

Documentation du Projet

LabXpert

Introduction:

Le laboratoire médical TechLab aspire à étendre ses capacités opérationnelles en développant des services et des API pour son système de gestion, LabXpert. L'objectif principal est de faciliter l'intégration avec d'autres systèmes, d'assurer un accès sécurisé aux données et de soutenir les pratiques de développement modernes.

Fonctionnalités Attendues

1. Services d'Échantillons

Enregistrement d'Échantillons : Service permettant l'enregistrement de nouveaux échantillons avec les détails du patient, le type d'analyse, et la date de prélèvement.

Récupération de Détails : Service pour récupérer les détails d'un échantillon spécifique.

2. Services d'Analyses

Suivi de l'Avancement : Service permettant de suivre l'état d'avancement des analyses en cours en temps réel.

Lancement d'une Nouvelle Analyse : Service pour lancer une nouvelle analyse avec les détails associés.

Récupération des Résultats : Service pour récupérer les résultats d'une analyse spécifique.

3. Services de Gestion des Résultats

Consignation des Résultats : Service pour consigner de manière systématique les résultats des analyses.

Partage avec les Professionnels de la Santé : Service pour partager les résultats avec les professionnels de la santé.

4. Services de Gestion des Patients

Gestion des Informations : Service pour gérer les informations relatives aux patients.

Historique des Analyses : Service pour récupérer l'historique des analyses d'un patient spécifique.

5. Services d'Inventaire des Réactifs

Suivi des Stocks : Service pour suivre les stocks de réactifs.

Notification de Réactifs Bas : Service pour notifier lorsque les réactifs sont bas.

6. Services de Gestion des Utilisateurs

Administration des Utilisateurs : Service d'administration pour gérer les droits d'accès et les informations des utilisateurs.

7. Services de Planification des Analyses

Planification des Analyses : Service pour planifier les analyses en fonction de la charge de travail.

8. Services de Rapports Statistiques

Génération de Rapports : Service pour générer des rapports statistiques sur les performances du laboratoire.

| | | | | | | |
|---|-----------|--------------------|-----|-------|---------|--|
|  | Nom : | mika | | | | |
| | Prenom : | momo | | | | |
| | Numero : | 0123456789 | | | | |
| | Adresse : | Adresse du patient | | | | |
| DMO | | | | | | |
| CHIMIE | | | | | | |
| hemolise | momo | 2.0 | 0.5 | mg/dL | NORMAL | |
| normalise | momo | 2.0 | 0.5 | mg/dL | NORMAL | |
| volumeGlobal | momo | 2.0 | 0.5 | mg/dL | ANORMAL | |
| CONC | momo | 2.0 | 0.5 | mg/dL | NORMAL | |
| BIOCHIMIE | | | | | | |
| CMIP | momo | 2.0 | 0.5 | mg/dL | NORMAL | |
| homolise | momo | 2.0 | 0.5 | mg/dL | NORMAL | |
| domitos | momo | 2.0 | 0.5 | mg/dL | NORMAL | |
| VIMIS | momo | 2.0 | 0.5 | mg/dL | ANORMAL | |

Exemple de report generer par l'application

Intégration des Technologies:

Utilisation de Spring Boot pour le développement des services.

Intégration de Swagger pour la documentation automatique de l'API.

Intégration avec Jenkins pour la mise en œuvre du CI/CD.

Stack Technique de l'Application "LabXpert":

Langage de Programmation : Java

Backend : Spring Boot

API RESTful

Gestion de Dépendances : Apache Maven

Base de Données : PostgreSQL

Serveur d'Application : Apache Tomcat

Testing : JUnit & Mockito

CI/CD : Jenkins

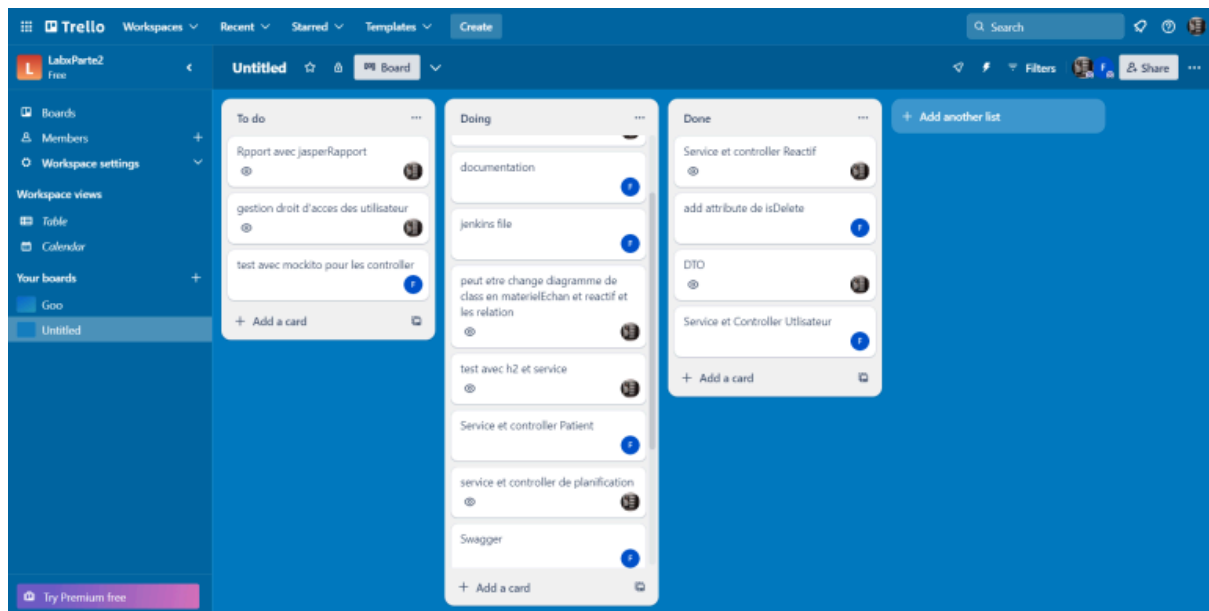
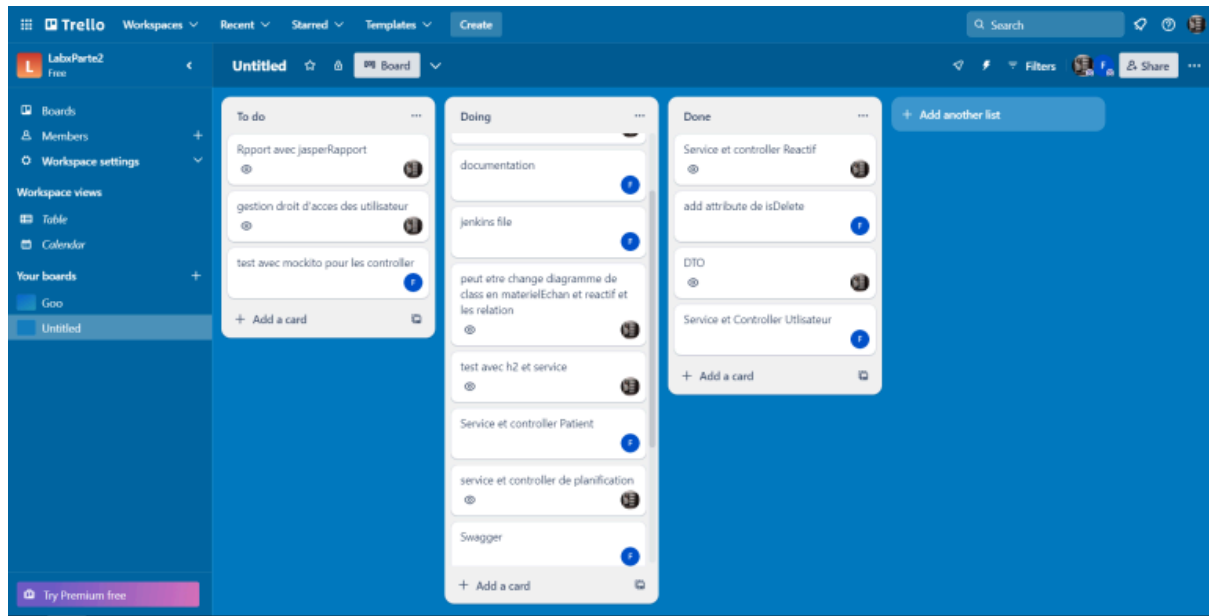
Gestion des Tâches : Outil de votre choix

Système de Gestion de Version : Git et Github

Documentation de l'API : Swagger

Gestion des Tâches avec Trello

Pour faciliter la gestion des tâches et la collaboration, nous utilisons Trello. Vous pouvez accéder à notre tableau Trello ici pour suivre l'avancement des différentes tâches assignées à chaque membre de l'équipe.



Trello

Modélisation:
Diagramme de cas d'utilisation

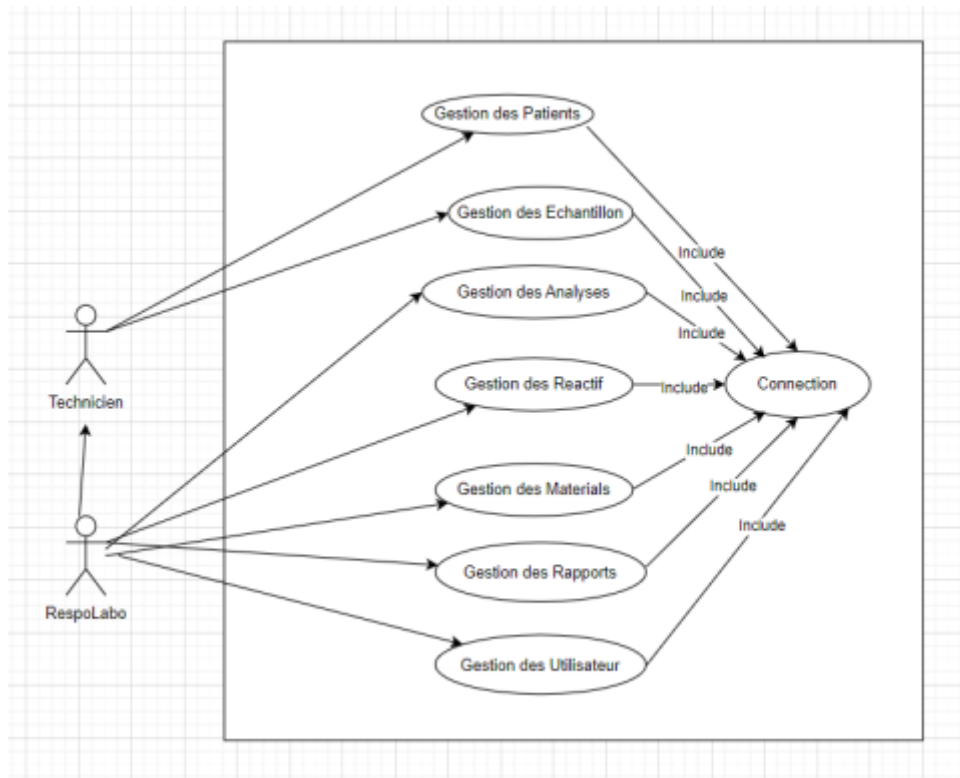


Diagramme de cas d'utilisation

Diagramme de classes:

Les tableaux suivants expliquent les relations entre les classes et détaillent leurs attributs ainsi que leurs méthodes .

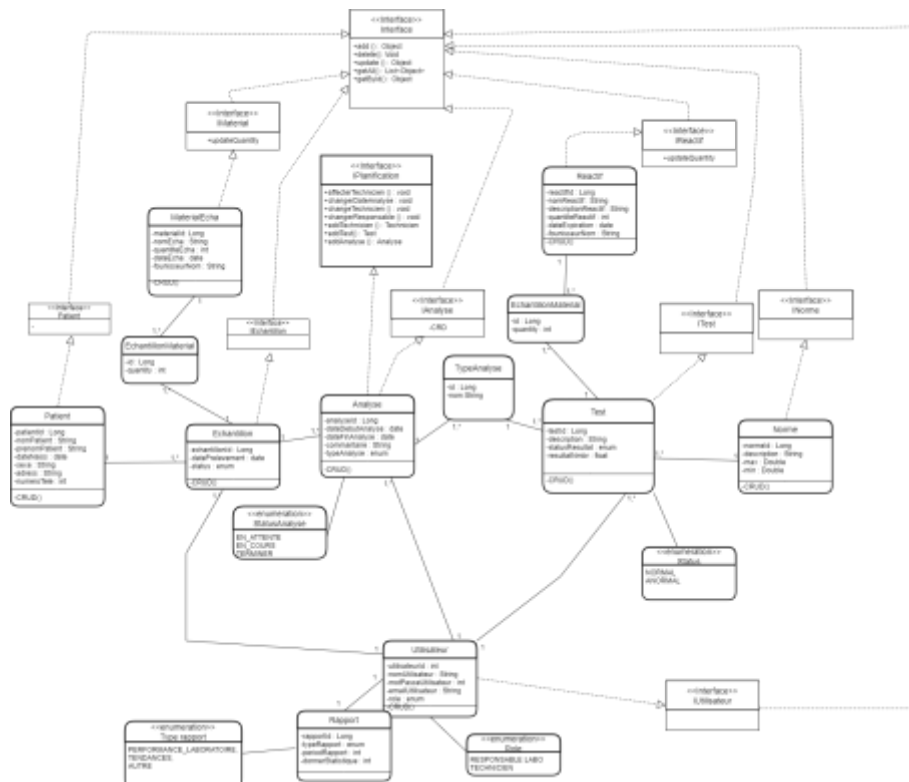
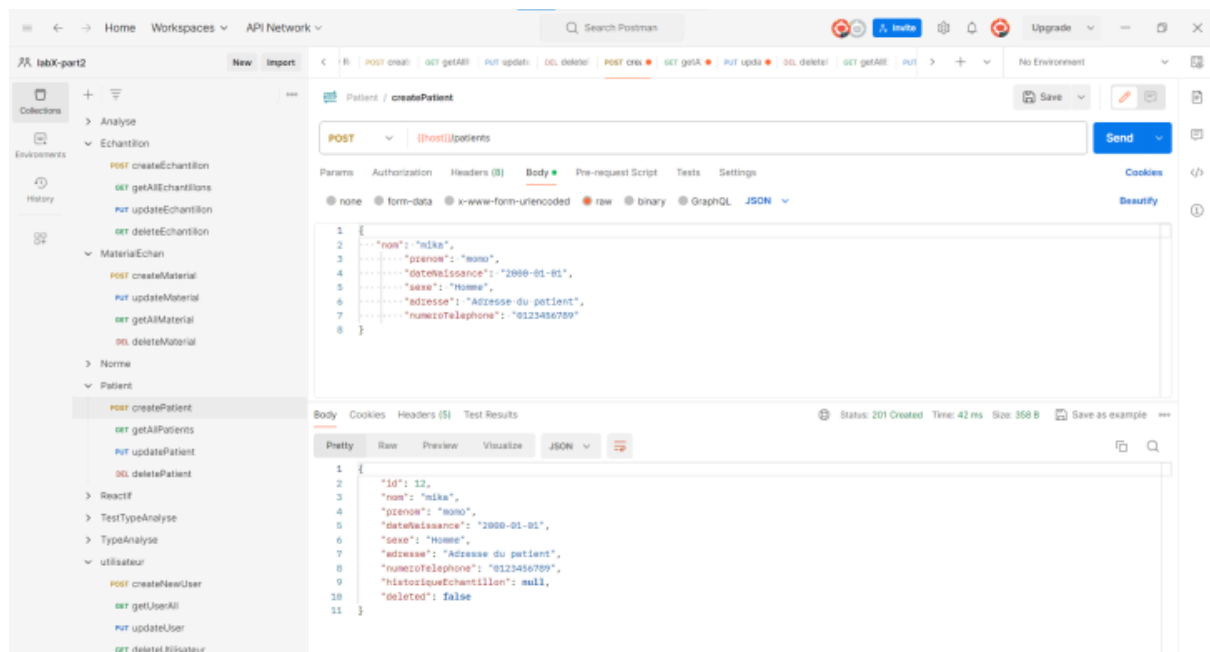


Diagramme de cas d'utilisation

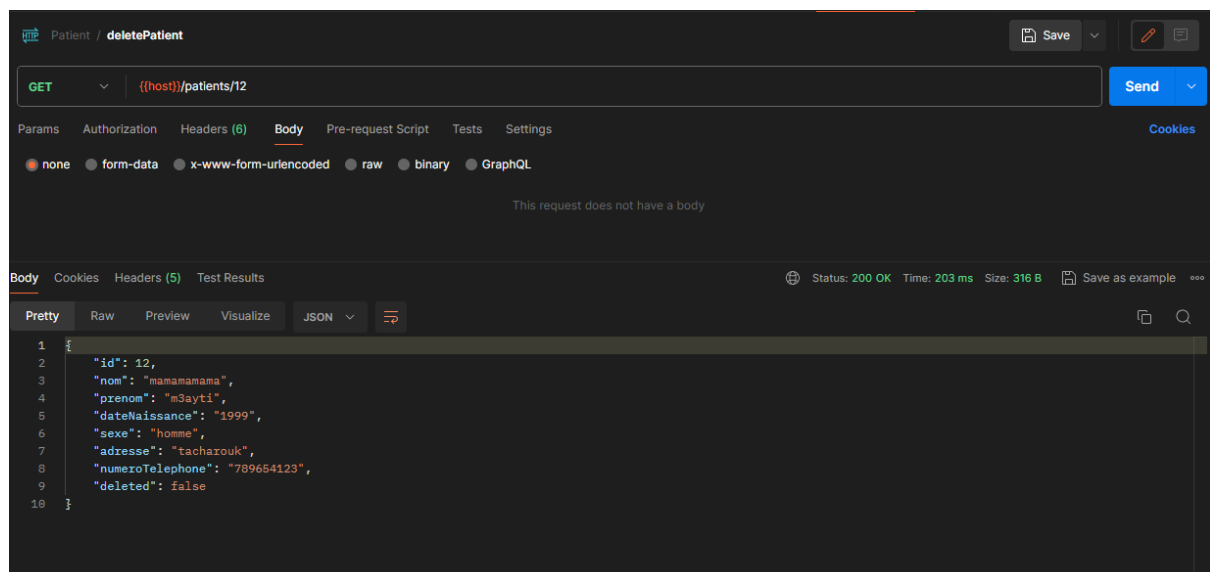
Test avec Postman:

Pour tester les différents services, nous utilisons Postman, un outil puissant pour le développement et le test d'API. Vous pouvez accéder à notre collection Postman ici pour explorer et tester les différentes requêtes.

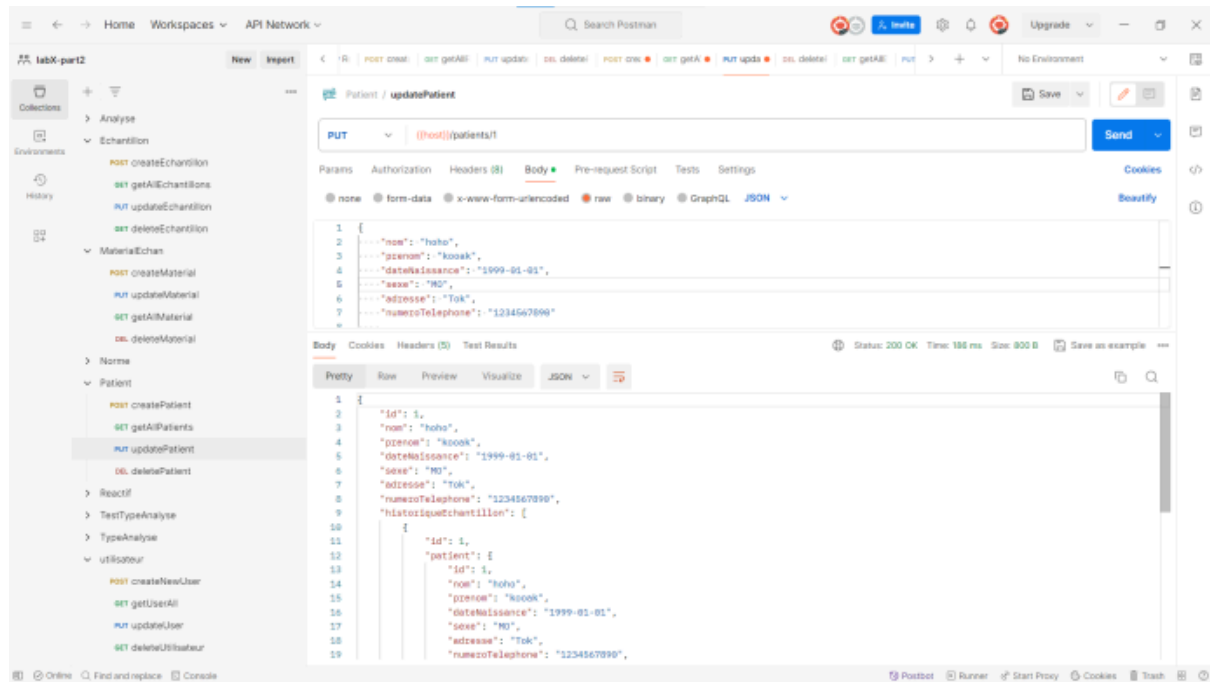
Enregistrement d'un Nouveau Patient (POST)



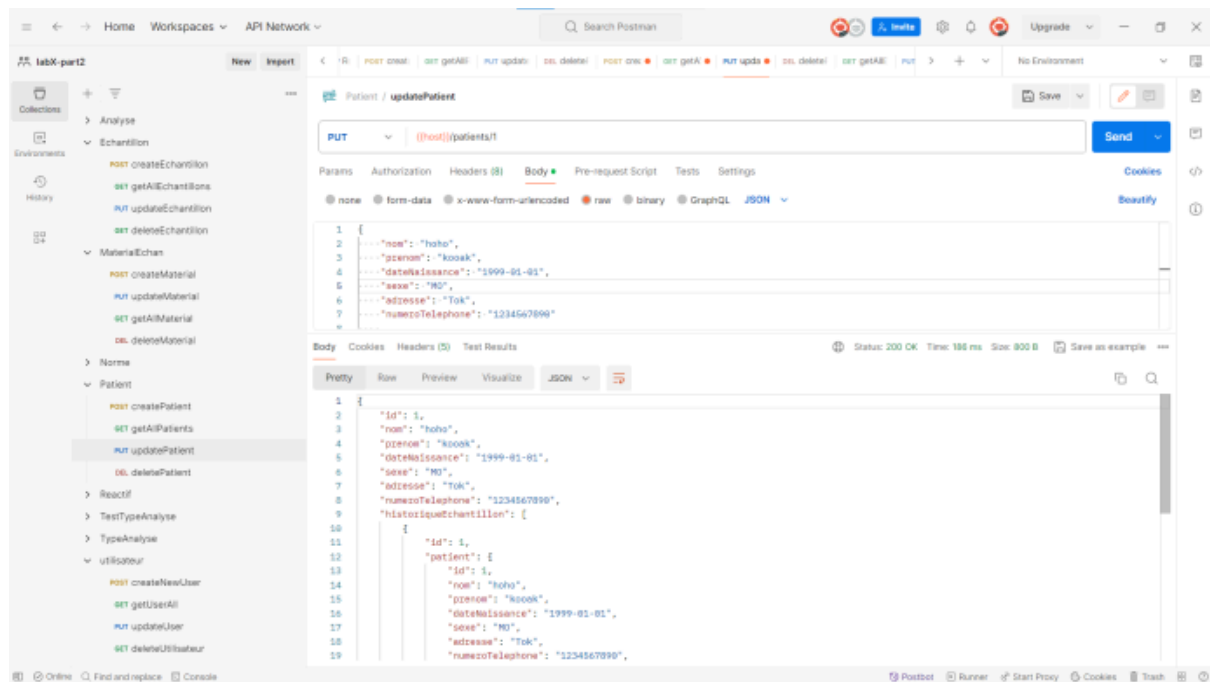
Récupération des Détails d'un Patient (GET)



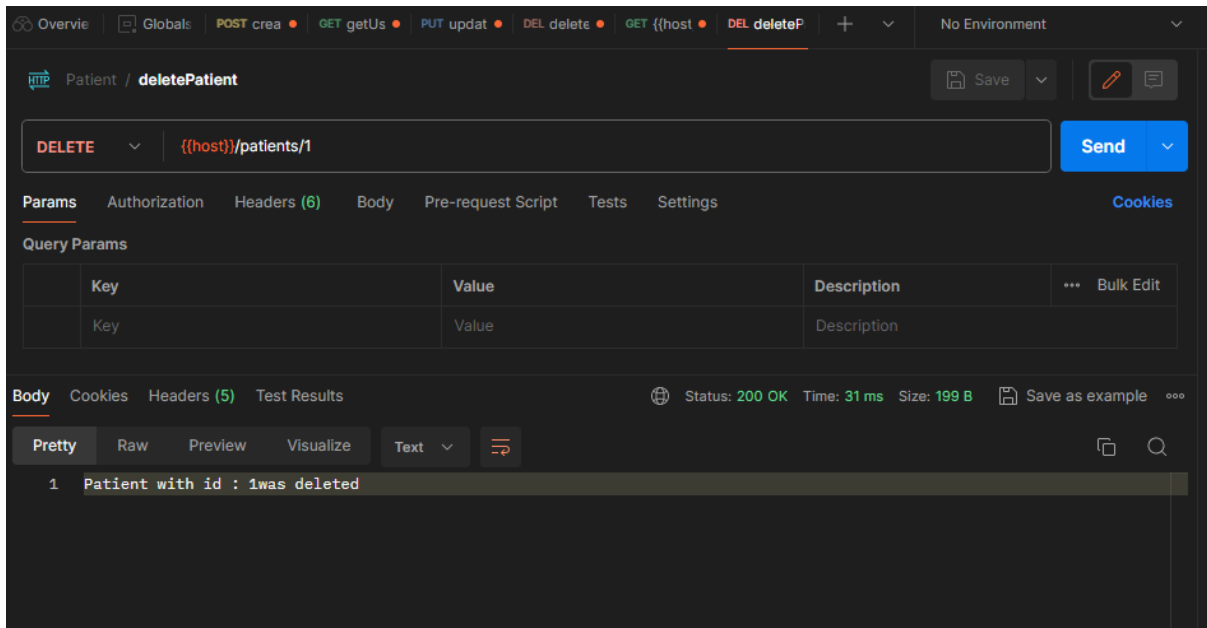
Mise à Jour des Informations d'un Patient (PUT)



Récupération de la Liste des Patients (GET)

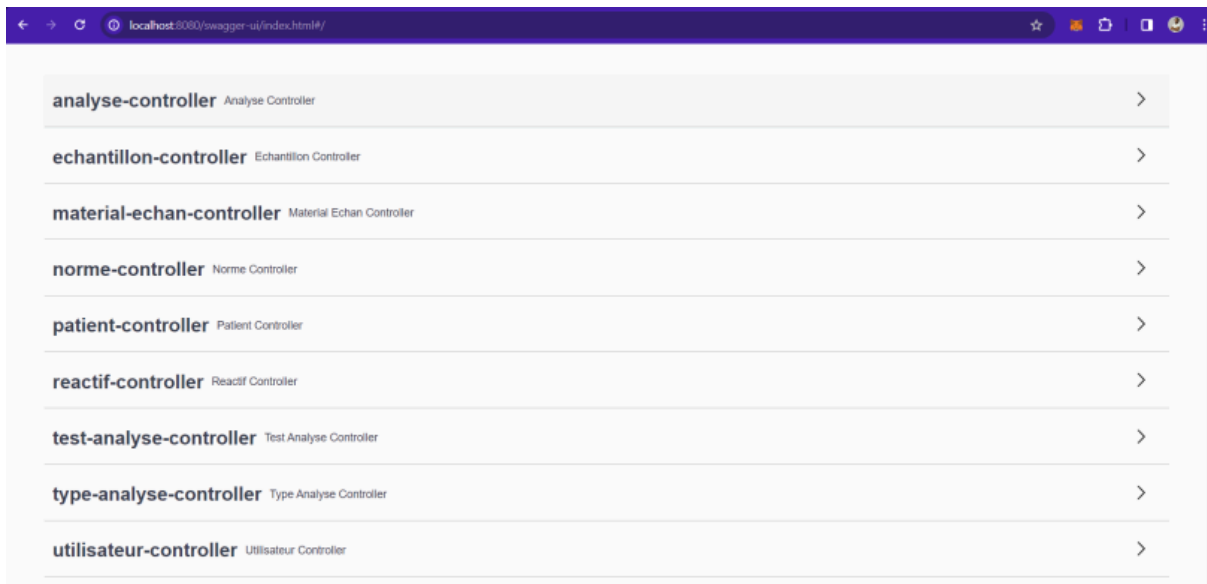


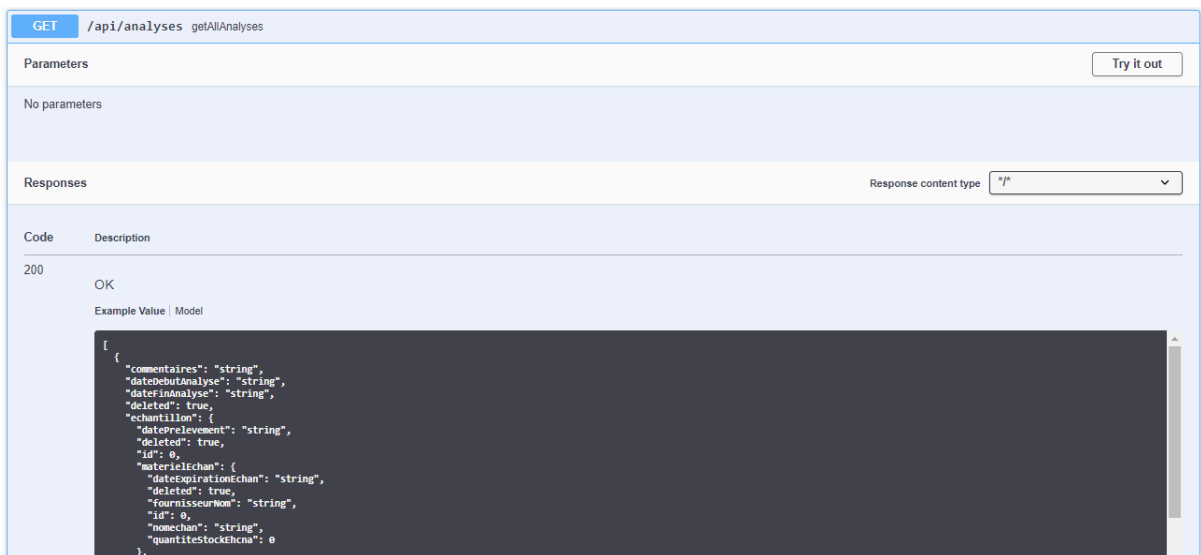
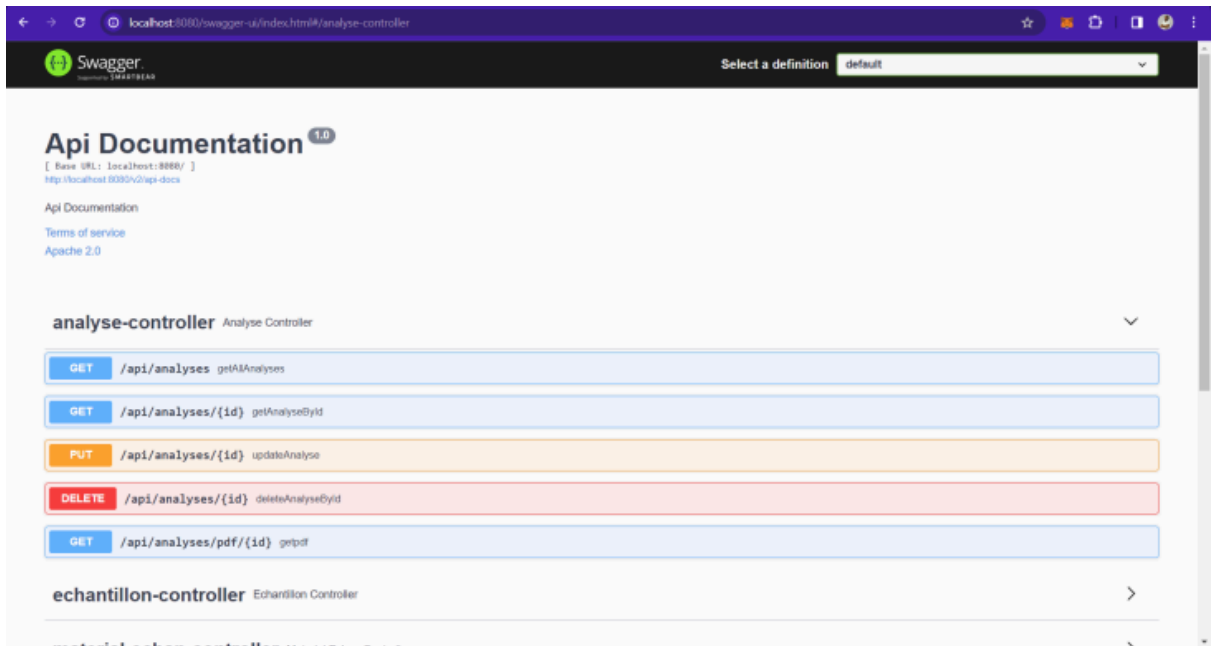
Suppression d'un Patient (DELETE)



Documentation Swagger

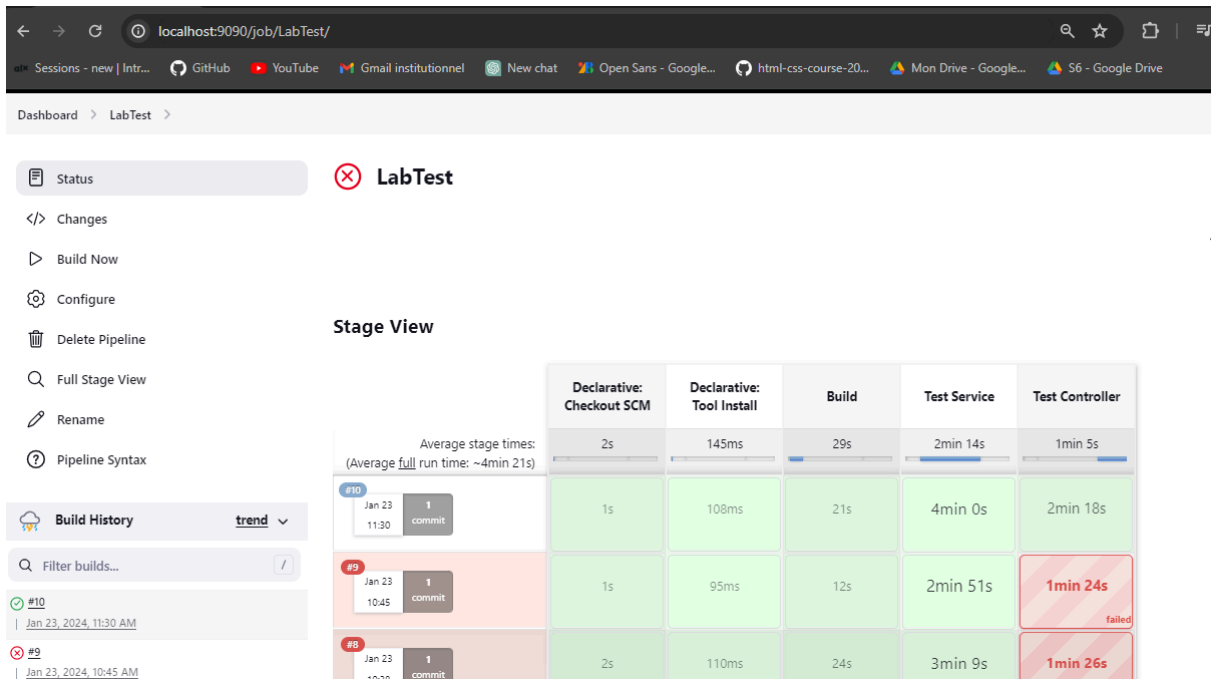
L'intégration de Swagger offre une documentation de l'API claire et interactive. La documentation Swagger est accessible via l'URL suivante après le déploiement de l'application :





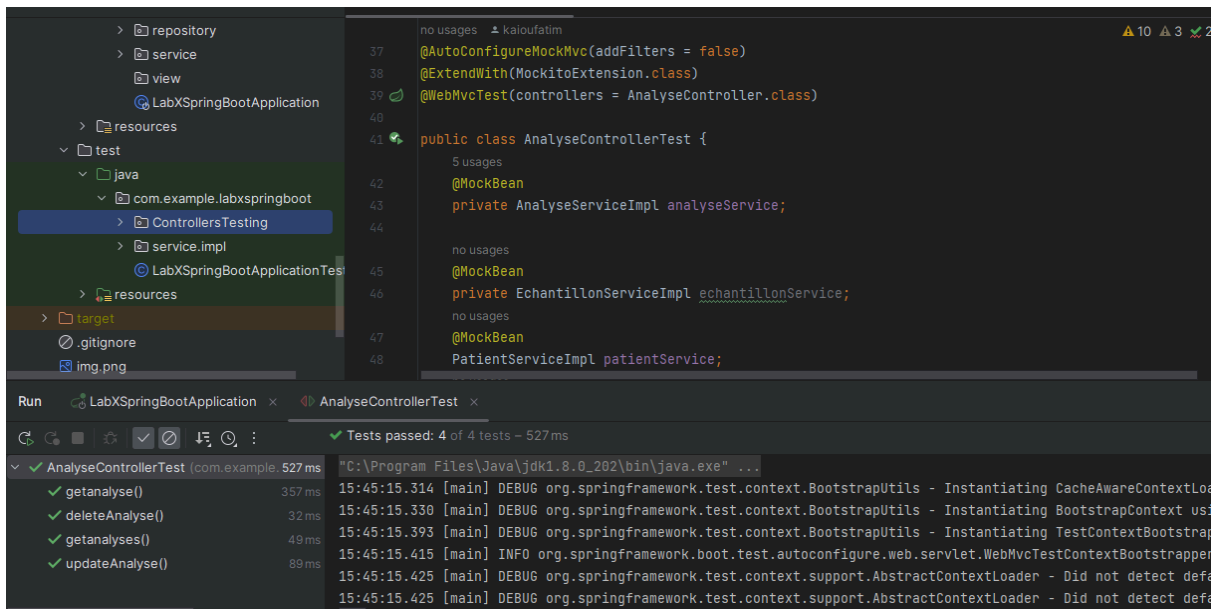
Intégration avec Jenkins

- Configuration des scripts pour l'intégration continue.
- Captures d'écran du pipeline Jenkins.



Jenkinsfile

Contrôleurs Testes :



AnalyseControllerTest

The screenshot shows an IDE with the project 'LabXSpringBootApplication'. The 'test' directory is expanded, showing 'com.example.labxspringboot' > 'ControllersTesting'. The 'EchantillonControllerTest' class is selected. The code for this class is displayed on the right, showing annotations like `@WebMvcTest`, `@AutoConfigureMockMvc`, and `@ExtendWith(MockitoExtension.class)`. The test class contains several methods: `UpdateEchantillonTest()`, `getEchantillonTest()`, `getAllEchantillons()`, `createEchantillonTest()`, and `DeleteEchantillonTest()`. The bottom panel shows the test results for 'EchantillonControllerTest', indicating that all 5 tests passed successfully within 520ms. The console output shows various Spring Framework debug messages.

```
50 @WebMvcTest(controllers = EchantillonController.class)
51 @AutoConfigureMockMvc(addFilters = false)
52 @ExtendWith(MockitoExtension.class)
53
54 public class EchantillonControllerTest {
55     5 usages
56     @Autowired
57     private MockMvc mockMvc;
58
59     5 usages
60     @MockBean
61     private EchantillonServiceImpl echantillonService;
62     no usages
63     @MockBean
64     private PatientServiceImpl patientService;
```

Run LabXSpringBootApplication x EchantillonControllerTest x

Tests passed: 5 of 5 tests - 520ms

EchantillonControllerTest (com.examp 520ms)

- ✓ UpdateEchantillonTest() 398 ms
- ✓ getEchantillonTest() 24 ms
- ✓ getAllEchantillons() 29 ms
- ✓ createEchantillonTest() 53 ms
- ✓ DeleteEchantillonTest() 16 ms

15:47:26.720 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoader
15:47:26.737 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using
15:47:26.799 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper
15:47:26.816 [main] INFO org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTestContextBootstrapper -
15:47:26.822 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
15:47:26.822 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default

EchantillonControllerTest

The screenshot shows the same IDE with the project 'LabXSpringBootApplication'. The 'test' directory is expanded, showing 'com.example.labxspringboot' > 'ControllersTesting'. The 'UtilisateurControllerTest' class is selected. The code for this class is displayed on the right, showing annotations like `@WebMvcTest`, `@AutoConfigureMockMvc`, and `@ExtendWith(MockitoExtension.class)`. The test class contains several methods: `DeleteUtilisateurTest()`, `getUtilisateurTest()`, `createUtilisateurTest()`, `UpdateUtilisateurDtoTest()`, and `GetAllUsers()`. The bottom panel shows the test results for 'UtilisateurControllerTest', indicating that all 5 tests passed successfully within 356ms. The console output shows various Spring Framework debug messages.

```
40 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
41
42 no usages kaoufatim
43 @WebMvcTest(controllers = UtilisateurController.class)
44 @AutoConfigureMockMvc(addFilters = false)
45 @ExtendWith(MockitoExtension.class)
46 public class UtilisateurControllerTest {
47     5 usages
48     @Autowired
49     private MockMvc mockMvc;
50
51     5 usages
52     @MockBean
53     private UtilisateurServiceImpl utilisateurService;
54
55     3 usages
56     @MockBean
```

Run LabXSpringBootApplication x UtilisateurControllerTest x

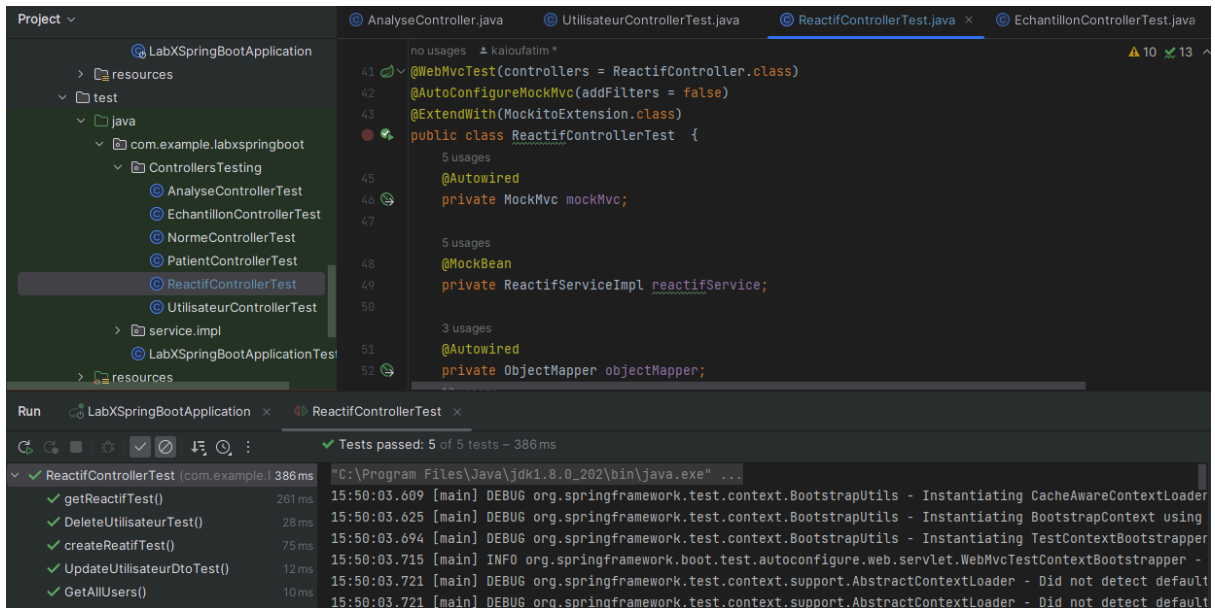
Tests passed: 5 of 5 tests - 356ms

UtilisateurControllerTest (com.examp 356ms)

- ✓ DeleteUtilisateurTest() 185 ms
- ✓ getUtilisateurTest() 85 ms
- ✓ createUtilisateurTest() 50 ms
- ✓ UpdateUtilisateurDtoTest() 10 ms
- ✓ GetAllUsers() 26 ms

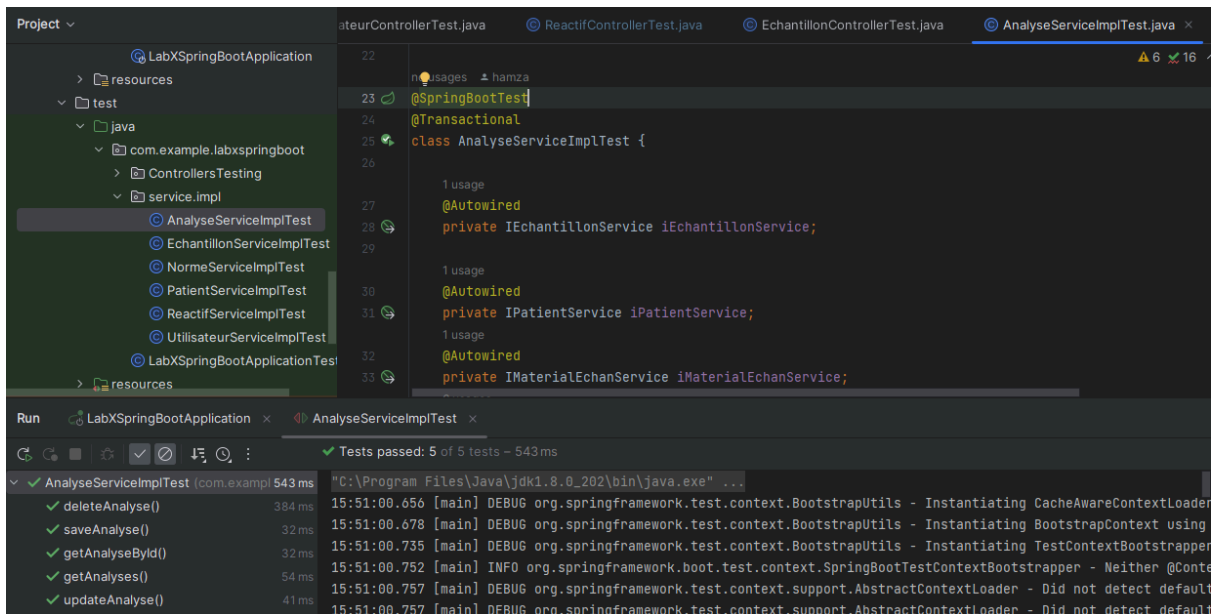
15:48:08.507 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoader
15:48:08.531 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using
15:48:08.590 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper
15:48:08.608 [main] INFO org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTestContextBootstrapper -
15:48:08.614 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
15:48:08.615 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default

UtilisateurControllerTest



ReactifControllerTest

Services Testes :



AnalyseServiceTest

The screenshot shows an IDE with the project 'LabXSpringBootApplication' open. The 'test' directory is expanded, showing the 'service.impl' package. The 'EchantillonServiceImplTest' class is selected. The code editor shows the following Java code:

```
30 no usages hamza
31 @SpringBootTest
32 @Transactional
33 class EchantillonServiceImplTest {
34     11 usages
35     @Autowired
36     private IEchantillonService iEchantillonService;
37     1 usage
38     @Autowired
39     private IPatientService iPatientService;
40     1 usage
41     @Autowired
42     private IMaterialEchanService iMaterialEchanService;
43 }
```

The Run tab shows the test results for 'EchantillonServiceImplTest' (com.example.labxspringboot) with 5 tests passed in 446 ms:

- ✓ getEchantillonById() 304 ms
- ✓ updateEchantillon() 33 ms
- ✓ getAllEchantillons() 31 ms
- ✓ delEchantillon() 48 ms
- ✓ addEchantillon() 30 ms

The console output shows the following logs:

```
15:51:38.831 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoader
15:51:38.859 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using
15:51:38.917 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper
15:51:38.934 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @Conte
15:51:38.941 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
15:51:38.941 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
```

EchantillonServiceTest

The screenshot shows an IDE with the project 'LabXSpringBootApplication' open. The 'test' directory is expanded, showing the 'service.impl' package. The 'UtilisateurServiceImplTest' class is selected. The code editor shows the following Java code:

```
19 no usages hamza
20 @SpringBootTest
21 @Transactional
22 class UtilisateurServiceImplTest {
23     11 usages
24     @Autowired
25     private IUtilisateurService iUtilisateurService;
26     no usages
27     @Autowired
28     private IMapper modelMapper;
29     8 usages
30     private UtilisateurDto utilisateurDto;
31 }
```

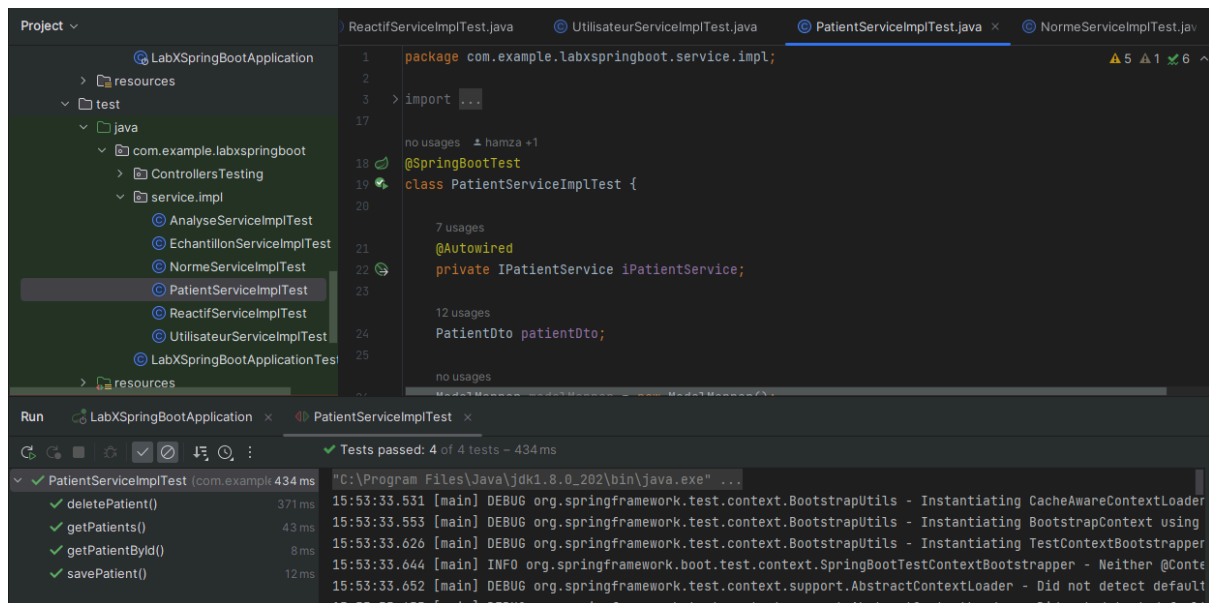
The Run tab shows the test results for 'UtilisateurServiceImplTest' (com.example.labxspringboot) with 5 tests passed in 414 ms:

- ✓ UtilisateurServiceImplTest 414 ms
- ✓ updateUtilisateur() 344 ms
- ✓ deleteUtilisateur() 31 ms
- ✓ getUtilisateurById() 12 ms
- ✓ getUtilisateurs() 12 ms
- ✓ saveUtilisateur() 15 ms

The console output shows the following logs:

```
15:52:53.844 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoader
15:52:53.866 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using
15:52:53.939 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper
15:52:53.956 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @Conte
15:52:53.963 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
15:52:53.963 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default
```

UtilisateurServiceTest



PatientServiceTest

Livraison:

Livraison du code source avec la documentation.

Conclusion:

LabXpert, intégrant des fonctionnalités complètes, une documentation Swagger claire, et une gestion efficace des tâches avec Trello, illustre notre engagement envers l'excellence médicale. Nous sommes confiants que cette solution apportera des améliorations notables aux opérations du laboratoire TechLab.