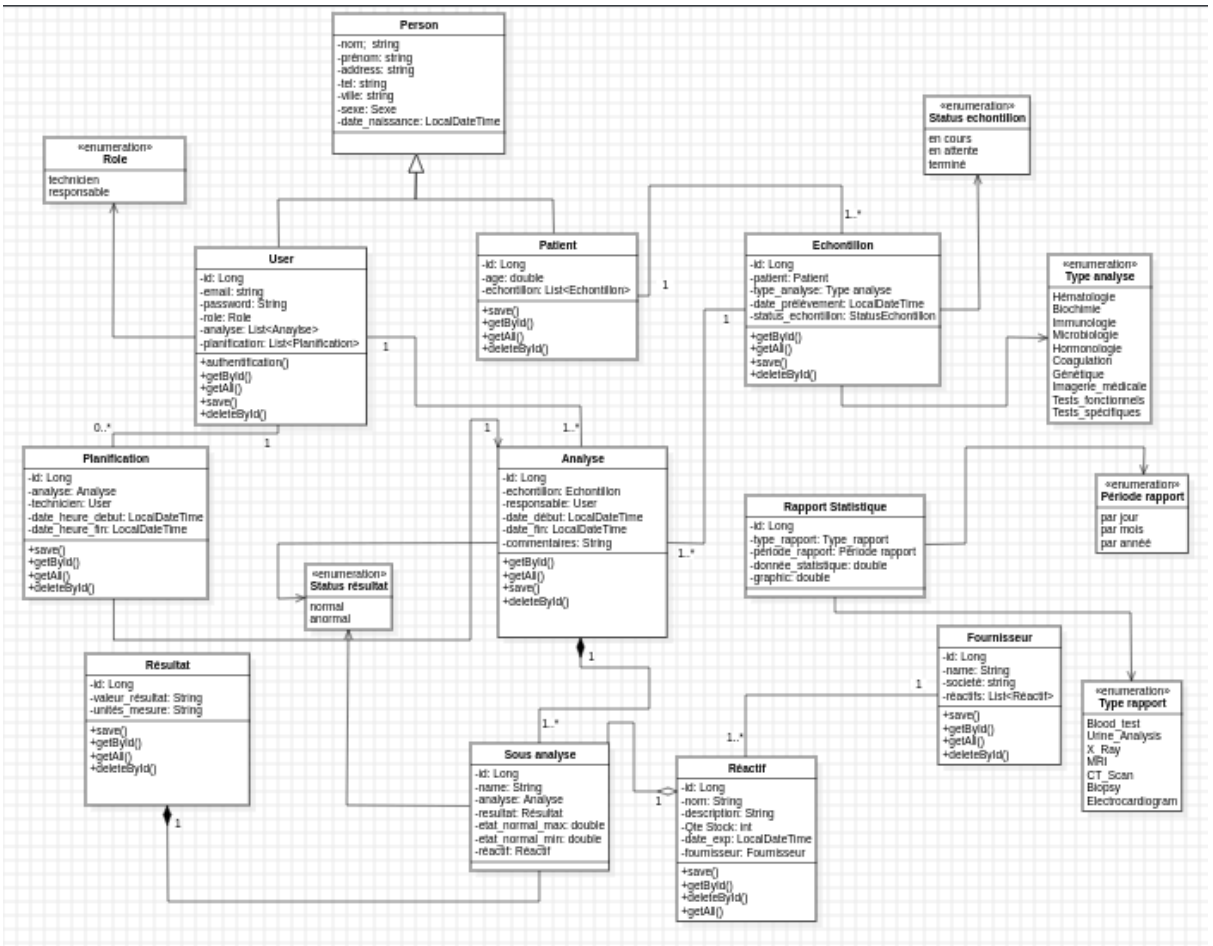
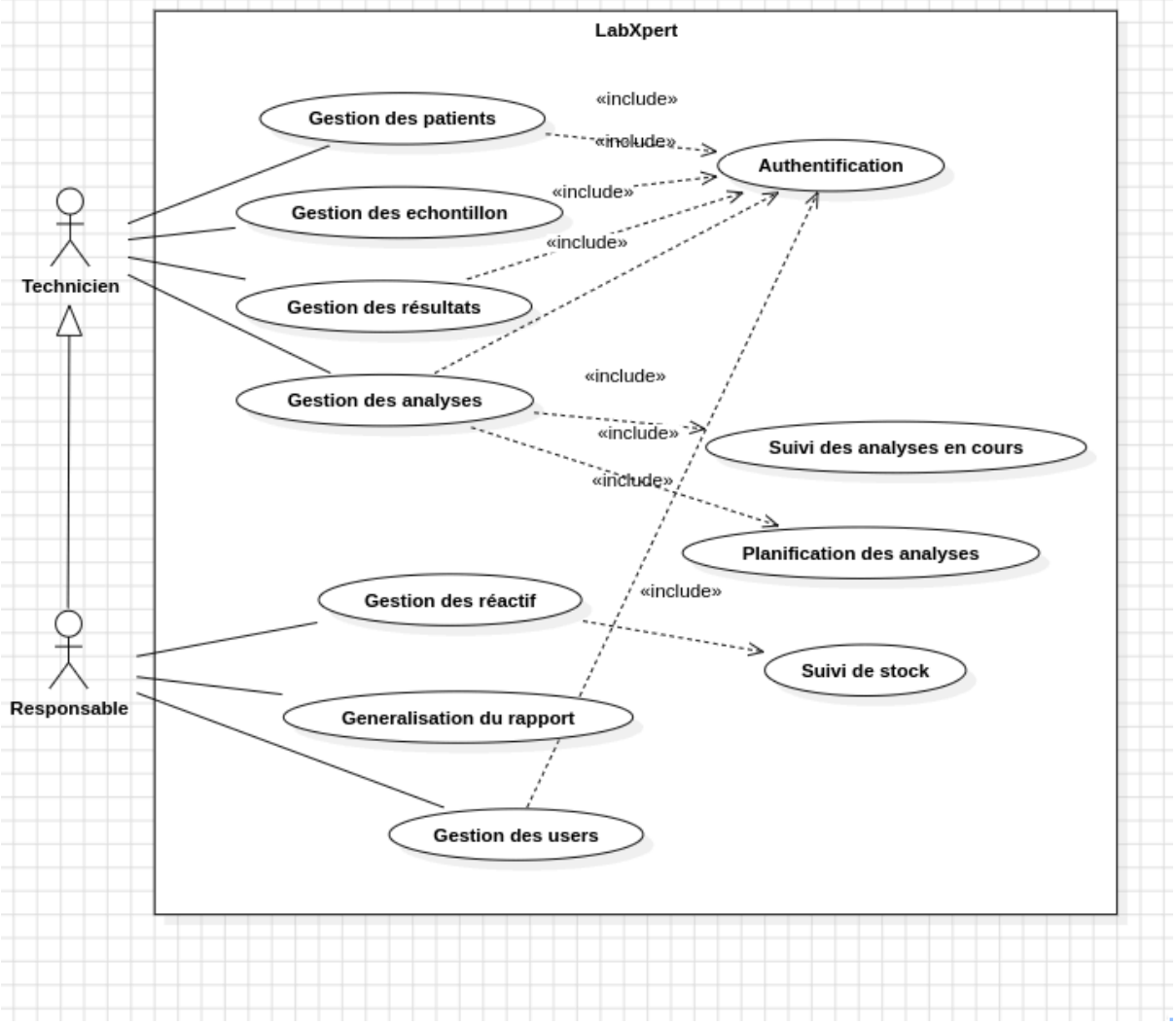


Diagramme de cas d’utilisation

Ce diagramme représente les différentes interactions possibles entre les acteurs externes (technicien et responsable) et un système logiciel. Il met l'accent sur les fonctionnalités ou les services que le système offre, en se concentrant sur les interactions entre le système et ses utilisateurs.



Main Test crud operation

Fournisseur

```java
/**
 *      fournisseur crud
 */


Fournisseur fournisseur = new Fournisseur();
fournisseur.setNameComplet("marouane mouslih");
fournisseur.setSocieteName("atlas");
iFournisseurService.add(fournisseur);

Fournisseur fournisseur1 = iFournisseurService.getById(3L);

System.out.println(fournisseur.getNameComplet() + "," + fournisseur.getSocieteName());

fournisseur1.setSocieteName("TIMAR");
iFournisseurService.update(fournisseur1);
```

##Réactif

```java
/**
 * Reactif crud
 */

Reactif reactif = new Reactif();
Fournisseur fournisseur = iFournisseurService.getById(3L);
reactif.setNom("nom reactif 3");
reactif.setDescription("description 3");
reactif.setFournisseur(fournisseur);
reactif.setQuantity_stock(18);
reactif.setDate_exp(LocalDate.now());
iReactifService.add(reactif);

iFournisseurService.delete( id: 2L);
```

##Patient

```
Patient patient = new Patient();
patient.setNom("marouane");
patient.setPrenom("mouslih");
patient.setTel("0630011276");
patient.setVille("casablanca");
patient.setAddress("address 1");
patient.setSexe(Sexe.MALE);
patient.setDate_naissance(LocalDate.now());
patient.setAge(22);
iPatientService.add(patient);

Patient patient = iPatientService.getById(1L);
patient.setAddress("address 111");
iPatientService.update(patient);

iPatientService.delete( id: 2L);

List<Patient> patients = iPatientService.getAll();

for (Patient p : patients)
{
    System.out.println("name : " + p.getNom());
    System.out.println("age: " + p.getAge());
```

##User

```java
/**
 * User crud
 */

User user = new User();
user.setNom("abdo");
user.setPrenom("ayoub");
user.setTel("0630011276");
user.setVille("casablanca");
user.setAddress("address 1");
user.setSexe(Sexe.MALE);
user.setDate_naissance(LocalDate.now());
user.setRole(Role.Responsable);
user.setEmail("maromouslih@gmail.com");
user.setPassword("111111");
iUserService.add(user);

User user = iUserService.getById(4L);
user.setNom("oussama");
//System.out.println(user);
iUserService.update(user);

iUserService.delete( id: 5L);
```

##Réactif

```
/**
 * Reactif crud
 */

Reactif reactif = new Reactif();
Fournisseur fournisseur = iFournisseurService.getById(3L);
reactif.setNom("nom reactif 3");
reactif.setDescription("description 3");
reactif.setFournisseur(fournisseur);
reactif.setQuantity_stock(18);
reactif.setDate_exp(LocalDate.now());
iReactifService.add(reactif);

iFournisseurService.delete( id: 2L);
```

##Echontillon

```
/**
 * Echontillon crud
 */

Patient patient = iPatientService.getById(1L);
Echontillon echontillon = new Echontillon();
echontillon.setPatient(patient);
echontillon.setStatusEchontillon(StatusEchontillon.en_attend);
echontillon.setDate_p(LocalDate.of( year: 2022, month: 2, dayOfMonth: 12));
iEchontillonService.add(echontillon);

Echontillon echontillon = iEchontillonService.getById(2L);
echontillon.setStatusEchontillon(StatusEchontillon.en_cours);
iEchontillonService.update(echontillon);

iEchontillonService.delete( id: 2L);
```

##Analyse

```java
/**
 * Analyse crud
 */

Echontillon echontillon = iEchontillonService.getById(1L);
User user = iUserService.getById(4L);
Analyse analyse = new Analyse();
analyse.setCommantaires("commentaire 2");
analyse.setEchontillon(echontillon);
analyse.setStatusResult(StatusResult.normal);
analyse.setDate_fin(LocalDate.of( year: 2022, month: 1, dayOfMonth: 12));
analyse.setTechnicienResponsable(user);
analyse.setDate_debut(LocalDate.of( year: 2021, month: 8, dayOfMonth: 16));
analyse.setTypeAnalyse(TypeAnalyse.Hormonologie);
iAnalyseService.add(analyse);
Analyse analyse = iAnalyseService.getById(1L);
analyse.setCommantaires("commantaire 11");
iAnalyseService.update(analyse);

iAnalyseService.delete( id: 2L);
```

##Planification

```java
/**
 * Planification crud
 */

Analyse analyse = iAnalyseService.getById(1L);
User user = iUserService.getById(4L);
Planification planification = new Planification();
planification.setAnalyse(analyse);
planification.setTechnicien(user);
planification.setDate_debut(LocalDate.of( year: 2022, month: 1, dayOfMonth: 22));
planification.setDate_fin(LocalDate.now());
iPlanificationService.add(planification);

Planification planification = iPlanificationService.getById(1L);
planification.setDate_debut(LocalDate.of( year: 2022, month: 1, dayOfMonth: 22));
iPlanificationService.update(planification);

iPlanificationService.delete( id: 2L);
```

##SousAnalyse

```
/**
 * Sous analyse crud
 */

SousAnalyse sousAnalyse = new SousAnalyse();
Analyse analyse = iAnalyseService.getById(1L);
Reactif reactif = iReactifService.getById(4L);
sousAnalyse.setTitle("sous analyse 1");
sousAnalyse.setEtat_normal_max(12.3);
sousAnalyse.setEtat_normal_min(8.03);
sousAnalyse.setAnalyse(analyse);
sousAnalyse.setReactif(reactif);
sousAnalyse.setStatusResult(StatusResult.anormal);
iSousAnalyseService.add(sousAnalyse);

SousAnalyse sousAnalyse = iSousAnalyseService.getById(1L);
sousAnalyse.setEtat_normal_max(14.1);
sousAnalyse.setEtat_normal_min(9.80);
iSousAnalyseService.update(sousAnalyse);

iSousAnalyseService.delete( id: 2L);
```

##Resultat

```
/**
 *  Result crud
 */

Result result = new Result();
SousAnalyse sousAnalyse = iSousAnalyseService.getById(1L);
result.setValeur_result(13);
result.setUnite_mesure("Up");
result.setSousAnalyse(sousAnalyse);
iResultService.add(result);

Result result = iResultService.getById(2L);
result.setUnite_mesure("WAT");
iResultService.update(result);

iResultService.delete( id: 2L);
```