

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

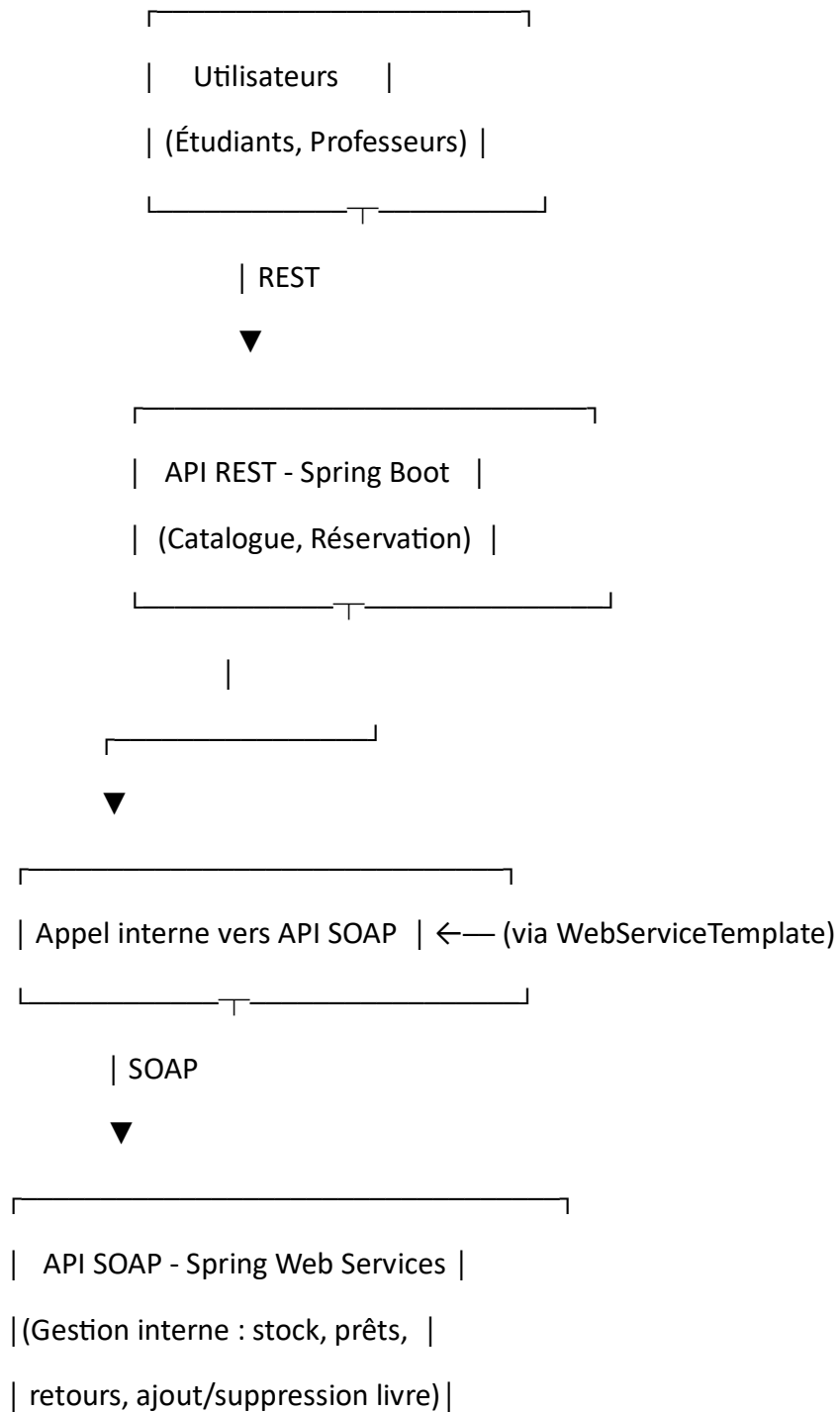
Ce projet a pour but de concevoir un système de gestion de bibliothèque moderne, combinant deux types d'API :

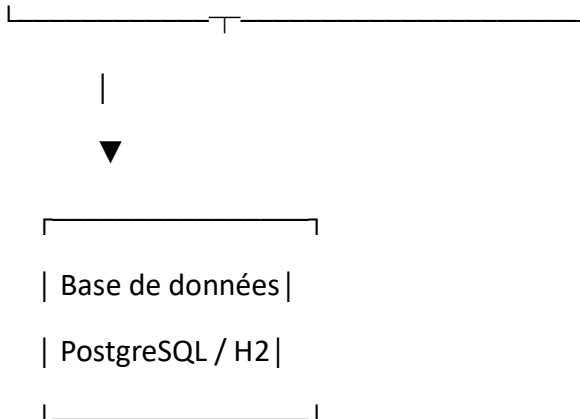
- **Une API REST** destinée aux étudiants et professeurs, leur permettant de consulter les livres, vérifier leur disponibilité et effectuer des réservations.
- **Une API SOAP** réservée aux bibliothécaires pour gérer le catalogue, les prêts et les retours de manière sécurisée.

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

Configuration de la base de donnée avec sprint boot et pgAdmin

- 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





🔗 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

The screenshot shows an IDE with a project named 'HYBRITE'. The project structure includes a 'src' directory with 'main' and 'test' subdirectories. The 'main' directory contains 'java' and 'resources' subdirectories. The 'resources' directory contains 'application.properties'. The 'test' directory contains 'test-compile' and 'test-run' subdirectories. The terminal window shows the output of a Maven build, including warnings about deprecated methods and information about the build process.

```

Java(TM) SE Runtime Environment (build 24.0.1+9-30)
Java HotSpot(TM) 64-Bit Server VM (build 24.0.1+9-30, mixed mode, sharing)

Yaye Fatou Beye@DESKTOP-9R0M3P4 MINGW64 ~/Desktop/websevice/hybrite/hybrite (front)
$ ./mvnw spring-boot:run
WARNING: A restricted method in java.lang.System has been called
WARNING: java.lang.System::load has been called by org.fusesource.jansi.internal.JansiLoader in an unnamed module
(file:/C:/Users/Yaye%20Fatou%20Beye/.m2/wrapper/dists/apache-maven-3.9.9/3477a4f1/lib/jansi-2.4.1.jar)
WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for callers in this module
WARNING: Restricted methods will be blocked in a future release unless native access is enabled

WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::objectFieldOffset has been called by com.google.common.util.concurrent.AbstractFuture$
UnsafeAtomicHelper (file:/C:/Users/Yaye%20Fatou%20Beye/.m2/wrapper/dists/apache-maven-3.9.9/3477a4f1/lib/guava-3
3.2.1-jre.jar)
WARNING: Please consider reporting this to the maintainers of class com.google.common.util.concurrent.AbstractFu
ture$UnsafeAtomicHelper
WARNING: sun.misc.Unsafe::objectFieldOffset will be removed in a future release
[INFO] Scanning for projects...
[INFO] -----< com.biblio:biblio-hybrite >-----
[INFO] Building Bibliothèque Hybride 0.0.1-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] >>> spring-boot:3.1.1:run (default-cli) > test-compile @ biblio-hybride >>>
[INFO] --- resources:3.3.1:resources (default-resources) @ biblio-hybride ---
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO] --- compiler:3.11.0:compile (default-compile) @ biblio-hybride ---
[INFO] Changes detected - recompiling the module! :source
  
```

🔗 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

```

6  const Livre = () => {
7    const [livres, setLivres] = useState([]);
8    const navigate = useNavigate();
9
10   // Charger les livres depuis l'API
11   useEffect(() => {
12     axios.get("http://localhost:8080/livres")
13       .then((response) => {
14         setLivres(response.data);
15       })
16       .catch((error) => {
17         console.error("Erreur lors du chargement des livres :", error);
18       });
19   }, []);
20
21   const supprimerLivre = (id) => {
22     axios.delete(`http://localhost:8080/livres/${id}`)
23       .then(() => {
24         setLivres(livres.filter((livre) => livre.id !== id));
25       });
26   }


```

localhost:3000/livres

Biblio-Hybride Home Livres Livres disponibles Suivi Réservations

Livres

ID	Titre	Auteur	Disponible	Actions
1	Les Misérables	Victor Hugo	Oui	Voir Modifier Supprimer
2	1984	George Orwell	Non	Voir Modifier Supprimer
3	Le Petit Prince	Antoine de Saint-Exupéry	Oui	Voir Modifier Supprimer
4	Moby Dick	Herman Melville	Oui	Voir Modifier Supprimer
5	Pride and Prejudice	Jane Austen	Oui	Voir Modifier Supprimer
6	To Kill a Mockingbird	Harper Lee	Non	Voir Modifier Supprimer



Bienvenue dans notre bibliothèque

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)

pgAdmin 4

File Object Tools Edit View Window Help

Welcome biblio_db/postgres... public.reservation/biblio_db/postgres@PostgreSQL 17 X

public.reservation/biblio_db/postgres@PostgreSQL 17

Query Query History Scratch Pad X

```
1 SELECT * FROM public.reservation
2 ORDER BY id ASC
```

Data Output Messages Notifications

Showing rows: 1 to 4 Page No: 1 of 1

	id [PK] bigint	date_reservation character varying (255)	nom_client character varying (255)
1	1	2025-05-03	Jean Dupont
2	2	2025-05-05	Fatou Ndiaye
3	3	2025-02-08	soda gueye
4	4	2025-05-16	awa

File Edit Selection View Go ... hybrite

POSTMAN

My Workspace

New HTTP Request

Filter

New Collection

ReservationRepository.java Reservation.java ReservationController.java http://localhost:8080/api/reservations

POST http://localhost:8080/api/reservations

Params Authorization Headers (9) Body Scripts Settings

Body

```
1 {
2   "nomClient": "awa",
3   "dateReservation": "2025-05-16"
}
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 13 ms Size: 221 B

Pretty Raw Preview JSON

```
1 {
2   "id": 4,
3   "nomClient": "awa",
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```
Hibernate: select r1_0.id,r1_0.date_reservation,r1_0.nom_client from reservation r1_0
Hibernate: select r1_0.id,r1_0.date_reservation,r1_0.nom_client from reservation r1_0
2025-05-03T21:56:06.563+01:00 WARN 6244 --- [nio-8080-exec-3] o.h.engine.jdbc.spi.SqlExceptionHelper : 
rning Code: 0, SQLState: 01000
2025-05-03T21:56:06.563+01:00 WARN 6244 --- [nio-8080-exec-3] o.h.engine.jdbc.spi.SqlExceptionHelper : 
se "biblio_db" has a collation version mismatch
```

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.

The screenshot shows a VS Code editor window with the file explorer on the left and the code editor in the center. The file explorer shows the project structure with folders like 'PARTIEFRONTEND', 'node_modules', 'public', 'src', 'components', 'pages', 'styles', and 'App.js'. The code editor shows the 'Reservations.js' file with the following code:

```
src > pages > JS Reservations.js > [e] Reservations
1 // src/pages/Reservations.js
2 import React, { useState, useEffect } from 'react';
3 import axios from 'axios';
4 import ReservationForm from '../components/ReservationForm';
5 import ReservationTable from '../components/ReservationTable';
6
7 const Reservations = () => {
8   const [reservations, setReservations] = useState([]);
9   const [loading, setLoading] = useState(true);
10  const [error, setError] = useState('');
11
12  const fetchReservations = async () => {
13    try {
14      const response = await axios.get('/api/reservations');
15      setReservations(response.data);
16      setLoading(false);
17    } catch (err) {
18      setError(err.message);
19      setLoading(false);
20    }
21  }
22
23  useEffect(() => {
24    fetchReservations();
25  }, []);
26
27  return (
28    <div>
29      <ReservationForm />
30      <ReservationTable />
31    </div>
32  );
33}
```

The terminal output shows the following messages:

```
On Your Network: http://192.168.1.37:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
(node:22012) [DEP0060] DeprecationWarning: The `util._extend` API is deprecated. Please use Object.assign() instead.
```

The screenshot shows the 'Biblio-Hybride' website with a dark teal header. The header contains the logo and the text 'Biblio-Hybride' on the left, and navigation links 'Livres disponibles', 'Réserver', 'Suivi', and 'Réservations' on the right. The main content area has a light blue background. On the left, there is a section titled 'Réservations' with the subtitle 'Réserver un livre'. In the center, there is a white card titled 'Réserver un livre' with a form. The form has two input fields: 'Nom du client:' and 'Date de réservation:'. The 'Date de réservation:' field has a calendar icon. Below the input fields is a blue button labeled 'Réserver'. A tooltip message 'Veuillez renseigner ce champ.' is visible next to the 'Nom du client:' input field.

Réserver

Liste des réservations

ID	Nom du client	Date de réservation
1	Jean Dupont	03/05/2025
2	Fatou Ndiaye	05/05/2025
3	soda gueye	08/02/2025

Réserver un livre

Nom du client:

Date de réservation:

Réserver

✕ 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