Nourou DIAKHO

INE : N04610220191

Présentation du Projet "Système de Bibliothèque"

▢ Introduction

Le projet que nous avons réalisé est une **application de gestion de bibliothèque** développée avec **Spring Boot**.
Notre objectif est de permettre deux types d'utilisateurs d'interagir avec notre système :

- **Les utilisateurs finaux** (lecteurs) peuvent consulter les livres disponibles via des **API REST**.
- **Les bibliothécaires** peuvent gérer les livres (ajouter, modifier, supprimer) via un **service SOAP** sécurisé.

Cette séparation permet de bien **distinguer les rôles** et de **modulariser** le projet.

▢ Architecture Globale

Notre projet suit une architecture **multicouche** propre et standard :

- **Contrôleur** : REST Controller ou SOAP Endpoint
- **Service** : Logique métier
- **Repository** : Accès aux données Base de données
- **Modèle** : Entités JPA

▢ Structure du projet

Dans Eclipse, le projet est organisé ainsi :

a) Package com. Bibliothèque

- BibliosystemApplication.java :
  Classe principale qui lance l'application Spring Boot (@SpringBootApplication).

b) Package com.bibliotheque.config

- WebServiceConfig.java :
  Configure les services SOAP dans Spring Boot (exposition des endpoints WSDL).
- DataInitializer.java :
  Initialise des données de test automatiquement dans la base au démarrage (livres, utilisateurs, réservations).

c) Package com.bibliotheque.controller (API REST)

- LivreController.java :
  Permet aux utilisateurs de :
  - Récupérer tous les livres
  - Voir un livre spécifique
- ReservationController.java :
  Permet de gérer les réservations.
- UtilisateurController.java :
  Permet de gérer les utilisateurs.

=> Ces contrôleurs exposent des endpoints REST.

d) Package com.bibliotheque.model (Entités)

- Livre.java :
  Représente un livre avec titre, auteur, disponibilité, etc.
- Reservation.java :
  Gère les réservations de livres par les utilisateurs.
- Utilisateur.java :
  Représente un utilisateur de la bibliothèque.

=> Les entités sont mappées avec JPA pour la base de données.

e) Package com.bibliotheque.repository (Accès aux données)

- LivreRepository.java
- ReservationRepository.java
- UtilisateurRepository.java

=> Repositories basés sur Spring Data JPA pour interagir avec la base de données (CRUD automatique).

f) Package com.bibliotheque.service (Services Métier)

Contient les logiques métiers générales.

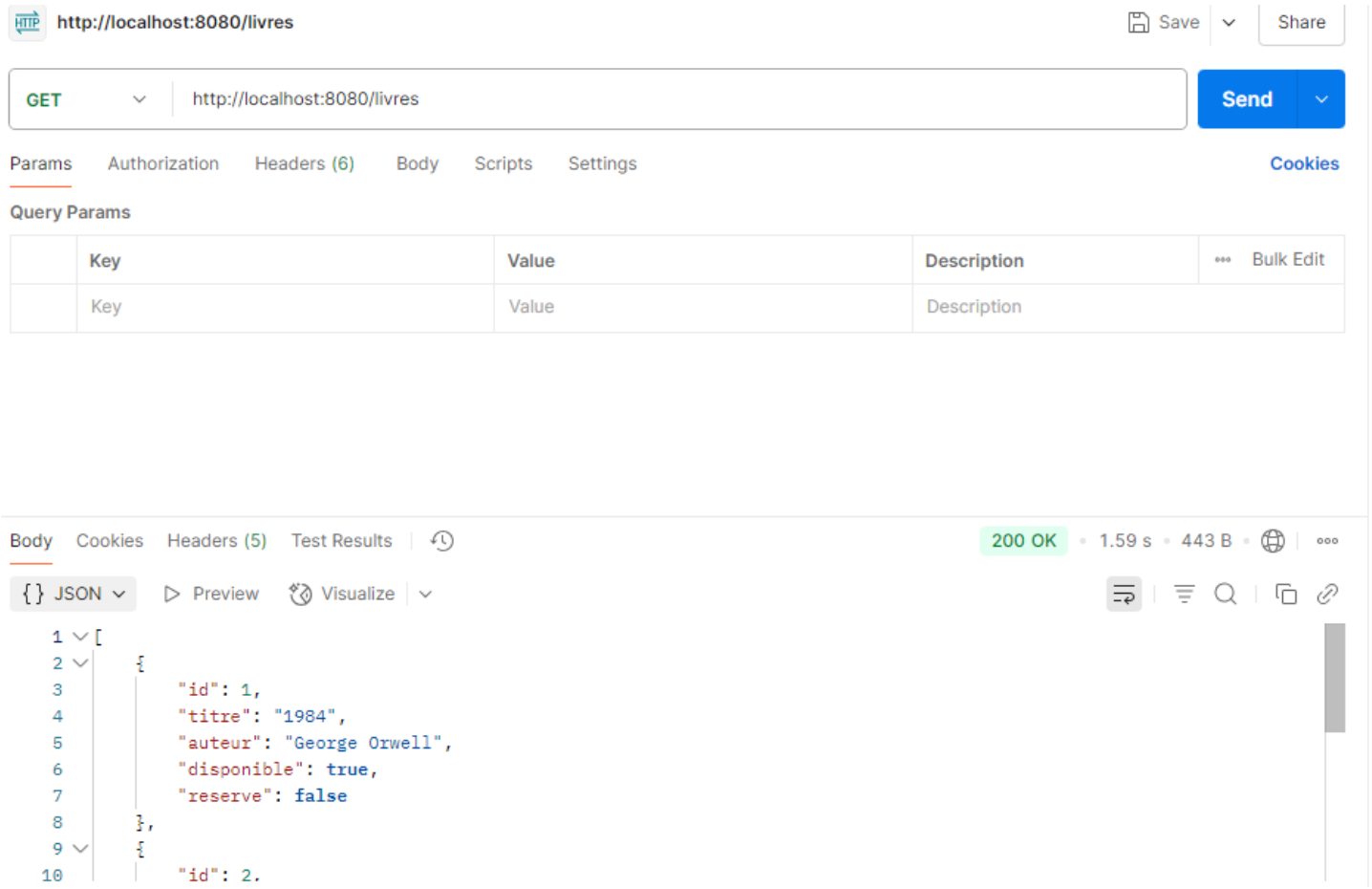
g) Package com.bibliotheque.soap (Services SOAP)

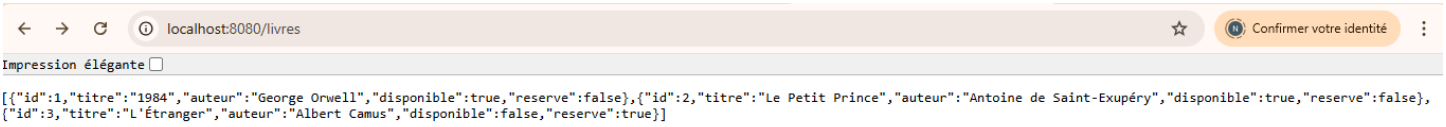
- BibliothecaireServiceImpl.java :
  Le service SOAP principal pour les bibliothécaires qui permet :
  - Ajouter un livre
  - Modifier un livre
  - Supprimer un livre
  - Consulter un livre par ID
  - Lister tous les livres
- BibliothecaireSOAPService.java, ReservationSOAPService.java :
  Autres services SOAP pour étendre si besoin.

▢ Structure du projet

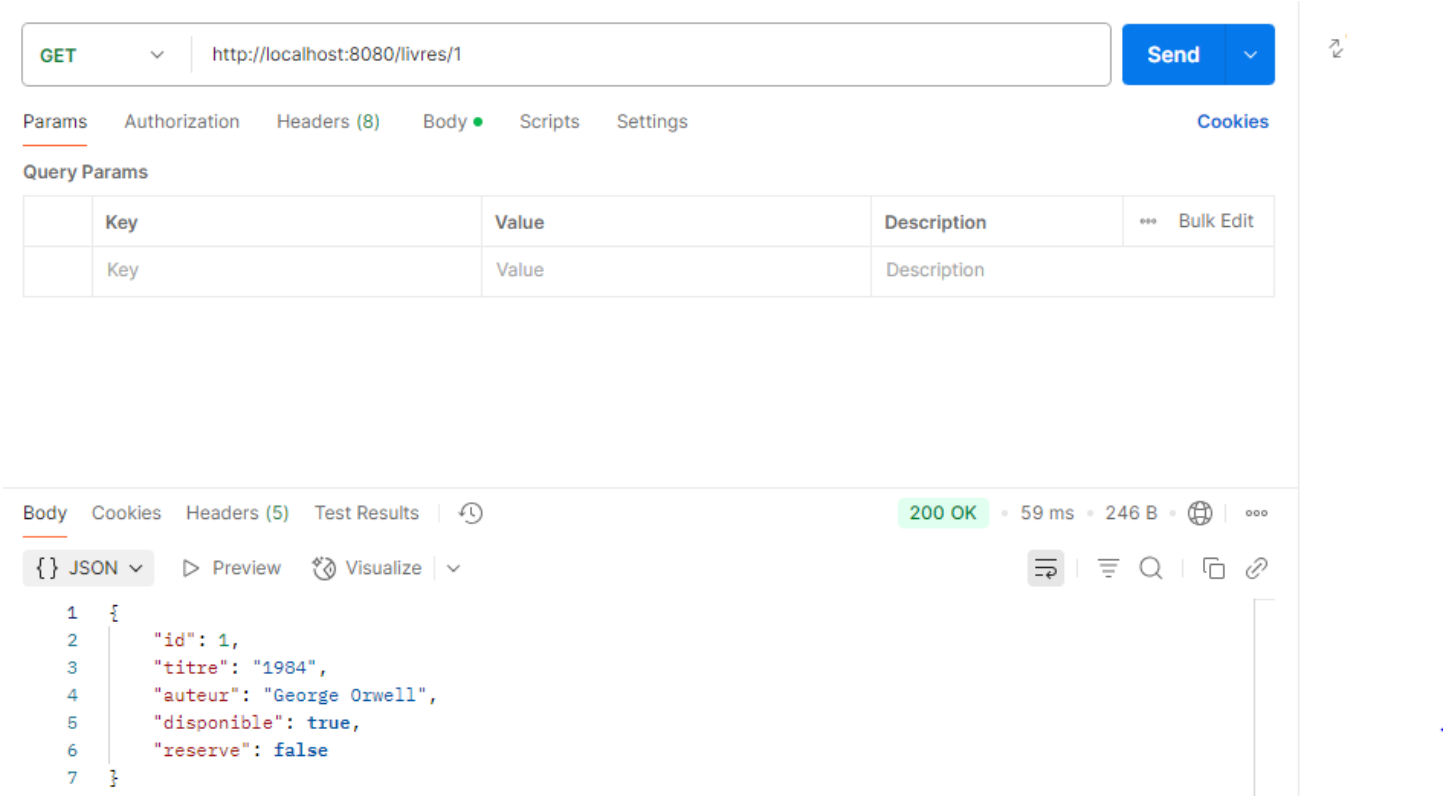
Les utilisateurs REST (clients) utilisent Postman ou un navigateur pour accéder :

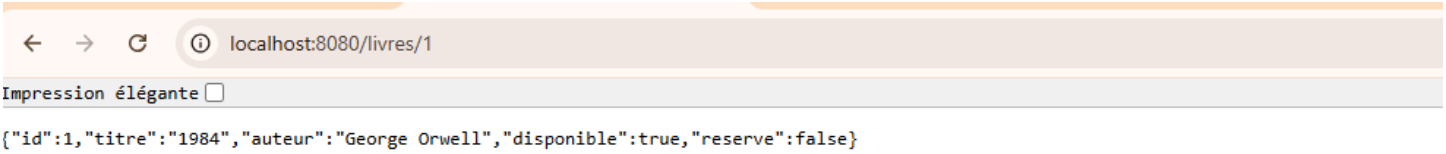
- (‘GET /livres’) : Récupération de la liste des livres



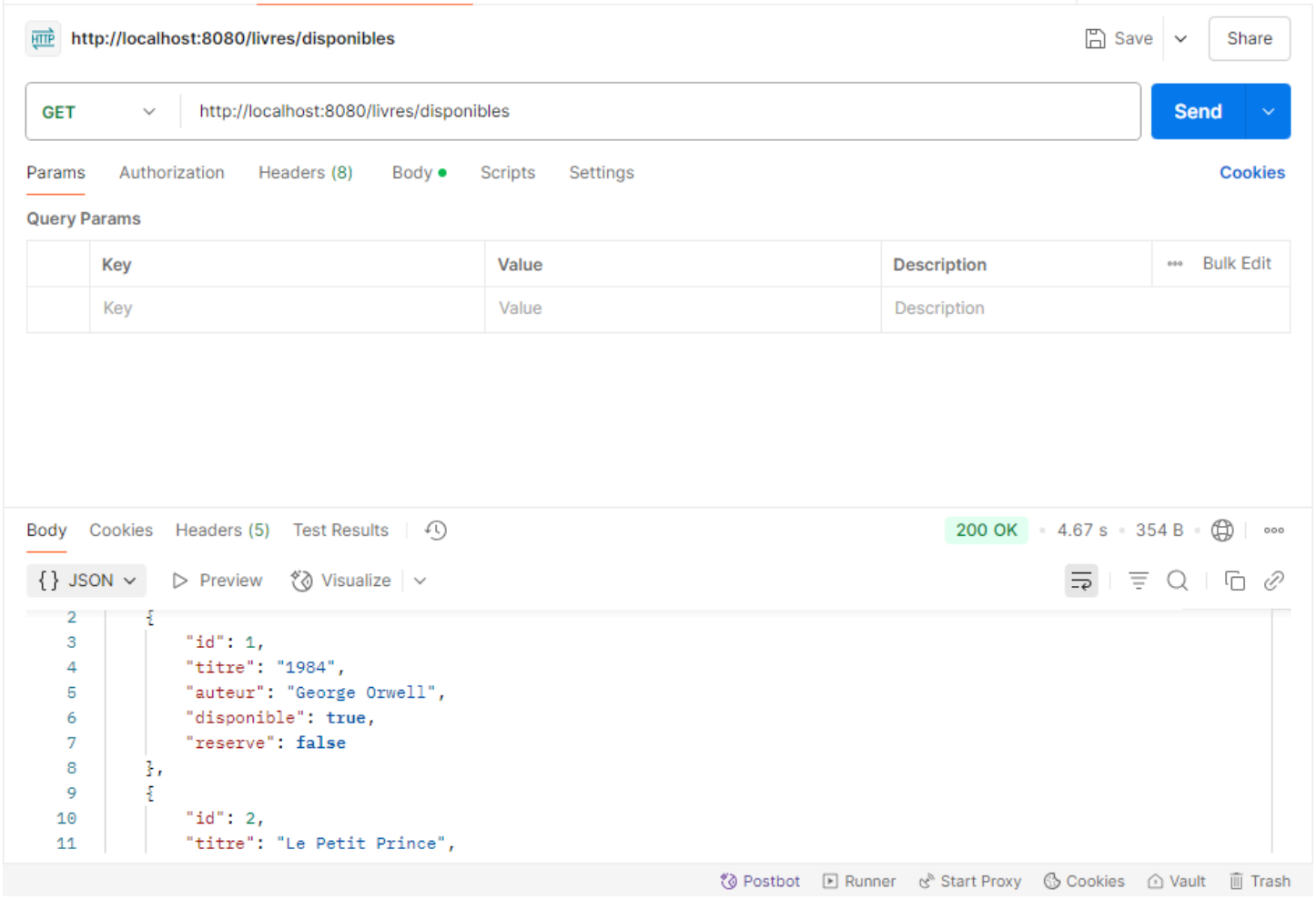


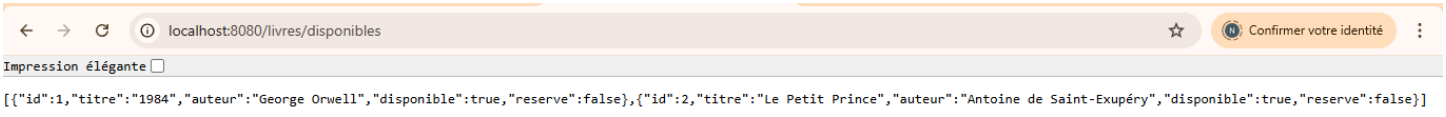
- (‘GET /livres/{id}’) : Afficher les informations d’un livre





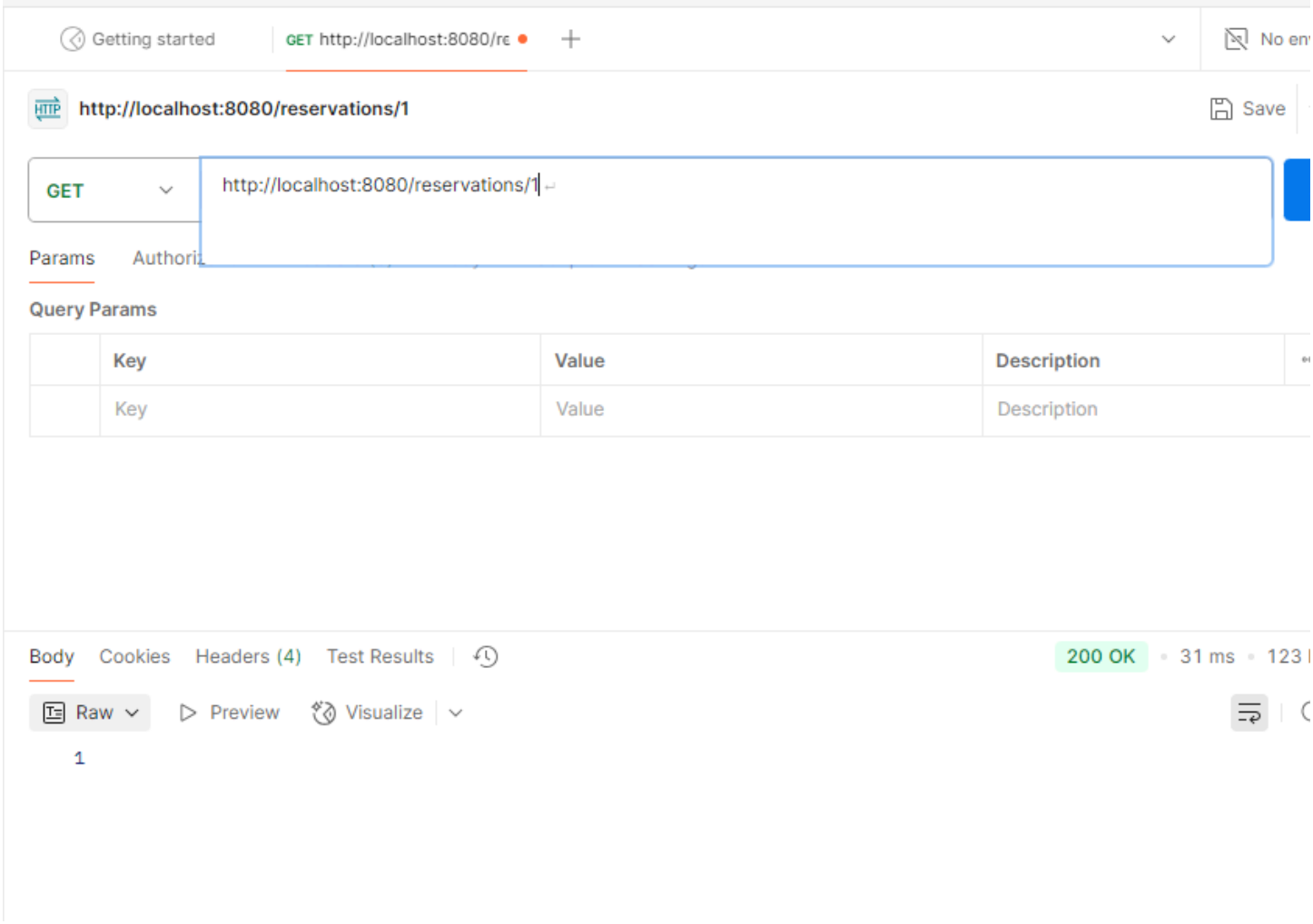
- (‘GET /livres/disponibles’) : Retourne les livres disponibles c’est-à-dire non prêtés et non réservés.



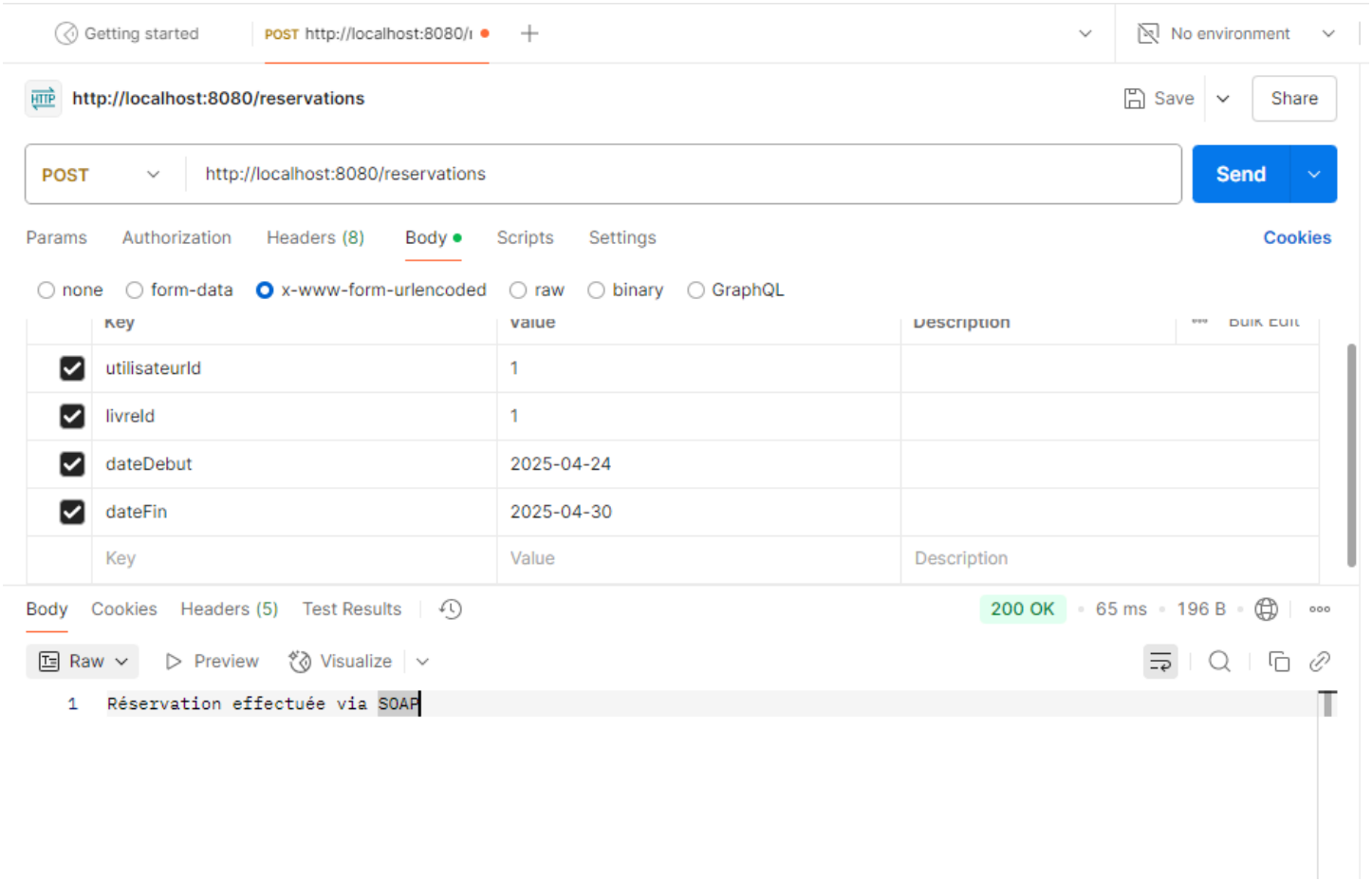


[{"id":1,"titre":"1984","auteur":"George Orwell","disponible":true,"reserve":false},{"id":2,"titre":"Le Petit Prince","auteur":"Antoine de Saint-Exupéry","disponible":true,"reserve":false}]

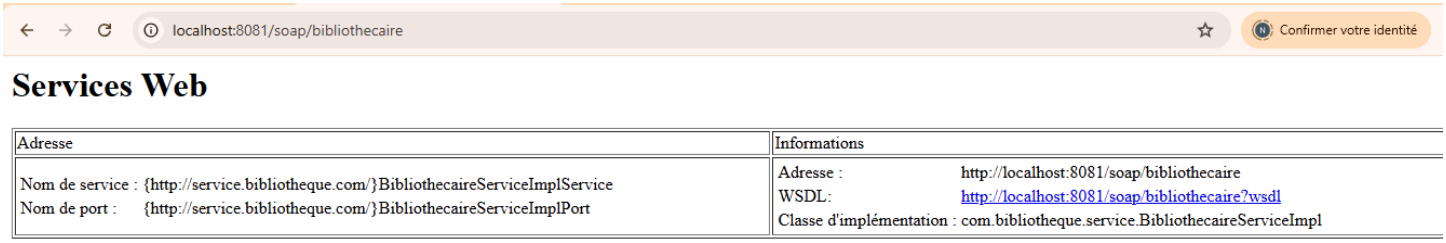
- (‘GET /reservations/{id}’) : Pour le suivi d’une réservation



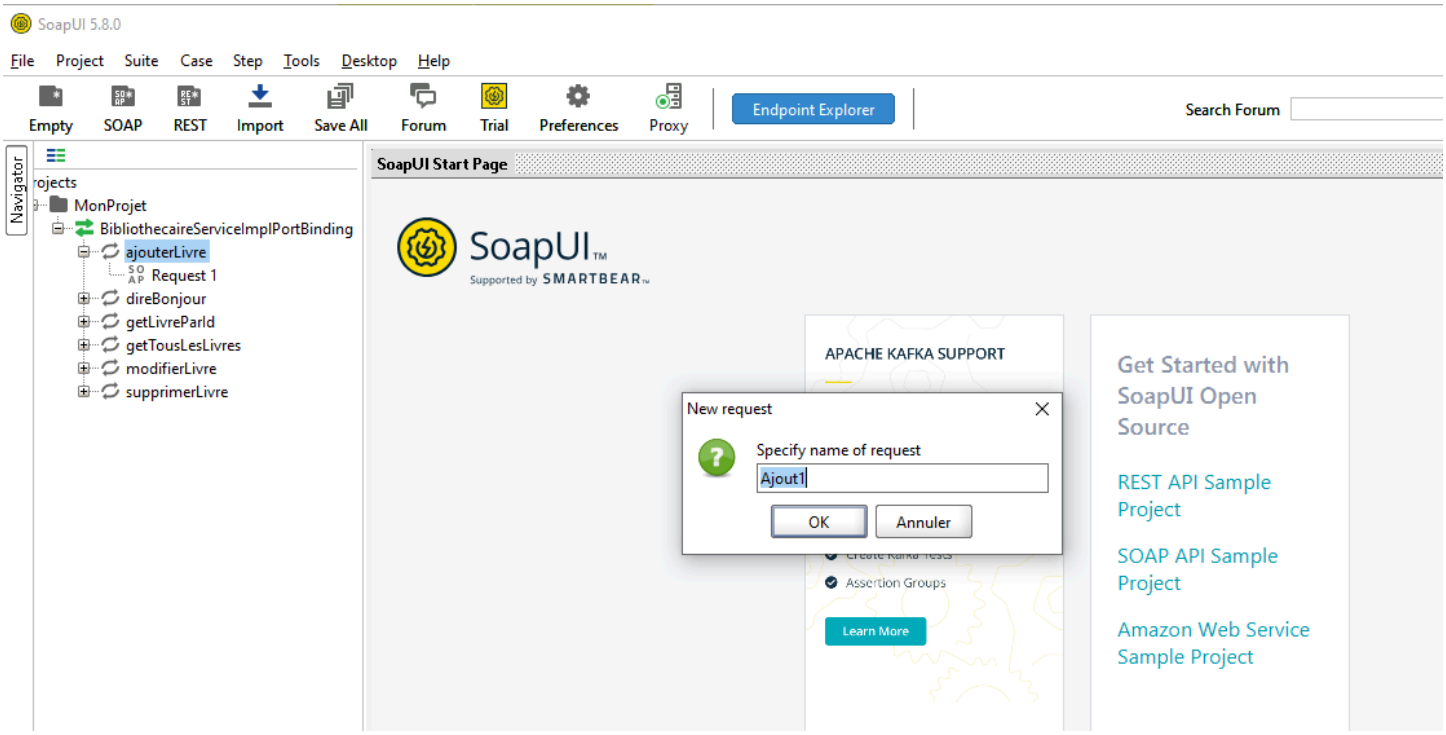
- (‘POST /reservations’) : Réservation d’un livre donné à une période précise (Passer par un endpoint SOAP)



- **Les bibliothécaires** utilisent SoapUI pour accéder :
  - /soap/bibliothecaire?wsdl → endpoint WSDL



  - Ils peuvent ajouter, modifier, supprimer des livres via des requêtes SOAP.



▢ Base de Données

▢

Nous utilisons **H2 Database** pour le développement (base en mémoire).

Les entités sont synchronisées automatiquement avec la base grâce à JPA.

Un jeu de données initial est chargé automatiquement via DataInitializer.java.

▢ Points Techniques

- **Spring Boot** pour le backend rapide et léger.
- **Spring Data JPA** pour la gestion de la persistance.
- **Jakarta JAX-WS** pour exposer des web services SOAP.
- **H2 Database** pour le développement local.

▢ Conclusion

Ce projet met en œuvre les bonnes pratiques de développement d'une application Spring Boot professionnelle,
en combinant API REST pour l'accès public et Web Service SOAP pour l'administration sécurisée.

Notre architecture modulaire permet de facilement ajouter de nouvelles fonctionnalités ou changer la base de données en production.