**Slide 1 : Conception et Implémentation d’un Système de Gestion de Bibliothèque en REST et SOAP avec Spring Boot**

**Projet : Système de gestion de bibliothèque**  
**Réalisé par : Yaye Fatou Beye et Tening Sene**  
**Encadrant :**  
**Date : 01/05/2025**

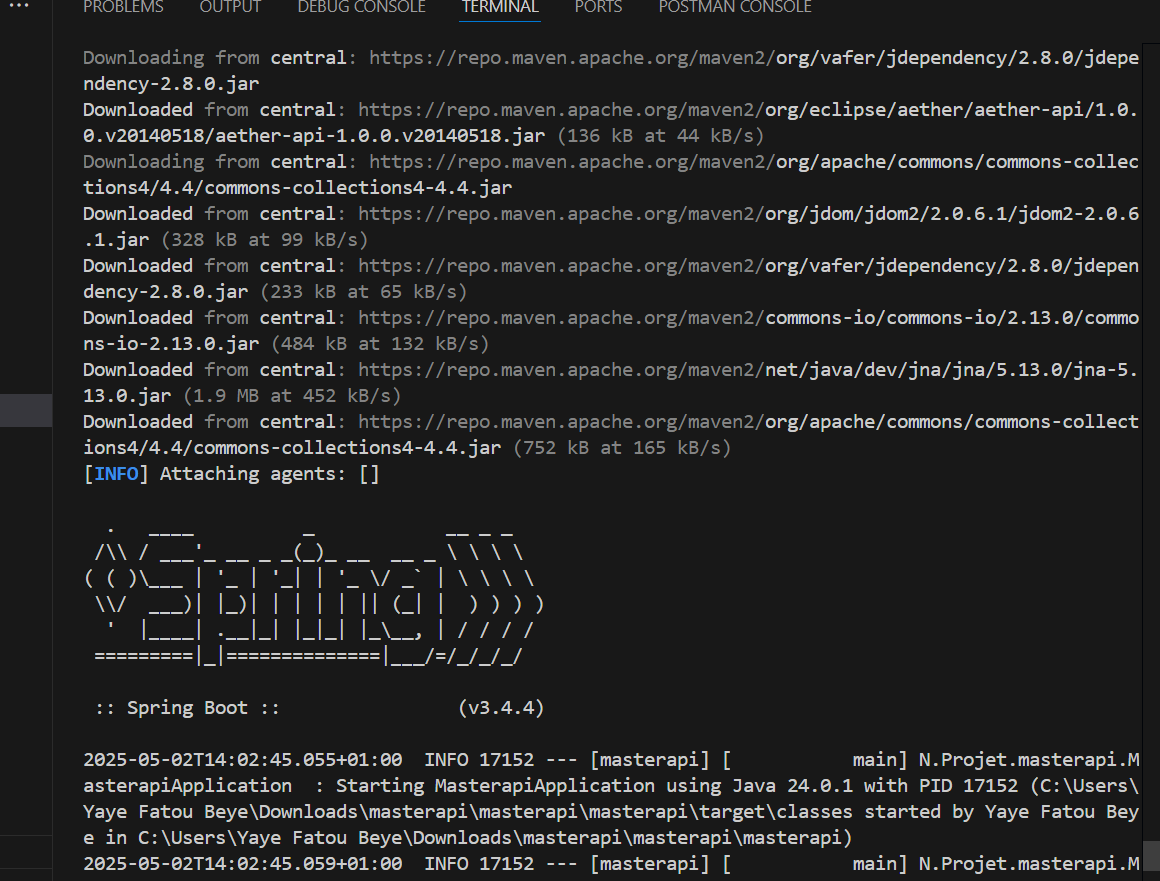
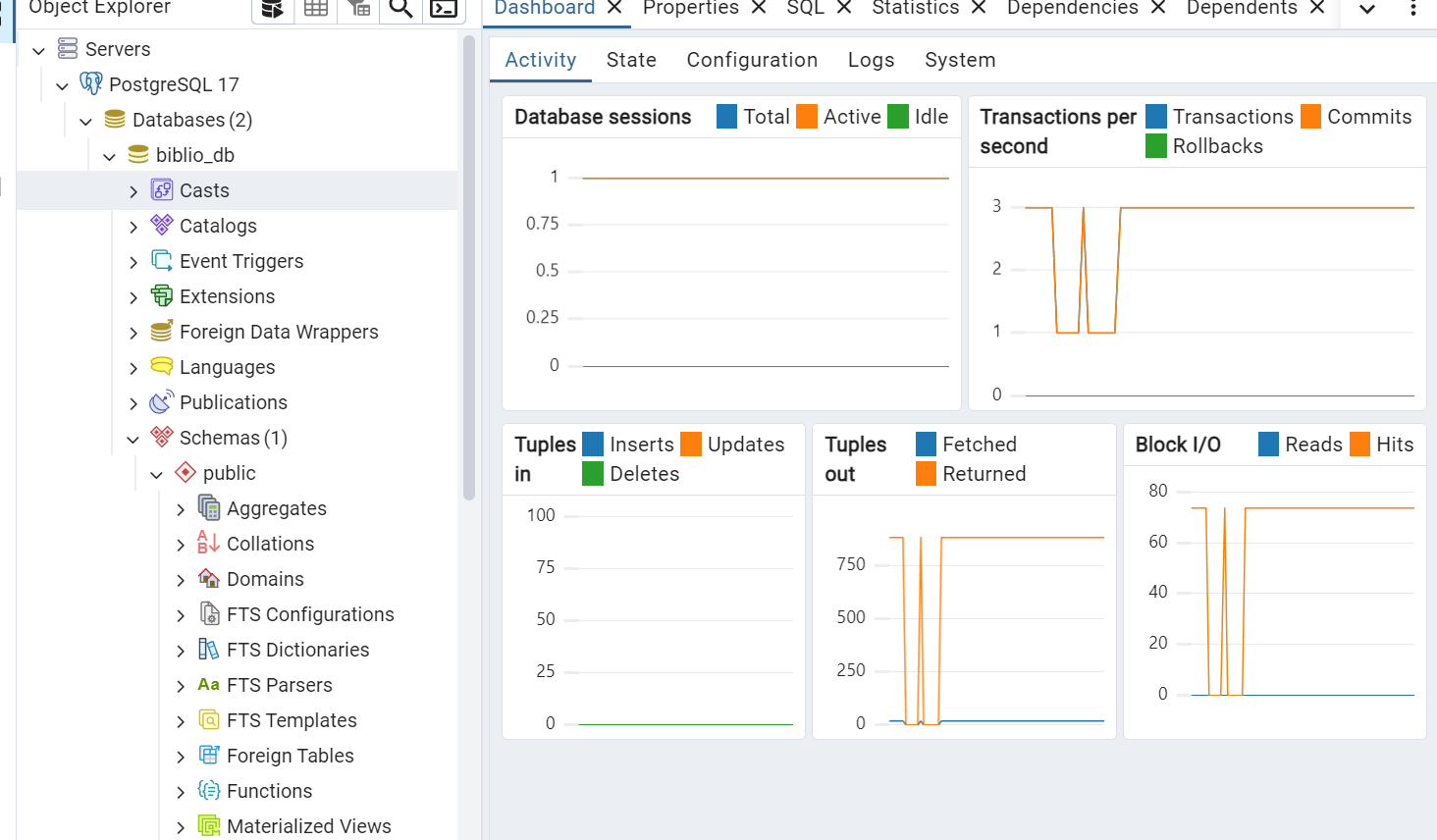
**🧭 Slide 2 : Objectif du projet**

Ce projet a pour but de concevoir un système de gestion de bibliothèque moderne, combinant deux types d’API :

* **Une API REST** destinée aux étudiants et professeurs, leur permettant de consulter les livres, vérifier leur disponibilité et effectuer des réservations.
* **Une API SOAP** réservée aux bibliothécaires pour gérer le catalogue, les prêts et les retours de manière sécurisée.

L’objectif est de séparer les responsabilités et de faciliter l’intégration future avec d’autres systèmes.

**Configuration de la base de donnee avec sprint boot et pgAdmin**



**⚙️ Slide 3 : Architecture générale du système**

L’application repose sur une architecture en couches avec séparation REST/SOAP.

* Backend développé en **Java avec Spring Boot**
* API SOAP via **Spring Web Services**
* Base de données relationnelle **PostgreSQL**
* Tests fonctionnels réalisés avec **Postman** (REST) et **SoapUI** (SOAP)
* (Optionnel) Interface utilisateur avec React

┌───────────────────────┐

│ Utilisateurs │

│ (Étudiants, Professeurs) │

└────────────┬──────────┘

│ REST

▼

┌────────────────────────────┐

│ API REST - Spring Boot │

│ (Catalogue, Réservation) │

└────────────┬───────────────┘

│

┌────────────────┘

▼

┌──────────────────────────────┐

│ Appel interne vers API SOAP │ ←── (via WebServiceTemplate)

└────────────┬─────────────────┘

│ SOAP

▼

┌──────────────────────────────────┐

│ API SOAP - Spring Web Services │

│(Gestion interne : stock, prêts, │

│ retours, ajout/suppression livre)│

└────────────┬─────────────────────┘

│

▼

┌────────────────┐

│ Base de données│

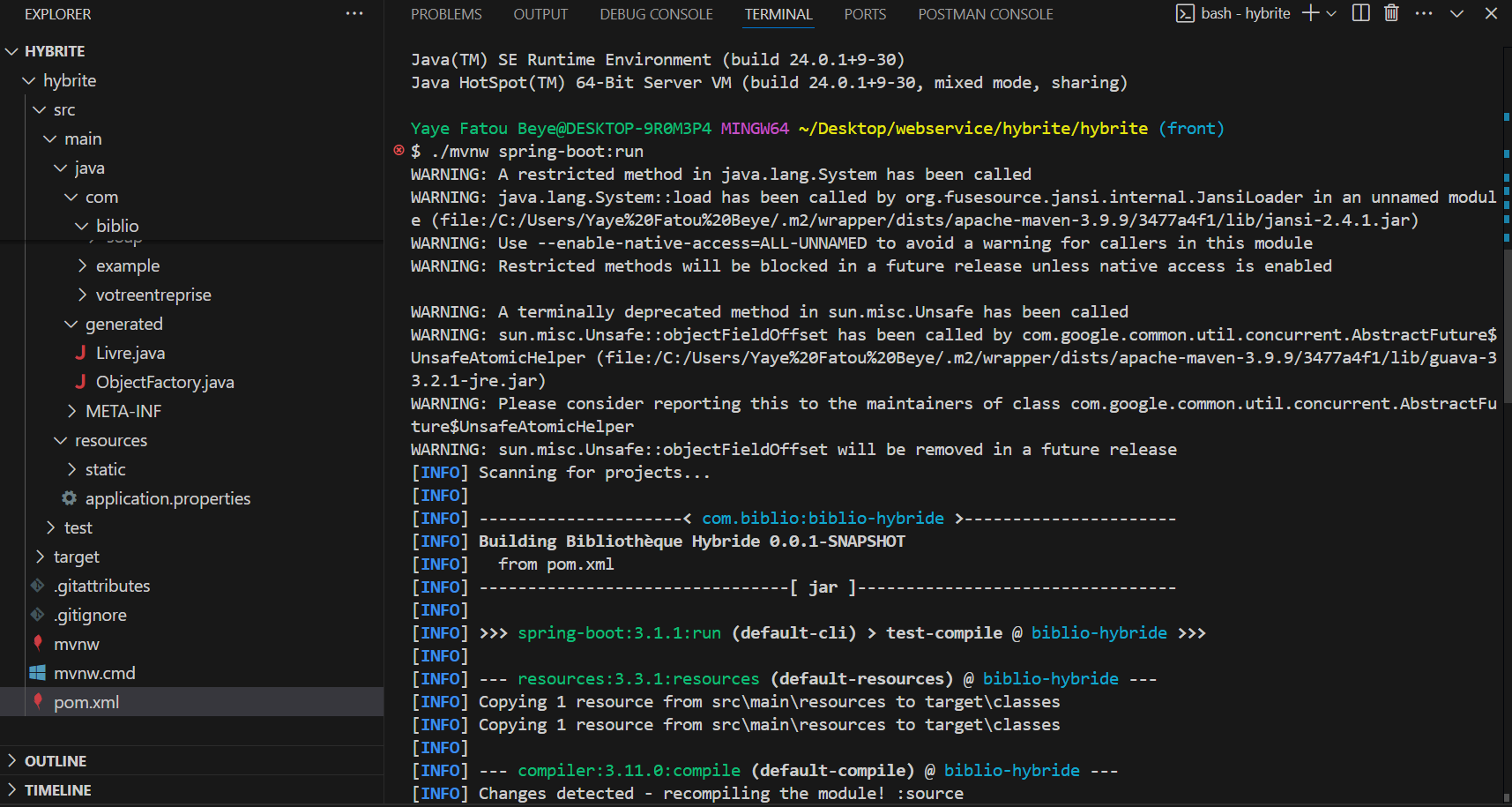
│ PostgreSQL / H2│

└────────────────┘

**🔗 Slide 4 : Fonctionnalités – API REST**

L’API REST permet l’accès public au catalogue. Elle comprend les routes suivantes :

* GET /livres → Liste de tous les livres
* GET /livres/{id} → Informations d’un livre précis
* GET /livres/disponibles → Liste des livres non réservés/prêtés
* GET /reservations/{id} → Détail d’une réservation
* POST /reservations → Réservation d’un livre → redirige vers une méthode SOAP

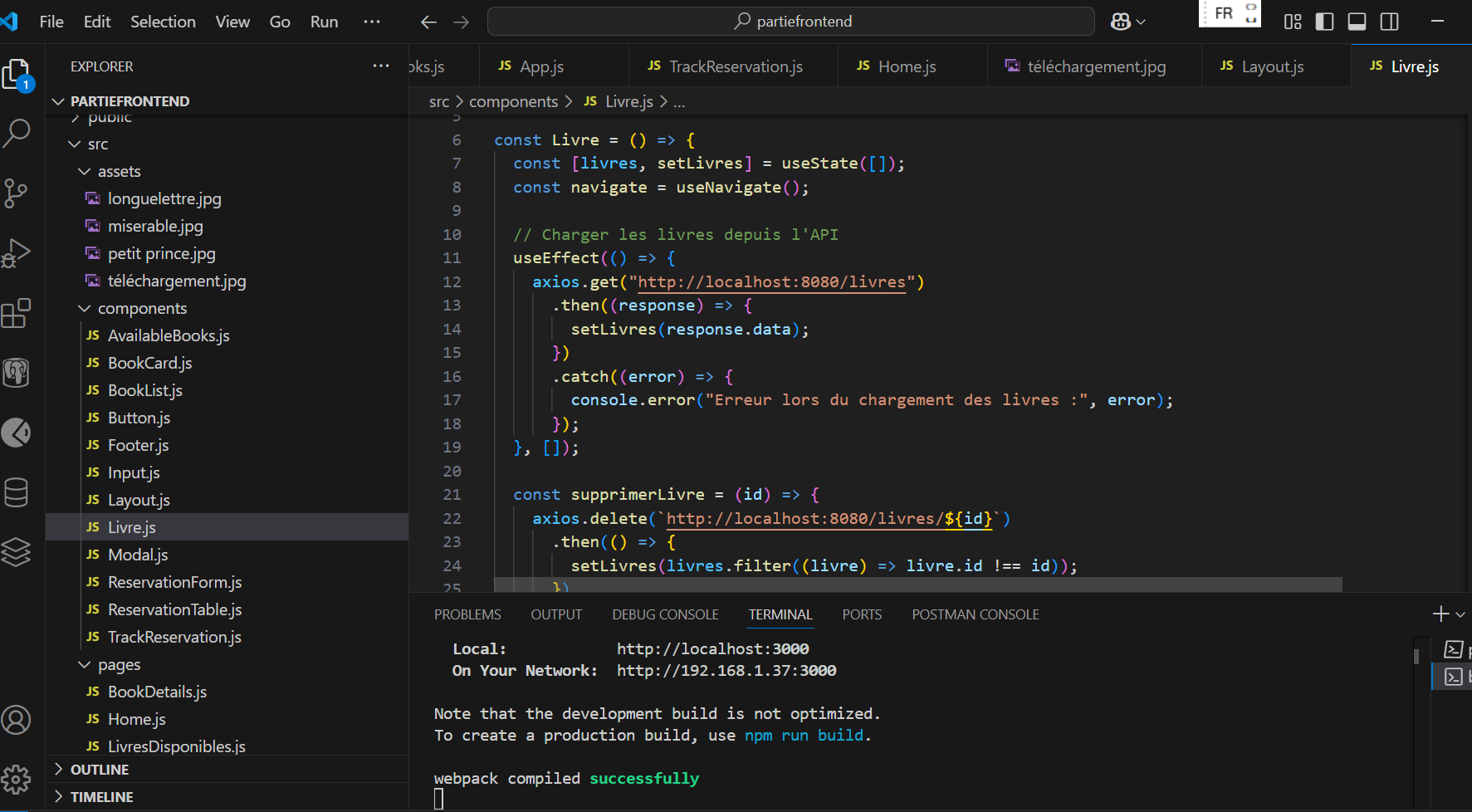


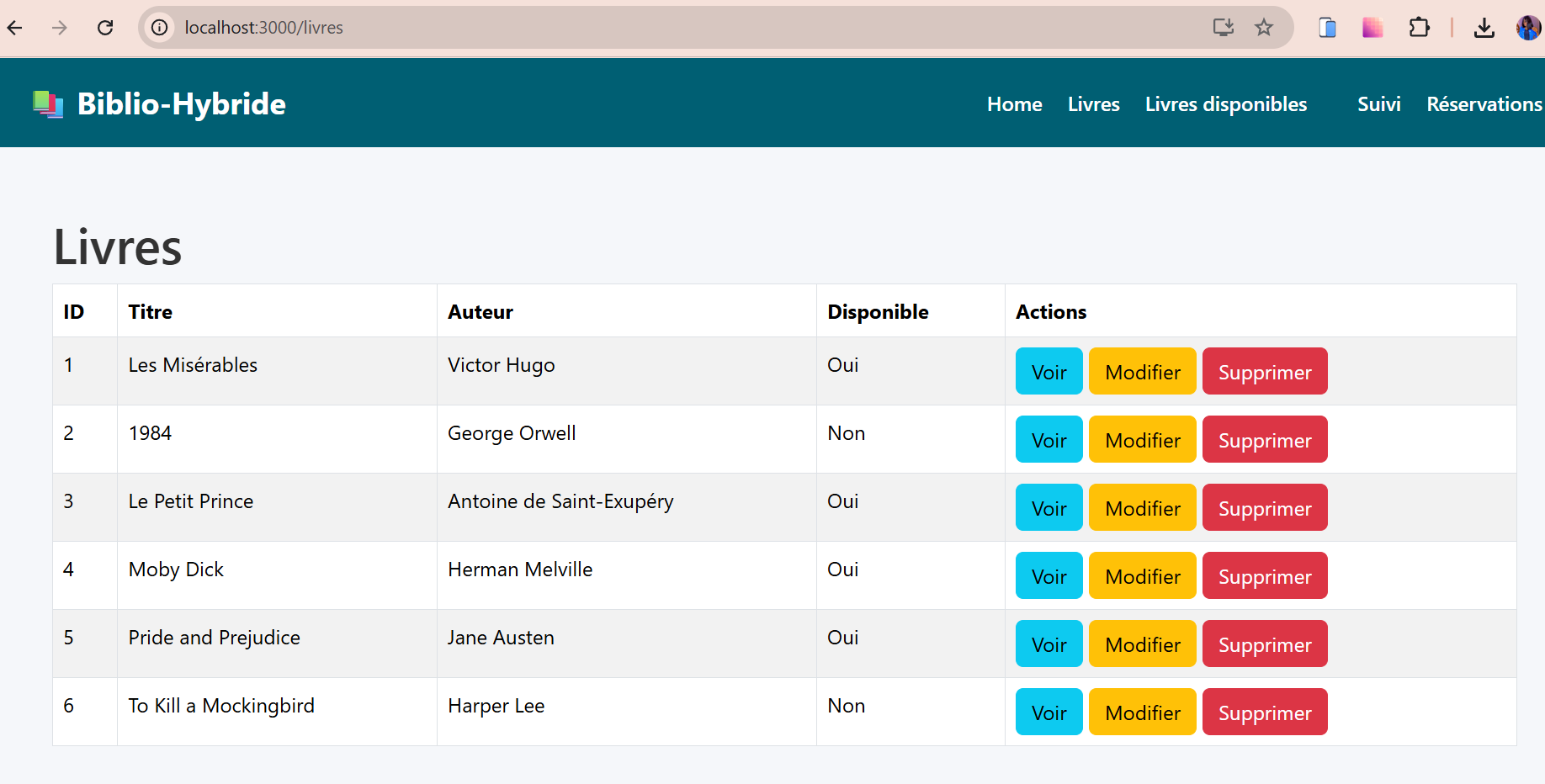
**🧼 Slide 5 : Fonctionnalités – API SOAP**

L’API SOAP gère les opérations internes des bibliothécaires :

* ajouterLivre(Livre)
* modifierLivre(Long id, Livre)
* supprimerLivre(Long id)
* preterLivre(Long userId, Long livreId)
* retournerLivre(Long userId, Long livreId)

Une interface utilisateur simple a été développée en React pour faciliter l’accès aux livres

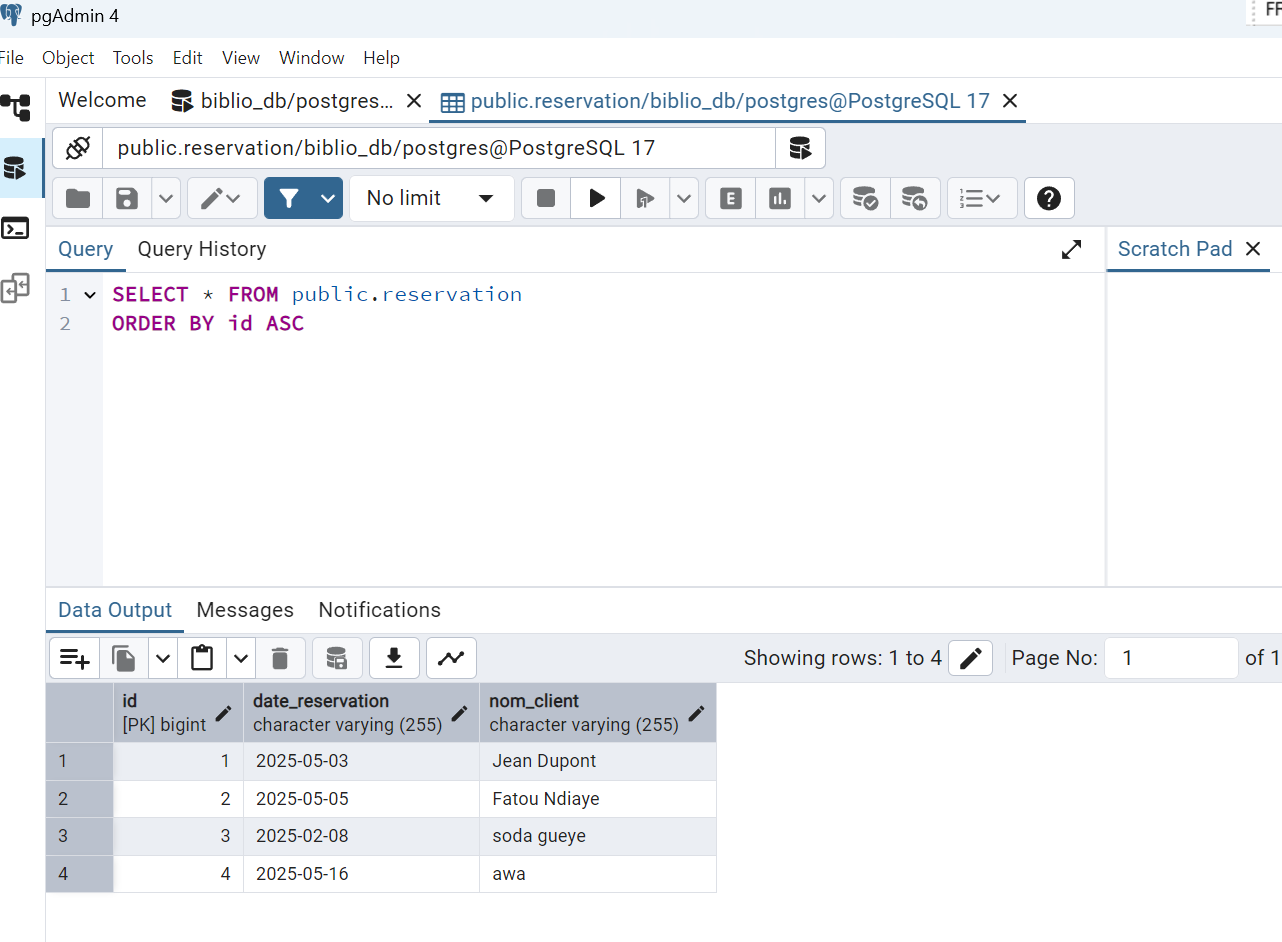


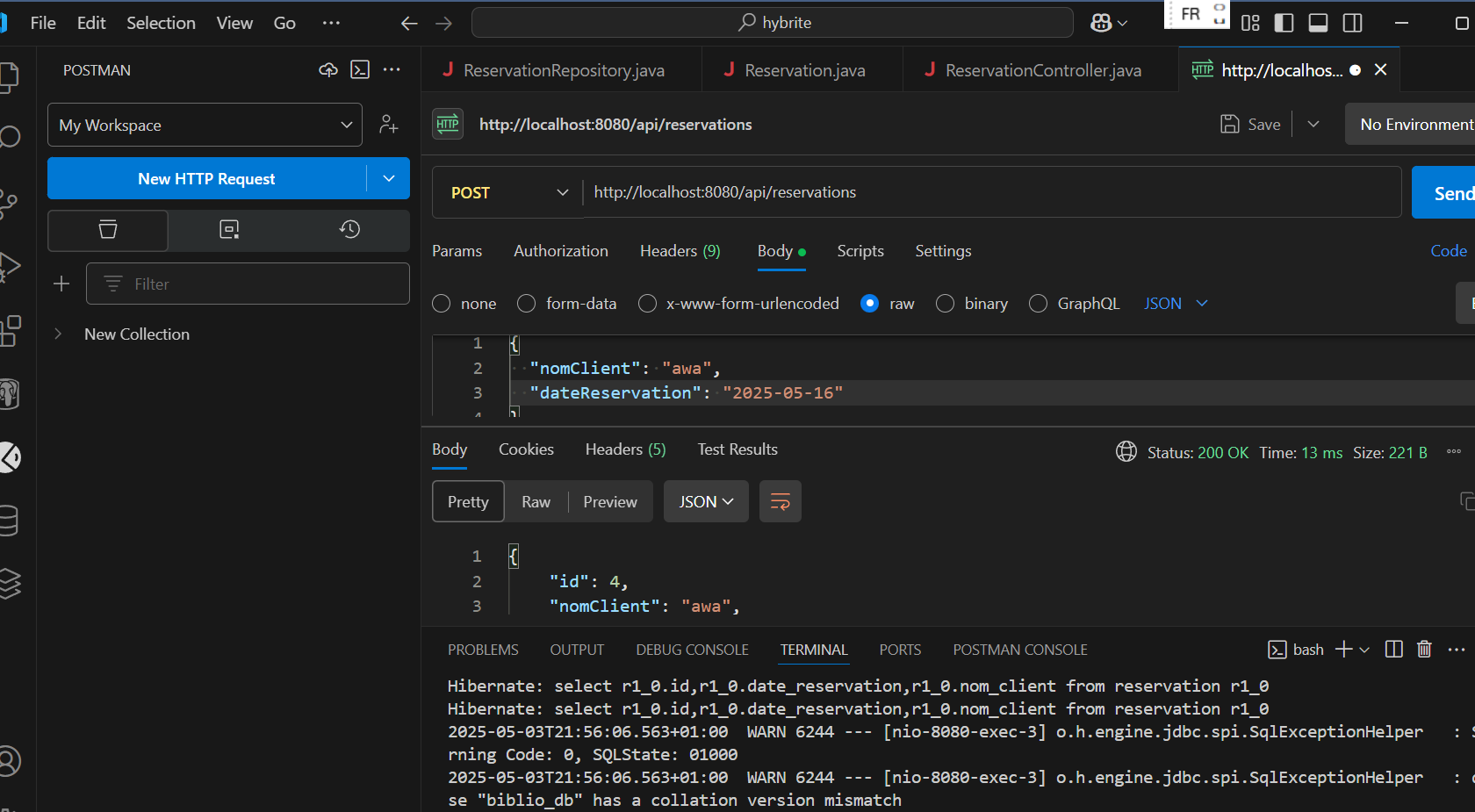




**🗃️ Slide 6 : Modélisation de la base de données**

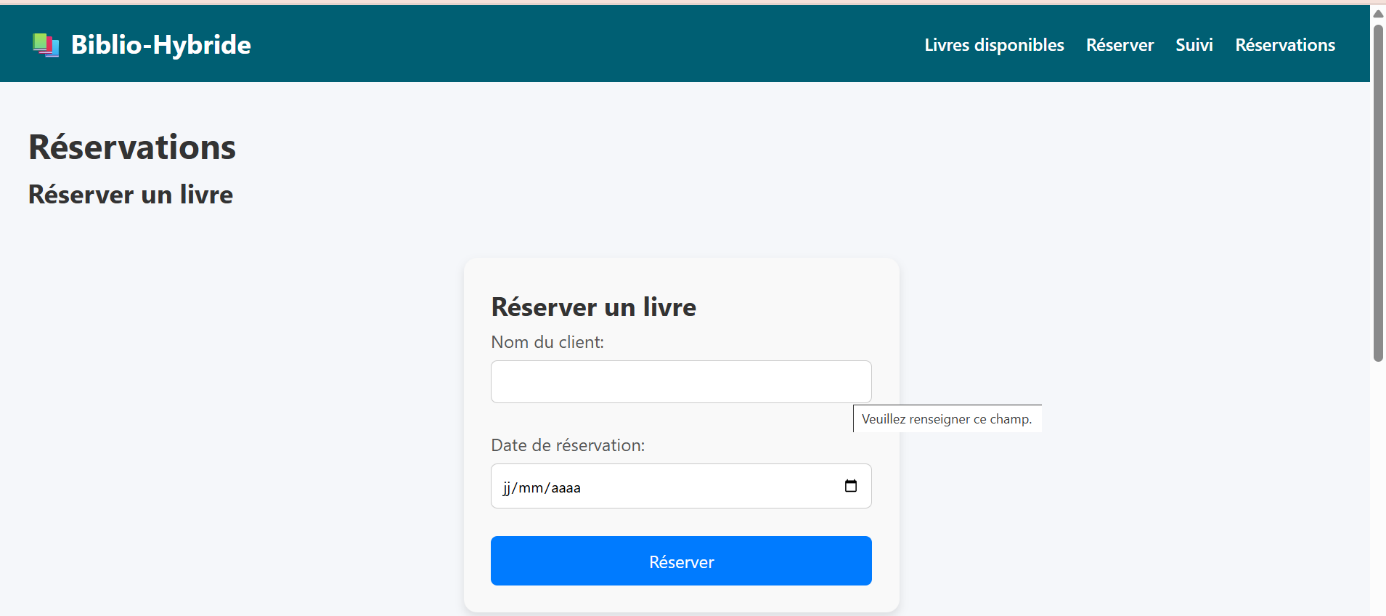
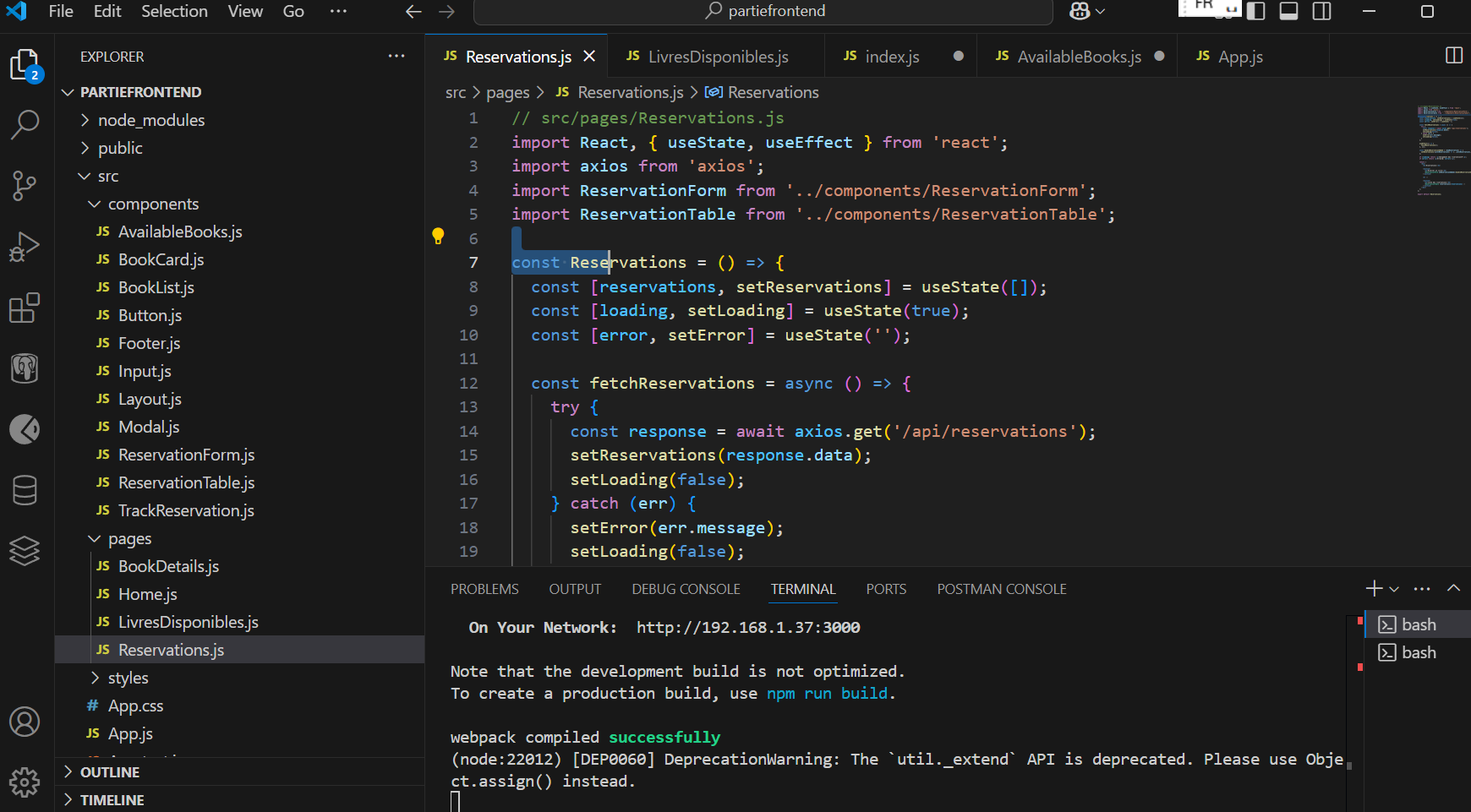
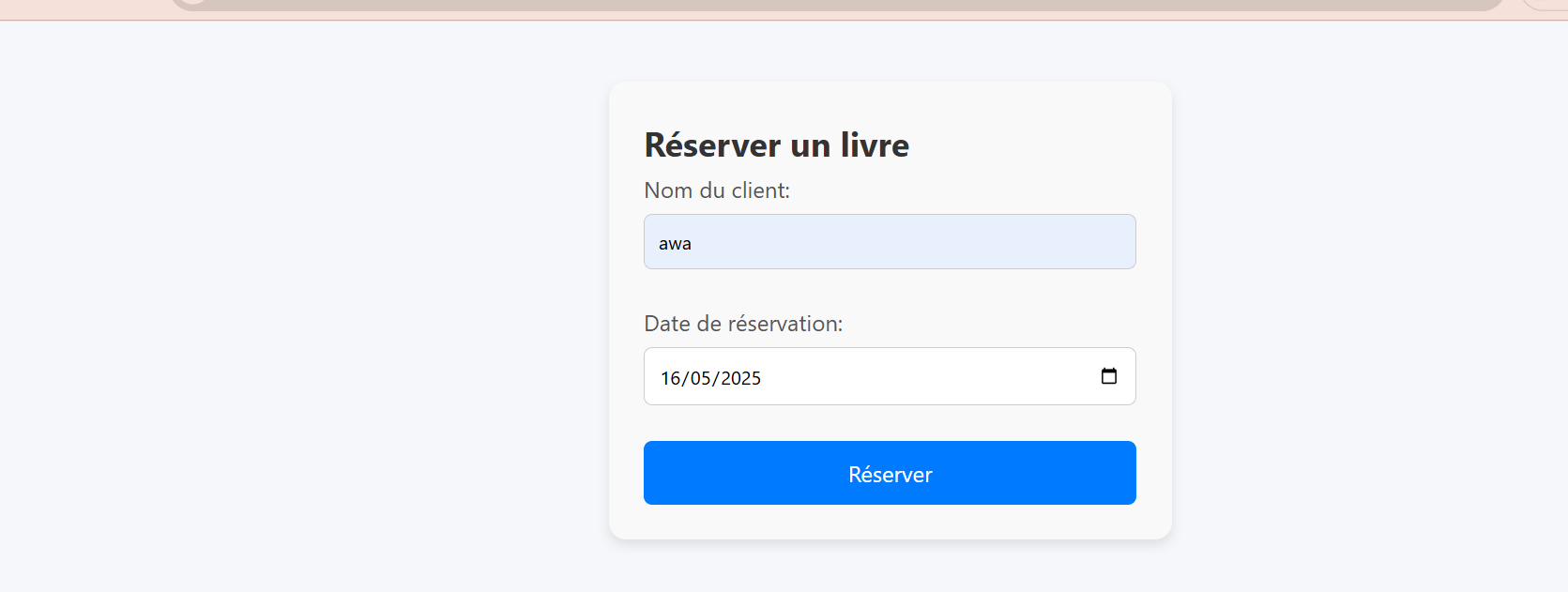
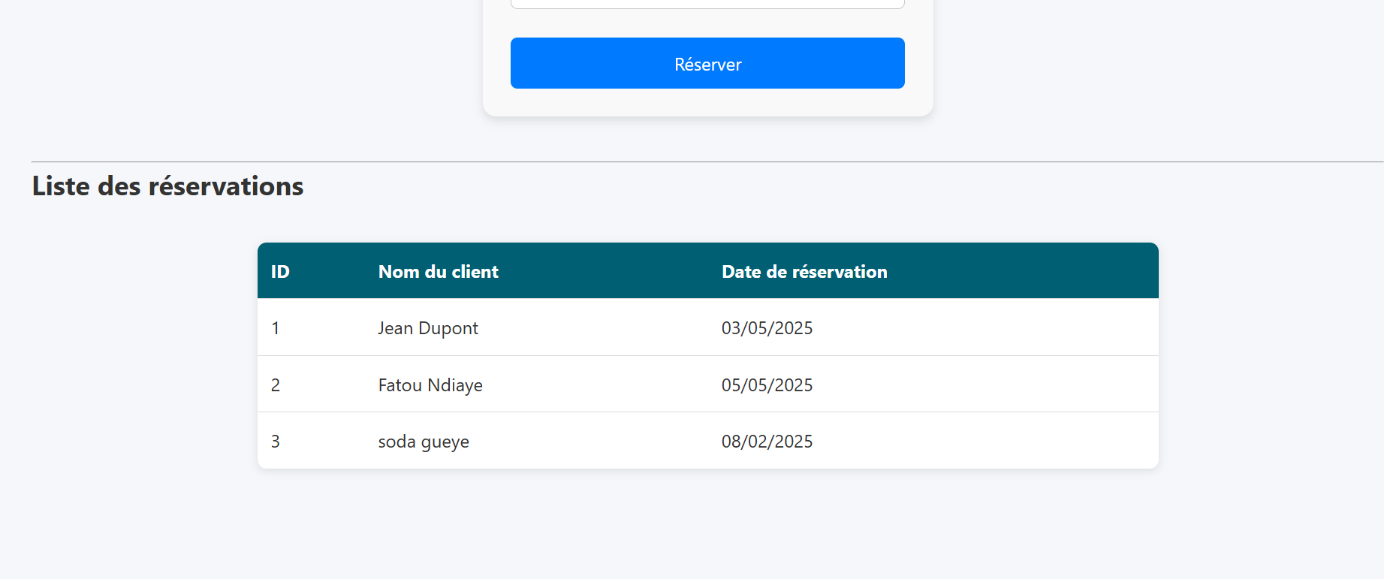
La base PostgreSQL comprend les entités suivantes :

* Livre (id, titre, auteur, statut)
* Utilisateur (id, nom, rôle)
* Reservation (id, utilisateur\_id, livre\_id, date\_debut, date\_fin)
* Pret (id, utilisateur\_id, livre\_id, date\_pret, date\_retour)
* 



**🖥️ Slide 7 : Frontend (optionnel)**

Une interface utilisateur simple a été développée en React pour faciliter l’accès au catalogue et aux réservations par les utilisateurs.

**❌ Slide 8 : Problèmes rencontrés**

🔧 **1. Lancement du projet Spring Boot**

* Lors de la création initiale du projet, j’ai rencontré des erreurs de configuration liées au port (8080 déjà utilisé) et à l’absence de dépendances SOAP.
* Résolu en changeant le port par défaut dans application.properties et en ajoutant les dépendances nécessaires (spring-boot-starter-web-services, JAXB, etc.)

🔧 **2. Création de la route REST /livres**

* La route GET /livres ne répondait pas car la méthode du contrôleur n’était pas correctement annotée avec @GetMapping.
* Problème aussi avec la sérialisation JSON (boucles infinies) à cause des relations bidirectionnelles JPA → résolu avec @JsonManagedReference / @JsonBackReference.

🔧 **3. Intégration REST ↔ SOAP**

* Le plus gros défi technique a été d’appeler une opération SOAP (réserverLivre) depuis une route REST. J’ai dû créer un **client SOAP avec WebServiceTemplate** côté REST.

**🧾 Slide 09 : Dépôt GitHub**

📁 Le projet est disponible sur GitHub :

https://github.com/fatoubeyeyaye/webservice/blob/main/Presentation%20du%20Projet%20webservice.docx

**Slide 11 : Conclusion et retour d’expérience**

Ce projet m’a permis de découvrir :

* La mise en œuvre d’une architecture REST & SOAP combinée
* La gestion fine des entités en Java avec Spring Data JPA
* Les tests multi-couches d’un backend

Haut du formulaire

Bas du formulaire