Application Web de Gestion de Bibliothèques Universitaires

Septembre 2025

Consignes préparatoires : Ce projet a pour objectif de vous placer dans une situation proche d'un contexte professionnel. Chaque groupe devra travailler en collaboration, définir clairement les rôles de chacun et progresser de manière régulière tout au long du semestre. Le sujet étant volontairement ouvert, il est attendu des étudiants qu'ils fassent preuve d'initiative et de créativité afin de mobiliser et mettre en valeur l'ensemble de leurs connaissances et compétences dans la réalisation d'un projet complet. Attention, les fonctionnalités demandées devront être impérativement implémentées.

1 Contexte

Le projet porte sur la gestion d'une, ou des **bibliothèques universitaires** (BU). Dans ce contexte, des étudiants ont la possibilité de consulter ou d'emprunter des ouvrages (livres, revues, etc.) pour une durée maximale de trois semaines. Pour assurer le bon fonctionnement de ce service d'emprunt, un administrateur en charge de la BU surveille les activités des étudiants et peut changer les règles de la BU.

L'application finale sera développée à l'aide de web services, en respectant une architecture basée sur **SpringBoot**. L'authentification des utilisateurs (étudiant ou administrateur) devra être implémenté avec **SpringSecurity** et **JWT** (JSON Web Token). L'interface utilisateur devra être dynamique ; et pourra être implémentée avec **JSP** (JavaServer Pages), **JavaFX** ou n'importe quelle autre technologie vue pendant votre parcours.

Remarque importante : Les données pourront être stockées de n'importe quelle manière, mais devront rester persistantes entre deux utilisations de l'application.

2 Objectifs

L'objectif du projet est de créer une application web qui permet :

- La gestion des utilisateurs : Un étudiant peut s'inscrire, sous la condition qu'il possède un numéro étudiant valide (7 digits, unique). L'administrateur a le pouvoir de modifier ou supprimer un utilisateur s'il le souhaite.
- La gestion des emprunts : Un étudiant peut faire la demande d'emprunt d'un ouvrage disponible. L'administrateur doit ensuite valider manuellement cette demande. Attention, un étudiant n'a le droit que d'emprunter quatre ouvrages simultanément. De plus, un emprunt ne peut pas durer plus de trois semaines. Si cette date est dépassée, l'administrateur peut envoyer une alerte à l'utilisateur concerné. Cependant, cette période d'emprunt peut être étendue si l'étudiant en fait la demande avant la date limite de retour. Cette extension ne peut être faite qu'une seule fois par emprunt. Finalement, un historique d'emprunt est associé à chaque étudiant.
- La gestion des stocks des ouvrages : Le catalogue des livres disponibles doit être consultable par tous, même sans connexion. L'affichage peut être filtré selon différents critères (genre, auteur, date de parution, etc.). L'administrateur a la possibilité d'ajouter de nouveaux ouvrages au stock ou d'en supprimer. Chaque exemplaire d'un ouvrage ne peut être emprunté qu'une seule fois à la fois.
- La gestion des authentifications doit être faite pour empêcher un utilisateur d'outrepasser ses permissions. Chaque étudiant a un espace privé, que seul lui et l'administrateur peut consulter.
- Une interface utilisateur claire : L'UI/UX de l'application doit la rendre facile d'utilisation et agréable. Elle doit donc être dynamique et réactive.

3 Architecture générale

3.1 Front-End

Le choix de la technologie pour développer le front-end de l'application est libre. Il est cependant fortement recommandé d'utiliser une technologie vue en cours (**JavaFX, JSP**), ou avec laquelle vous êtes à l'aise et autonome. L'aspect principal de ce projet est le développement de web services, le front-end ne doit donc pas représenter la majorité de votre production.

3.2 Back-End

L'utilisation de services interfacés par des **APIs REST** est obligatoire. Les services seront développés et gérés à l'aide du framework **SpringBoot**, afin de gérer les requêtes HTTPS et les sessions utilisateur.
L'ensemble des données manipulées par l'application (informations relatives aux utilisateurs, catalogue des ouvrages, gestion des emprunts) devra être stocké de manière **persistante**. Plusieurs approches sont possibles :

- L'utilisation d'une base de données relationnelle (par ex. MongoDB), permettant de garantir l'intégrité et la pérennité des données.
- Un stockage en mémoire (par ex. Map, List) avec sérialisation régulière dans des fichiers au format `JSON`, afin d'assurer une persistance entre deux exécutions de l'application.

3.3 Authentification

L'application doit implémenter l'authentification par **JWT** avec les dernières versions de **SpringBoot** et **SpringSecurity**.

4 Fonctionnalités à implémenter

L'intégralité des fonctionnalités suivantes devra être implémentée. Les technologies pour le faire sont exposées plus haut dans la section Architecture générale.

4.1 Gestion des étudiants

- **Inscription** : Implémentez une page permettant à un utilisateur de se créer un compte. Le compte sera forcément de type étudiant, le compte administrateur sera lui un compte créé au premier démarrage de l'application. L'étudiant devra renseigner son email, un mot de passe, et un numéro étudiant. Le numéro d'étudiant devra être valide.
- **Connexion** : Implémentez une page permettant à un utilisateur de se connecter. Un formulaire demandera les informations basiques (email, mot de passe).
- **Tableau de bord** : Implémenter une page pour que l'utilisateur connecté ait accès à ses emprunts en cours, et son historique d'emprunts. Au sujet des emprunts en cours, une demande d'extension doit pouvoir être demandée, conformément aux règles de la section Objectifs. Il doit aussi être possible pour l'étudiant de déclarer un ouvrage retourné à la bibliothèque. Cette déclaration devra être validée par l'administrateur. De plus, une liste de message lui étant adressé devra apparaitre clairement sur son tableau de bord.
- **Parcourir et emprunter des ouvrages** : Un catalogue d'ouvrages disponibles dans la ou les bibliothèques doit être consultable dans une page dédiée. Une première version devra afficher la liste de tous les ouvrages, et une deuxième permettra de filtrer cette liste selon différent critère. Si les conditions le permettent, cette page doit aussi permettre à l'utilisateur courant de demander un emprunt.

4.2 Gestion de l'administrateur

- **Connexion** : En utilisant la même page de connexion proposée plus haut, l'administrateur de la BU doit pouvoir se connecter.
- **Tableau de bord** : Implémenter une page pour que l'administrateur ait une vue des différents emprunts en cours. Un filtre par nom pourra être appliqué dans une version future. Pour chaque réservation, un message pourra être automatiquement généré, puis envoyé, aux étudiants dont l'emprunt arrive à son terme. De plus, une liste des étudiants doit aussi être accessible depuis le tableau de bord.
- **Historique** : Pour un étudiant donné, l'administrateur doit pouvoir avoir accès à l'intégral de son historique de réservations. Une page, accessible depuis la liste des étudiants du tableau de bord, devra être développée.
- **Réponse des demandes** : L'administrateur doit pouvoir gérer les différentes demandes des autres utilisateurs (emprunt ou retour). Une page dédiée devra être accessible pour l'administrateur seulement, permettant d'accepter ou refuser les demandes.
- **Modifier le catalogue** : Une page permettant d'ajouter, ou de supprimer, un exemplaire d'un ouvrage de la bibliothèque devra être accessible par l'administrateur.

- **Modifier les informations de la BU** : Pour une bibliothèque donnée, les informations utiles pourront être modifiées (horaires et dates d'ouverture, adresse, etc.).

4.3 Gestion de la bibliothèque

- **Accès aux informations relatives à la BU** : Une page contenant les informations pratiques (horaires et dates d'ouverture, adresse, etc.) devra être accessible depuis n'importe quel autre page.
- **Catalogue** : Une liste d'ouvrages (disponible ou non) devra être accessible depuis n'importe où, même sans connexion.

5 API et interfaces

Vous devrez définir vous-même les end-points REST pour respecter les fonctionnalités demandées. Un rapport final des points d'entrée devra être remis sous la forme d'une spécification OpenAPI ou Swagger.

6 Déroulement

Dans ce projet, vous allez devoir faire preuve de professionnalisme. Vous allez devoir scinder votre projets en lots distincts. Pour chaque lot, vous devrez établir : un descriptif fonctionnel du lot, un plan de tests, une date de livraison. Au dernier lot, l'application devra donc être complète.

Petit conseil : vous comprendrez qu'un certain nombre de technologies nécessaires à l'élaboration de ce projet sont en cours d'apprentissage. Par exemple, Spring Security arrive assez tard dans le programme du module de Web Services. Il faudra donc établir vos lots en fonction de ces paramètres.

De même pour le test structurel spécifique aux services REST. On ne demande pas d'automatisation obligatoire de vos tests fonctionne, mais des preuves comme quoi, vous les avez faits. Tout ceci devra être documenté à chaque livraison. Vous devrez à minima avoir trois lots distincts. Vous devrez annoncer votre planning prévisionnel avant le 30 septembre.

7 Livrables

- Pour chaque livraison, **un document décrivant le lot** et la **qualification du lot** (fonctionnel / structurel) devra être fourni. Le **code source et un rapport des tests** devront aussi être fourni. Toute documentation supplémentaire sera appréciée.
- **Rendu final**
  - **Un rapport** contenant l'histoire de votre projet et vos choix de conceptions
  - **Une spécification de vos points d'entrée** REST (OpenAPI ou Swagger)
  - **Le code source** final de l'application

8 Note sur les outils

L'utilisation d'outils utiles au développement de l'application (par ex. versionnage, suivi de ticket, diagramme de Gantt, etc.) est fortement encouragé. Cette liste d'outils pourra être présentée dans le rapport final. L'utilisation d'IA générative n'est pas prohibée. Mais attention ! Chaque membre de l'équipe doit être capable d'expliquer et reproduire l'intégralité du code.

9 Évaluation du projet

L'évaluation du projet reposera à la fois sur les livrables produits et sur une présentation finale devant l'enseignant.
Les critères d'évaluation principaux sont les suivants :

- **Respect de la spécification** : conformité à l'architecture définie, gestion correcte des utilisateurs, respect des contraintes de réservations, etc.

- **Gestion du projet** : organisation du travail en équipe, répartition des rôles, respect du calendrier et progression régulière.
- **Qualité du code** : structure, lisibilité, respect des bonnes pratiques Java/Spring, ainsi que la présence et la qualité des tests structurels (obligatoires).
- **Interface utilisateur** : simplicité d'utilisation, ergonomie et cohérence générale.

À l'issue de la présentation finale, chaque équipe passera également un entretien individuel avec l'enseignant. Une note globale de projet sera attribuée à l'ensemble du groupe. Toutefois, dans le cas d'une participation inégale entre les membres de l'équipe, un **coefficient individuel** (compris entre 0 et 1.4) pourra être appliqué. La note finale individuelle sera alors calculée comme suit :

$$\text{Note individuelle} = \text{Note du projet} \times \text{Coefficient individuel}$$

10 Conclusion

Ce projet vous permettra de manipuler des technologies récentes comme SpringBoot, SpringSecurity avec JWT, tout en construisant une application web avec JSP pour le front-end. Vous travaillerez également sur la conception d'un service RESTful tout en respectant les contraintes de sécurité et de gestion des rôles. La qualité sera aussi un axe d'évaluation important. Vos tests seront importants.

11 Groupes

`https://docs.google.com/spreadsheets/d/12O5Q7oW6nBWIHkO6QNVN8vKCcOVQScozSibvQzOMwN4/edit?usp=sharing`