

Sujet d'examen : Bibliothèque universitaire hybride (REST+SOAP)

Membres :

Khadim Faye (khadim.faye2@unchk.edu.sn / N01362520191)

Mame Diarra Lô (Mamediarra.lo2@unchk.edu.sn / N00982420192)

Fonctionnalités minimales attendues :

I. REST (Consultation* publique) :

1 /L'environnement de développement Eclipse Backend REST et SOAP : Spring Boot :

eclipse-workspace - ApiBiblio/src/main/java/com/bibliotheque/ApiBiblio/wsdl/SupprimerLivreRequest.java - Eclipse IDE

File Edit Source Refactor Source Navigate Search Project Run Window Help

Package Explorer X application... ApiBiblioApp... ApiBiblio/po... SupprimerLi... ModifierLiv... PreterLivreR... RetournerLi... LivreSoapEnd... 22

```

1 package com.bibliotheque.ApiBiblio.wsdl;
2
3 import jakarta.xml.bind.annotation.*;
4
5 @XmlAccessorType(XmlAccessType.FIELD)
6 @XmlType(name = "", propOrder = {
7     "id"
8 })
9 @XmlRootElement(name = "SupprimerLivreRequest", namespace = "http://www.bibliotheque.com/wsdl")
10 public class SupprimerLivreRequest {
11
12     @XmlElement(namespace = "http://www.bibliotheque.com/wsdl")
13     private long id;
14
15     public long getId() {
16         return id;
17     }
18
19     public void setId(long id) {
20         this.id = id;
21     }
22 }
23

```

Console X

```

ApibiblioApplication [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 avr. 2025, 06:10:02) [pid: 10420]
2025-04-30T06:25:53.385-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] .w.s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004,^
2025-04-30T06:25:53.429-12:00 WARN 10420 --- [Apibiblio] [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by
2025-04-30T06:25:53.528-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'.
2025-04-30T06:25:53.553-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 3
2025-04-30T06:25:53.569-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) wit
2025-04-30T06:25:53.578-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] c.b.Apibiblio.ApiBiblioApplication : Started ApibiblioApplication in 1.374
2025-04-30T06:25:53.581-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
2025-04-30T06:26:24.201-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[]/_ : Initializing Spring DispatcherS
2025-04-30T06:26:24.201-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.s.w.t.http.MessageDispatcherServlet : Initializing Servlet 'messageDispatche
2025-04-30T06:26:24.203-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.s.ws.soap.SaajSoapMessageFactory : Creating SAAJ 1.3 MessageFactory with
2025-04-30T06:26:24.211-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.s.w.t.http.MessageDispatcherServlet : Completed initialization in 9 ms

```

Console X

```

ApibiblioApplication [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (30 avr. 2025, 06:10:02) [pid: 10420]
2025-04-30T06:25:53.385-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] .w.s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004,^
2025-04-30T06:25:53.429-12:00 WARN 10420 --- [Apibiblio] [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by
2025-04-30T06:25:53.528-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'.
2025-04-30T06:25:53.553-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 3
2025-04-30T06:25:53.569-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) wit
2025-04-30T06:25:53.578-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] c.b.Apibiblio.ApiBiblioApplication : Started ApibiblioApplication in 1.374
2025-04-30T06:25:53.581-12:00 INFO 10420 --- [Apibiblio] [ restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
2025-04-30T06:26:24.201-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[]/_ : Initializing Spring MessageDispatcherS
2025-04-30T06:26:24.201-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.s.w.t.http.MessageDispatcherServlet : Initializing Servlet 'messageDispatche
2025-04-30T06:26:24.203-12:00 INFO 10420 --- [Apibiblio] [nio-8080-exec-1] o.s.ws.soap.SaajSoapMessageFactory : Creating SAAJ 1.3 MessageFactory with

```

2/La connexion de la Basse de Données H2 à notre Projet :

The screenshot shows the H2 Console interface running in a browser window. The URL is `localhost:8080/h2-console/login.do?jsessionid=94c9f7c7ebc4d66447dbcd09a8548f05`. The browser tabs include "H2 Console", "Write scripts to test API responses", "Comment tester votre première", and "da". Below the tabs, the address bar shows the same URL. The page includes a toolbar with icons for refresh, back, forward, and search, along with links to YouTube, Gmail, Maps, Actualités, Applications, and Traduire. The main area displays the database schema:

- jdbc:h2:mem:testdb
- LIVRE
- RESERVATION
- UTILISATEUR
- INFORMATION_SCHEMA
- Users

Version: H2 2.3.232 (2024-08-11)

Important Commands

	Displays this Help Page
	Shows the Command History
	Ctrl+Enter Executes the current SQL statement
	Shift+Enter Executes the SQL statement defined by the text selection
	Ctrl+Space Auto complete
	Disconnects from the database

Sample SQL Script

Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

Adding Database Drivers

3/ Les requêtes avec la Basse de Données H2 console :

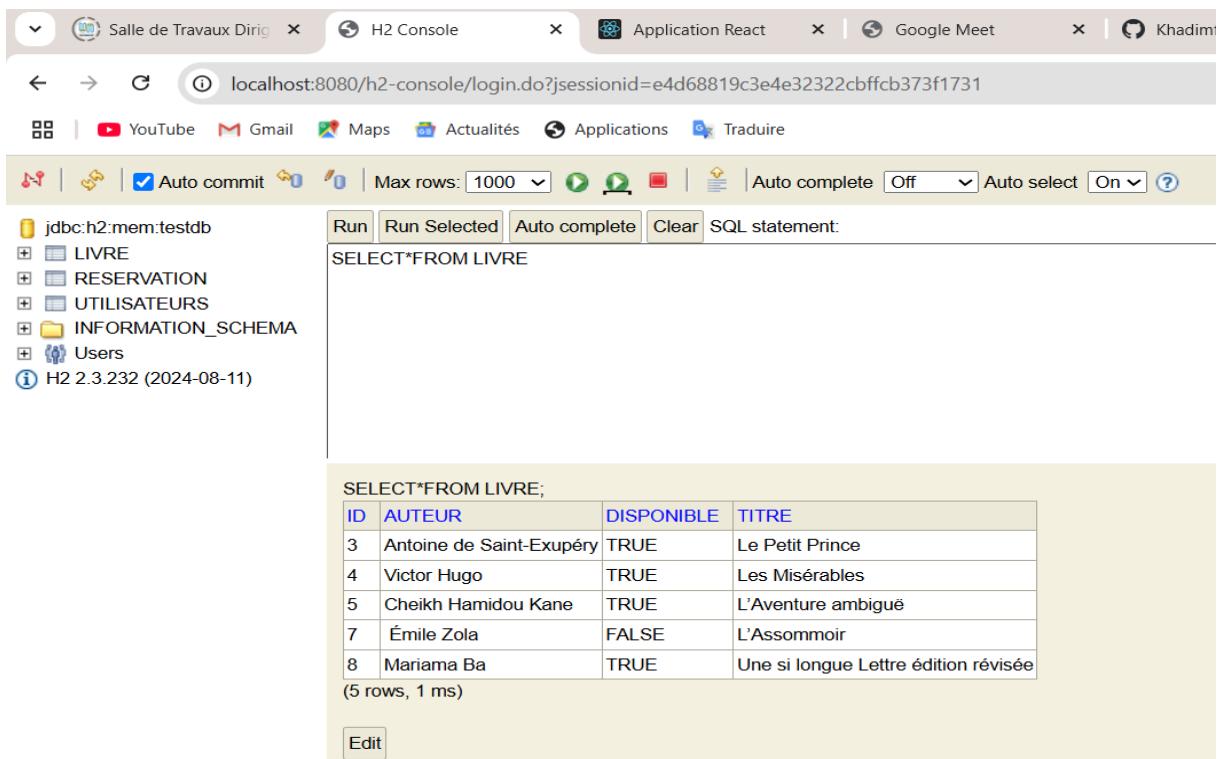
Ajouter un livre

SQL statement:
 INSERT INTO LIVRE (TITRE, AUTEUR, DISPONIBLE) VALUES ('Le Cavalier et son ombre', 'Boubacar Boris Diop', TRUE);

```

INSERT INTO LIVRE (TITRE, AUTEUR, DISPONIBLE) VALUES ('Le Cavalier et son ombre'
', 'Boubacar Boris Diop', TRUE);
Update count: 1
(1 ms)
  
```

Afficher les livres



Salle de Travaux Dirigé H2 Console Application React Google Meet Khadim

localhost:8080/h2-console/login.do?jsessionid=e4d68819c3e4e32322cbffcb373f1731

YouTube Gmail Maps Actualités Applications Traduire

jdbc:h2:mem:testdb LIVRE RESERVATION UTILISATEURS INFORMATION_SCHEMA Users H2 2.3.232 (2024-08-11)

Auto commit Max rows: 1000 Run Run Selected Auto complete Clear SQL statement:

```
SELECT*FROM LIVRE
```

ID	AUTEUR	DISPONIBLE	TITRE
3	Antoine de Saint-Exupéry	TRUE	Le Petit Prince
4	Victor Hugo	TRUE	Les Misérables
5	Cheikh Hamidou Kane	TRUE	L'Aventure ambiguë
7	Émile Zola	FALSE	L'Assommoir
8	Mariama Ba	TRUE	Une si longue Lettre édition révisée

(5 rows, 1 ms)

Edit

*Edit livre :

```
@edit SELECT*FROM LIVRE;
```

Action	ID	AUTEUR	DISPONIBLE	TITRE
	3	Antoine de Saint-Exupéry	TRUE	Le Petit Prince
	4	Victor Hugo	TRUE	Les Misérables
	5	Cheikh Hamidou Kane	TRUE	L'Aventure ambiguë
	7	Émile Zola	FALSE	L'Assommoir
	8	Mariama Ba	TRUE	Une si longue Lettre édition révisée

(5 rows, 1 ms)

Ajouter une réservation

Run Run Selected Auto complete Clear SQL statement:

```
INSERT INTO reservation (nom_utilisateur, titre_livre, date_reservation) VALUES ('Coumba Sarr', 'Le Cavalier et son ombre', '2025-07-01');
```

INSERT INTO reservation (nom_utilisateur, titre_livre, date_reservation) VALUES ('Coumba Sarr', 'Le Cavalier et son ombre', '2025-07-01');
Update count: 1
(0 ms)

Afficher les réservations

The screenshot shows the H2 Console interface with the following details:

- Database:** jdbc:h2:mem:testdb
- Tables:** LIVRE, RESERVATION, UTILISATEURS, INFORMATION_SCHEMA, Users
- Version:** H2 2.3.232 (2024-08-11)
- SQL Statement:** SELECT*FROM reservation WHERE id=1
- Result:**

ID	DATE_RESERVATION	NOM_UTILISATEUR	TITRE_LIVRE
1	2025-06-24	Jean Dupont	Le Petit Prince

(1 row, 1 ms)

*Edit réservation :

The screenshot shows the H2 Console interface with the following details:

- Action:** Edit
- Query:** @edit SELECT*FROM reservation WHERE id=1;
- Result:**

Action	ID	DATE_RESERVATION	NOM_UTILISATEUR	TITRE_LIVRE
	1	2025-06-24	Jean Dupont	Le Petit Prince

(1 row, 0 ms)

4/ Les requêtes avec Postman :

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

HTTP <http://localhost:8080/livres>

GET <http://localhost:8080/livres>

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "titre": "Germinal",
3   "auteur": " Émile Zola",
4   "disponible": true
5 }
6

```

Send

Body Cookies Headers (8) Test Results

200 OK 82 ms 692 B

	id	titre	auteur	disponible
0	3	Le Petit Prince	Antoine de Saint-Exupéry	true
1	4	Les Misérables	Victor Hugo	true
2	5	L'Aventure ambiguë	Cheikh Hamidou Kane	true
3	7	L'Assommoir	Émile Zola	false
4	8	Une si longue Lettre édition révisée	Mariama Ba	true

• ('GET /livres/{id}') : Afficher les informations d'un livre

HTTP <http://localhost:8080/livres/3>

GET <http://localhost:8080/livres/3>

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "titre": "Germinal",
3   "auteur": " Émile Zola",
4   "disponible": true
5 }
6

```

Send

Body Cookies Headers (8) Test Results

200 OK 18 ms 342 B

```

1 {
2   "id": 3,
3   "titre": "Le Petit Prince",
4   "auteur": "Antoine de Saint-Exupéry",
5   "disponible": true
6 }

```

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

HTTP <http://localhost:8080/livres/disponibles>

GET <http://localhost:8080/livres/disponibles>

Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

1 {
2   "titre": "Germinal",
3   "auteur": " Émile Zola",
4   "disponible": true
5 }
6

```

Body Cookies Headers (8) Test Results

200 OK 12 ms 613 B

	id	titre	auteur	disponible
0	3	Le Petit Prince	Antoine de Saint-Exupéry	true
1	4	Les Misérables	Victor Hugo	true
2	5	L'Aventure ambiguë	Cheikh Hamidou Kane	true
3	8	Une si longue Lettre édition révisée	Mariama Ba	true

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

HTTP <http://localhost:8080/reservations/6>

GET <http://localhost:8080/reservations/6>

Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

1 {
2   "nomUtilisateur": "Diarra Lo",
3   "titreLivre": "Germinal",
4   "dateReservation": "2025-06-27"
5 }
6

```

Body Cookies Headers (8) Test Results (2/4)

200 OK 6 ms 345 B

id	6
nomUtilisateur	Diarra Lo
titreLivre	Germinal
dateReservation	2025-06-27

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

The screenshot shows the SoapUI interface with a POST request to <http://localhost:8080/reservations>. The request body is a JSON object:

```

1  {
2   "nomUtilisateur": "Diarra Lo",
3   "titreLivre": "Germinal",
4   "dateReservation": "2025-06-27"
5 }
6

```

The response status is 200 OK, with a response time of 22 ms and a size of 299 B. The response content is: "Réservation enregistrée avec succès (ID: 6)".

II. SOAP (Gestion interne) :

5/Les requêtes lors du Test avec SOAPUI :

- AjouterLivre(Livre livre)

The SoapUI Start Page shows the Request 1 configuration for the AjouterLivre service. The Request Properties table includes:

Property	Value
Name	Request 1
Description	
Message Size	467
Encoding	UTF-8
Endpoint	http://localhost:8080/ws

The Request Body XML is:

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsdl="http://www.bibliotheque.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <wsdl:AjouterLivreRequest>
      <wsdl:titre>L'Étranger</wsdl:titre>
      <wsdl:auteur>Albert Camus</wsdl:auteur>
      <wsdl:disponible>true</wsdl:disponible>
    </wsdl:AjouterLivreRequest>
  </soapenv:Body>
</soapenv:Envelope>

```

The Response XML is:

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <AjouterLivreResponse>
      <livre>
        <id>1</id>
        <titre>L'Étranger</titre>
        <auteur>Albert Camus</auteur>
        <disponible>true</disponible>
      </livre>
    </AjouterLivreResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

- ModifierLivre(Long livreID, Livre livre)

The screenshot shows the SoapUI interface with the following details:

- Projects:** SoapBiblio, BibliothèquePortSoap11
- Request 1:** A successful modification of a book.
- Request Properties:**

Property	Value
Name	Request 1
Description	
Message Size	506
Encoding	UTF-8
- Response Headers:** Auth, Headers (0), Attachments (0), WS-A, WS-RM, JMS Headers, JMS Property (0)
- Response Content:**

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsdl="http://www.bibliothèque.com/wsdl">
  <soapenv:Header/>
  <soapenv:Body>
    <ns3:ModifierLivreRequest>
      <wsdl:id>1</wsdl:id>
      <wsdl:titre>Nouvelle édition</wsdl:titre>
      <wsdl:auteur>Albert Camus</wsdl:auteur>
      <wsdl:disponible>true</wsdl:disponible>
    </ns3:ModifierLivreRequest>
  </soapenv:Body>
</soapenv:Envelope>

```

• SupprimerLivre(Long livreID)

The screenshot shows the SoapUI interface with the following details:

- File:** Empty, SOAP, REST, Import, Save All, Forum, Trial, Preferences, Proxy, Endpoint Explorer
- Search Forum:** Search bar
- Projects:** BibliothèqueSOAP, LivresPortSoap11
- Request 1:** A successful deletion of a book.
- Response Content:**

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsdl="http://www.bibliothèque.com/wsdl">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:SupprimerLivreRequest>
      <ws:id>6</ws:id>
    </ws:SupprimerLivreRequest>
  </soapenv:Body>
</soapenv:Envelope>

```

• PreterLivre(Long userID, Long livreID)

The screenshot shows the SoapUI interface with the following details:

- Projects:** SoapBiblio / BibliothèquePortSoap11
- Request 1:** A successful request to `http://localhost:8080/ws` with status `OK`.
- Request Properties:** Request 1, Value.
- Response Headers:** response time: 70ms (235 bytes)
- Response Body (Raw XML):**

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<FrerLivreResponse>
<message>Livre prêté avec succès</message>
</FrerLivreResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- RetournerLivre(Long userID, Long livreID)

The screenshot shows a sequence of three requests in SoapUI, all labeled "Request 1".

- Request 1 (Left):** A RetournerLivre request. The XML payload is:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.bibliotheque.com/wsdl">
<soapenv:Header/>
<soapenv:Body>
<ws:RetournerLivreRequest>
<ws:id>l</ws:id>
</ws:RetournerLivreRequest>
</soapenv:Body>
</soapenv:Envelope>
```

- Request 2 (Middle):** A RetournerLivreResponse message. The XML payload is:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<RetournerLivreResponse>
<message>Livre retourné avec succès</message>
</RetournerLivreResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- Request 3 (Right):** A RetournerLivre request. The XML payload is identical to the first one.

Request Properties:

Property	Value
Name	Request 1
Description	
Message Size	310
Encoding	UTF-8

Response Headers:

Header	Value
response-time	119ms (243 bytes)

6/L'utilisation du Frontend React pour (Ajouter, Modifier, Supprimer, Prêter et Retourner) des Livres :

A screenshot of a web browser window showing multiple tabs. The active tab is 'Application React' at localhost:3000, displaying a library management application with a table of books and edit/delete buttons. Other tabs include 'Salle de Travaux Diric', 'H2 Console', 'Google Meet', 'Khadimf/ProjetWeb', 'Google Meet', and 'Erreurs push Git main'. The browser interface includes a back/forward button, search bar, and various icons.

Gestion de bibliothèque

Liste des livres

Ajouter un livre				
IDENTIFIANT	Titre	Auteur	Disponibilité	Actes
3	Le Petit Prince	Antoine de Saint-Exupéry	Disponible	<button>Modificateur</button> <button>Supprimer</button> <button>Prêter</button> <button>Retourner</button>
4	Les Misérables	Victor Hugo	Disponible	<button>Modificateur</button> <button>Supprimer</button> <button>Prêter</button> <button>Retourner</button>
5	L'Aventure ambiguë	Cheikh Hamidou Kane	Disponible	<button>Modificateur</button> <button>Supprimer</button> <button>Prêter</button> <button>Retourner</button>
7	L'Assommoir	Émile Zola	Prêté	<button>Modificateur</button> <button>Supprimer</button> <button>Prêter</button> <button>Retourner</button>
8	Une si longue Lettre édition révisée	Mariama Ba	Disponible	<button>Modificateur</button> <button>Supprimer</button> <button>Prêter</button> <button>Retourner</button>

💡 Retours d'expérience

Ce projet nous a permis de mettre en pratique l'intégration d'un backend SOAP développé avec Spring Boot à un frontend React. Il nous a apporté une compréhension plus approfondie du fonctionnement des services web SOAP et de la manière dont ils peuvent être consommés depuis une interface moderne. Nous avons également appris à manipuler des outils comme SoapUI pour tester les endpoints SOAP, et à gérer les échanges de données XML à l'aide de fichiers XSD.

La transition entre le backend Java et le frontend JavaScript a renforcé notre capacité à travailler dans des environnements hétérogènes, à gérer les communications HTTP (via un proxy Express), et à organiser efficacement un projet en couches — modèle, service, contrôleur et endpoint.

⚠ Difficultés rencontrées

1. Gestion des namespaces SOAP

- L'une des erreurs les plus fréquentes concernait le nommage incorrect ou l'absence de namespace dans les requêtes SOAP.

- Solution : Nous avons soigneusement défini le targetNamespace dans le fichier XSD et veillé à ce qu'il soit correctement repris dans les annotations Java des classes générées et des endpoints.

2. Adaptation des points de terminaison (endpoints)

- Spring Web Services ne reconnaissait pas certains endpoints à cause de méthodes mal annotées ou de noms de requêtes erronés.

- Solution : Nous avons systématiquement utilisé @Endpoint combiné avec @PayloadRoot, en précisant le bon namespace et le localPart correspondant aux requêtes.

3. Consommation du backend SOAP avec React

- Étant donné que React ne traite pas nativement le XML, il a été nécessaire de créer un proxy côté serveur.

- Solution : Nous avons mis en place un serveur Express (index.js) en Node.js pour faire la conversion entre les requêtes SOAP XML et des appels REST/JSON exploitables par le frontend React.

4. Réservations et relations entre entités :

- La fonctionnalité de réservation a exigé une gestion rigoureuse des relations entre entités telles que Livre, Utilisateur et Réservation.

- Solution : Nous avons utilisé des classes DTO pour organiser proprement les données échangées, et défini des services métier clairs pour centraliser la logique de réservation.

5. Erreurs HTTP 500 liées à JAXB

- Plusieurs erreurs critiques provenaient de la non-conformité au schéma XSD ou de l'absence de classes JAXB.

- Solution : Nous avons corrigé le fichier XSD, régénéré les classes à l'aide de jaxb2-maven-plugin, et ajusté les requêtes XML pour respecter scrupuleusement la structure définie.

SFINS