AP 3.2 : Gestion de la bibliothèque SIO

Le BTS SIO dispose d'une bibliothèque de revues informatiques dont l'objectif est d'être mises à disposition des étudiants avec la possibilité de les emprunter pour les consulter.
Actuellement, ces revues sont stockées dans un local accessible uniquement aux professeurs, et ne sont pas encore proposées aux étudiants.
Pour aider à la gestion de cette bibliothèque, on souhaite développer une application qui permettrait de gérer l'alimentation du fonds, ainsi que le suivi des emprunts et des retours.

L'application à développer doit respecter un ensemble de règles de gestion, et réaliser assez grand nombre de fonctionnalités. Elle devra également proposer un niveau de qualité de code et de sécurité acceptables, et, pour finir, être disponible rapidement de manière à pouvoir ouvrir cette bibliothèque aux prêts prochainement !
Pour respecter toutes ces contraintes, il a donc été décidé de développer cette application en respectant certains principes d'agilité :

- L'application sera développée pas à pas, les fonctionnalités seront ajoutées progressivement. On commencera avec un Produit Minimum Viable (*MVP*), permettant de commencer à utiliser l'application, qui sera progressivement complété avec des fonctionnalités supplémentaires.
- Le développement sera fait par étapes, et à l'issue de chacune d'elle, une nouvelle version utilisable de l'application sera disponible.
- Les versions devront toujours respecter des critères de qualité de code et de sécurité, quelle que soit l'étape.

Vous êtes choisis pour développer cette application, avec les contraintes suivantes :

1. <u>Cahier des charges technique :</u>
    - La solution est divisée en un back-end et un front-end
    - Le back-end doit être développé avec le framework Spring Boot
    - Les données sont hébergées sur une base de données gérée par Mysql
    - La communication avec la base de données doit se faire avec JPA
    - Chaque cas d'utilisation doit être testé (unitaire ou intégration)
    - Un début de front-end répondant aux demandes de l'étape 1 a été développé par une autre équipe, et vous sera fourni. Votre back-end et ce front-end devront être compatibles

2. Gestion de projet :

La plateforme GitHub sera utilisée pour héberger le code, et assurer la gestion de projet.

Un projet a été créé sur cette plateforme. Il contient trois étapes (“*milestones”*), elles-mêmes découpées en tâches (“*issues*”) “macroscopiques”, qui représentent les fonctionnalités souhaitées, c'est-à-dire le cahier des charges du projet.
Un contexte fourni en annexe décrit le fonctionnement souhaité de l’application pour chaque étape et permet de mieux comprendre le cahier des charges.

Vous utiliserez le projet GitHub fourni en respectant la procédure suivante :

- suivre impérativement les étapes dans l’ordre fixé. Ne pas traiter l'étape 2 avant que l’étape 1 ne soit finie.
- découper les tâches “macro” en tâches “micro”
- essayer de respecter les délais prévisionnels affectés à chaque étape :
    - étape 1 : 2 semaines,
    - étape 2 : 3 semaines
    - étape 3 : 2 semaines. *On peut parler de “sprint” pour ces périodes*.

Au début de chaque étape : à partir de la description des tâches contenues dans l'étape, modélisez les données **métier** manipulées, en utilisant le formalisme de votre choix (MEA, diagramme de classes), et en faisant particulièrement attention aux liens entre les entités/classes. Faites valider ce modèle par un professeur pour pouvoir créer les classes conformément au modèle, puis continuer le code : vous adapterez les classes définies dans le frontend, et coderez entièrement le backend.

Pour changer d’étape : appelez un professeur *(qui est un représentant des commanditaires du projet, rôle qu’on appelle typiquement le Product Owner ou PO)* et faites lui une démonstration de votre application, puis répondez à ses questions. Selon les remarques faites par le PO, créez de nouvelles tâches, et insérez les :

- soit dans l’étape suivante (si le PO valide votre travail et vous donne le go pour passer),
- soit dans l’étape courante (si votre travail n’est pas validé et demande trop de retouches).

Par ailleurs, le projet contient trois colonnes : Backlog, In Progress et Done, correspondant à trois niveaux d’avancement. Lorsque vous travaillez sur une tâche, elle doit être inscrite dans la colonne In Progress, puis passer dans la colonne Done lorsque vous avez fini. Chaque tâche doit être affectée/assignée à un membre du groupe.

Ressources fournies :

- Ce document
- Un template de projet GitHub à importer pour créer votre projet (tableau d'issues)
- Un template de repository GitHub contenant des issues, des labels et des milestones
- Un front-end

Prise en main du projet :

- Rejoignez la Classroom GitHub en cliquant sur le lien fourni dans commun
- Au moment où GitHub Classroom le demande, le chef du groupe crée une Team, le binôme rejoint cette Team une fois qu'elle a été créée
- Lorsque le nouveau repository est créé :
    - Allez dans l'onglet Projects et créez un nouveau projet à partir du template [TEMPLATE] BiblioSIO
    - Ouvrez dans une autre fenêtre le repository https://github.com/Lycee-Gustave-Eiffel-Bordeaux-SIO/BiblioSio-template et allez dans l'onglet Issues
    - Dans le repository de votre équipe, allez dans l'onglet Issues
    - Copiez un par un et **dans cet ordre** chaque Milestone, Label puis Issue du repository BiblioSIO-Template dans votre propre repository. **Lorsque vous copiez des Issues**, faites bien **attention à bien assigner les labels et milestones correspondant**

Livrables :

- Le code du projet sur un repository GitHub
- Les *issues* du projet *(présentes dans le repository GitHub)*, qui doivent contenir votre découpage en tâches "micro", la répartition des tâches *(en passant par le mécanisme d'assignation des tâches)* et votre progression sur le projet *(visible en temps réel par les commentaires automatiques dans les issues)*
- Eventuellement une documentation technique *(réalisée avec Swagger et Javadoc comme vous le verrez en cours)*
- **Échéance : le vendredi 22 décembre à 17h**

Annexes :

Fonctionnement de l'application par étapes :

Étape 1 : MVP
L'application permettra aux emprunteurs de consulter le fonds (les revues et exemplaires enregistrés dans la bibliothèque) sans avoir besoin de manipuler physiquement les exemplaires. Elle permettra de voir les revues disponibles *(ex. : Linux Magazine, Magazine Programmez!, etc.)* et de voir les exemplaires de ces revues *(ex. : les n°97, 98 et 99 de la revue Programmez!, ou les n°103, 104 et 105 de Linux Pratique, etc.)*. Les exemplaires contiennent des articles *(ex. : Magazine Programmez! n°98 -> article n°1 "La gestion des droits Linux", article n°2 "Nouveautés de Java 20", article n°3 "WSL2 avec Windows 11", etc.)*. On peut consulter les articles d'un exemplaire en le sélectionnant dans l'application.

Certains jours (ou à la demande d'étudiants), les professeurs pourront ouvrir la bibliothèque et permettre l'emprunt d'exemplaires aux emprunteurs. La date de début de l'emprunt est enregistrée, et l'emprunt est marqué comme "En cours".

Les emprunteurs peuvent rendre des emprunts (en ramenant les exemplaires empruntés). La date de retour de l'emprunt est alors enregistrée, et l'emprunt est marqué comme "Terminé".

Étape 2 : Fonctionnalités supplémentaires

- **authentification des utilisateurs** : il faudra se connecter pour accéder à l'application. Cela nécessite la création de comptes utilisateurs : un utilisateur est identifié par un login (unique) et un mot de passe, et possède un rôle (user ou admin).
  Il faudra distinguer les concepts de :
    - utilisateur : concept technique nécessaire à la phase de connexion et vérification des droits,
    - emprunteur : ensemble de données représentant l'utilisateur pouvant faire des emprunts.

  Un utilisateur est lié à un et un seul emprunteur.

- **demandes d'emprunt à distance** : avant de venir emprunter, un emprunteur peut faire une demande d'emprunt pour un exemplaire. Ces demandes d'emprunts peuvent être validées au moment de l'emprunt, et se transforment alors automatiquement en emprunt. Elles peuvent aussi être rejetées.

- **Contraintes d'emprunt** : pour empêcher de repartir avec toute la bibliothèque chez soi, le nombre d'emprunts simultanés est limité à 2 : il est donc impossible pour un emprunteur de faire un nouvel emprunt s'il en a déjà 2 en cours.

  De plus, la durée maximale d'un emprunt est de 21 jours. Au-delà, l'emprunt est marqué comme "En retard".

- **Informations sur les emprunteurs** : On souhaite avoir quelques informations supplémentaires sur les emprunteurs : pour les étudiants, on veut connaître leur promo (année où ils doivent être diplômés) et leur classe (SIO1A, 1B, 2A, 2B). Si un emprunteur est un professeur, on lui attribue simplement la classe PROF. Un professeur n'a pas de promo.

- **Règles de tri pour l'affichage de l'application** (affichage des emprunts, revues, etc.). L'algèbre relationnelle étant très performante pour ce genre d'opérations, on souhaite pouvoir demander les informations triées au backend, puis les afficher directement dans le front-end *(ce comportement changera peut-être dans le futur si la quantité de données transmise est trop grande, on fera alors le tri côté front-end)*

Étape 3 : Administration

On souhaite tirer profit des rôles attribués aux utilisateurs en mettant en place une **gestion des droits basée sur le rôle *(RBAC)***.

A chaque rôle sont associées les actions suivantes :

1. les users peuvent :
   - visualiser les exemplaires disponibles, en sélectionnant un titre de revue
   - visualiser les informations de leur compte. (info personnelles, nombre d'emprunts en cours, nombre de retards)
   - modifier leurs informations personnelles (nom, prénom, adresse mail)
   - faire une demande d'emprunt d'un exemplaire
   - visualiser les demandes d'emprunt en cours, ou passées.

2. les admin peuvent :
   - enregistrer de nouvelles revues, exemplaires et articles
   - gérer les comptes utilisateurs avec leurs infos personnelles
   - valider les demandes d'emprunt, enregistrer les emprunts et les retours d'exemplaires par les utilisateurs
   - visualiser la liste des emprunts en retard, la liste des utilisateurs ayant des retards, la liste des exemplaires empruntés, la liste des emprunts par classe, la liste des retards par classe

- envoyer des mails de relance à un ou plusieurs utilisateurs ayant des emprunts en retard
- bloquer la capacité d'emprunt d'un utilisateur, suite à un retard
- rétablir la capacité d'emprunt d'un emprunteur suite à un retard

Si un emprunteur a un emprunt en retard, l'emprunteur perd sa capacité d'emprunt : il ne peut plus emprunter ou faire de demande d'emprunt. Un admin peut bloquer la capacité d'emprunt d'un emprunteur, ou la rétablir.

On souhaite aussi pouvoir envoyer des notifications et des mails aux emprunteurs :
- Notifications quand la bibliothèque est ouverte (et les emprunts disponibles)
- Notifications à 19 et 21 jours d'emprunt
- Mails en cas de retard d'emprunt

- Un exemplaire emprunté est indisponible tant qu'il n'est pas rendu, et ne peut pas être emprunté. Dans le futur, d'autres statuts pourront être ajoutés (perdu, etc.)
- Lors d'un update ou d'une insertion, on ne peut pas modifier la valeur d'une date par une valeur antérieure au moment présent

Listes des actions autorisées pour chaque rôle : *(peut être utile pour vos tests, ou pas)*

- Utilisateur non-connecté :
  - ne peut rien voir ou faire, reste bloqué sur une page de connexion

- User :
  - peut faire une demande d'emprunt d'un exemplaire
  - ne peut pas faire de demande pour plus de deux exemplaires simultanément
  - ne peut pas faire de demande si son compte est bloqué
  - reçoit une notification au bout de 21 jours d'emprunt d'un exemplaire
  - peut visualiser la liste de toutes les revues :
    - peut choisir de trier la liste pour ne voir que certaines revues (de 1 à n revues)
  - peut visualiser la liste de tous les exemplaires :
    - peut trier la liste des exemplaires par ordre croissant ou décroissant du numéro ou de la date de parution
    - peut afficher les exemplaires pour certaines années seulement (de 1 à n années)
  - peut visualiser les informations de son compte
  - peut modifier son nom, prénom et adresse mail

- Admin :
    - peut ajouter, modifier ou supprimer une revue :
        - pas de doublons
        - si n'existe pas, modification impossible
        - si n'existe pas, suppression impossible
    - peut ajouter, modifier ou supprimer un exemplaire d'une revue :
        - pas de doublons
        - si n'existe pas, modification impossible
        - si n'existe pas, suppression impossible
    - peut ajouter, modifier ou supprimer un article d'un exemplaire :
        - pas de doublons
        - si n'existe pas, modification impossible
        - si n'existe pas, suppression impossible
    - peut créer, modifier ou supprimer des emprunteur :
        - pas de doublons
        - si n'existe pas, modification impossible
        - si n'existe pas, suppression impossible
    - peut créer, modifier ou supprimer des comptes utilisateurs :
        - pas de doublons
        - si n'existe pas, modification impossible
        - si n'existe pas, suppression impossible
    - peut bloquer et débloquer un emprunteur
    - peut valider une demande d'emprunt d'un emprunteur
    - peut créer un emprunt pour un emprunteur :
        - création impossible si emprunteur a déjà deux emprunts en cours
    - peut enregistrer un retour d'un exemplaire par un emprunteur
    - peut visualiser la liste des emprunteurs :
        - par classe (1 à n classes)
        - par ordre croissant ou décroissant du nom de famille
    - peut visualiser la liste des emprunts :
        - par revue
        - par utilisateur
        - par classe
    - peut visualiser la liste des demandes d'emprunts :
        - par revue
        - par emprunteur
        - par classe
    - peut visualiser la liste des emprunts en retard :
        - peut trier cette liste par classe
    - peut envoyer des mails aux emprunteurs ayant des emprunts en retard
    - peut faire tout ce qu'un User peut faire