Contexte : MediaTek86
Application : MediatekDocuments

Missions à réaliser

Le projet revient à votre équipe de développeurs, gérée par un responsable qui a déterminé les différentes missions à effectuer.

Remarques générales

Votre responsable retient votre attention sur plusieurs aspects :

- Le code ajouté doit respecter la structure en couches et la logique de code de l'application actuelle.
- Les manipulations des utilisateurs doivent être sécurisées, comme c'est le cas dans l'application actuelle (par exemple, en mettant certains objets graphiques en lecture seule).
- Les requêtes SQL doivent éviter les actes malveillants.
- Les données correspondant à des valeurs prédéfinies ne doivent pas être saisies mais sélectionnées dans une liste.

Mission 1 : gérer les documents (8h)

L'application doit permettre d'ajouter, modifier ou supprimer des documents, sous certaines conditions. Les documents concernés sont les livres, les DVD et les revues.
L'ajout d'un document ne suppose pas forcément une commande, même s'il intervient classiquement au moment d'une commande. Dans cette mission, on ne s'occupe pas encore des commandes.
La modification d'un document ne peut pas porter sur son numéro.
La suppression d'un document n'est possible que s'il ne possède aucun exemplaire ni commande. Cela permet d'éviter les accidents de suppression.

Mission 2 : gérer les commandes (12h)

Tâche 1 : gérer les commandes de livres ou de DVD (8h)

Il doit être possible de commander un ou plusieurs exemplaires d'un livre ou d'un DVD et de suivre l'évolution d'une commande.
Une commande passe par différents stades :
- au moment de son enregistrement, elle est "en cours" ;
- au moment de sa réception, elle est "livrée" ;
- une fois livrée, le paiement est effectué, elle est alors "réglée" ;
- dans le cas où la livraison tarde, la commande est "relancée" (ce qui provoque un envoi de mail au fournisseur : cet aspect n'est pas à gérer) ;

La gestion des stades n'a pas encore été modélisée dans la base de données actuelle : vous devez donc vous en charger.
L'application doit permettre de voir, pour les livres ou les DVD, la liste des commandes et gérer le suivi.
Lorsqu'une commande de livre ou de DVD est "livrée", il faut que les exemplaires concernés soient automatiquement générés dans la BDD, avec un numéro séquentiel par rapport au document concerné.
Une commande doit pouvoir être supprimée si elle n'est pas encore livrée.

Tâche 2 : gérer les commandes de revues (4h)

Il doit être possible de commander une revue. Une commande de revue revient à réaliser un abonnement (ou un renouvellement d'abonnement, ce qui revient au même).
L'application doit permettre de voir la liste des commandes, même pour les abonnements expirés. Une commande ne peut être supprimée que si aucun exemplaire (parution) lié à cette commande n'est enregistré.
Au démarrage de l'application, une petite fenêtre doit s'ouvrir automatiquement pour afficher les revues dont l'abonnement se termine dans moins de 30 jours.

Mission 3 : gérer le suivi de l'état des exemplaires (5h)

L'application doit permettre de gérer l'état des documents physiques, donc des exemplaires.
Aussi bien pour les exemplaires de livres, de DVD et de revues (donc les parutions), leur création les met dans l'état "neuf". L'application doit permettre de changer l'état ("neuf", "usagé", "détérioré", "inutilisable").
Il doit être possible de supprimer un exemplaire.

Mission 4 : mettre en place des authentifications (4h)

Il existe plusieurs services dont les employés ne doivent pas avoir les mêmes droits d'accès aux fonctionnalités de l'application.
Les employés du service administratif gèrent les commandes. Ils ont aussi accès à la consultation et la mise à jour du catalogue (livres, DVD, revues). Donc ils ont accès à toutes les fonctionnalités de l'application.
Les employés du service Prêts s'occupent des prêts de documents, des retours, de l'étiquetage et du rangement dans les rayons. Ils ont accès en consultation seulement au catalogue (livres, DVD, revues) disponible.
Les employés du service Culture organisent divers évènements (conférences, expositions, lecture aux enfants, etc..) et gèrent la venue d'intervenants pour ces évènements. Ils ne doivent donc pas avoir accès aux fonctionnalités de cette application.
L'administrateur a accès à toutes les fonctionnalités de toutes les applications.

Dans le cadre d'une première version du système d'authentification, le but est de gérer les employés (avec login/pwd) et les services directement dans la base de données.
L'application doit donc démarrer sur une authentification qui permet de déterminer si l'employé est reconnu et son service d'affectation.
Suivant le service d'appartenance de l'employé, certaines fonctionnalités ne seront pas accessibles. Dans le cas des employés du service Culture, un message doit les informer que cette application n'est pas accessible pour eux.

Mission 5 : assurer la sécurité, la qualité et intégrer des logs (8h)

Tâche 1 : corriger des problèmes de sécurité (3h)

Pour chaque problème, créer une "issue" dans le dépôt GitHub correspondant, affecter l'issue à un des développeurs (si vous êtes 2, vous ou votre collègue, sinon vous-même), créer une branche pour proposer une correction de code en expliquant la correction, faire un pull request, accepter le pull request si le résultat correspond bien aux attentes.
Problème n°1 :
Actuellement, l'accès à l'API se fait en authentification basique, avec le couple "login:pwd" en dur dans le code de l'application (dans le constructeur de la classe Access). Le but est de sécuriser cette information.
Problème n°2 :
Si, pour accéder à l'API directement dans un navigateur, on donne juste l'adresse de l'API sans mettre de paramètres :
http://localhost//rest_mediatekdocuments/
on obtient la liste des fichiers contenus dans le dossier de l'API. Le but est d'avoir un retour d'erreur.

Tâche 2 : contrôler la qualité (3h)

Mettre en place, en local, un serveur SonarQube et le lien avec l'application C# avec un intégration continue entre C# et SonarQube, avec Jenkins.
Le code d'origine a été nettoyé en suivant les conseils de Sonarlint et des avertissements de Visual Studio. Le seul point qui n'a pas été corrigé est le nom des méthodes événementielles, qui commencent par une minuscule car les objets graphiques commencent par une minuscule.
Excepté ce point, les règles d'écriture de code sont les règles standards.
Le but est donc de nettoyer le code ajouté en suivant la même logique.
Contrôler la qualité avec SonaQube.

Tâche 3 : intégrer des logs (2h)

Dans la classe Access, tous les affichages console doivent être aussi enregistrés dans un fichier de logs.

Mission 6 : tester et documenter (8h)

Tâche 1 : gérer les tests (5h)

Tests unitaires :
Écrire les tests unitaires sur les classes du package Model.
Tests fonctionnels :
Le responsable désire voir le fonctionnement des tests avec Specflow : dans l'application C#, écrire les tests fonctionnels sur les recherches d'un onglet (par exemple les livres).
Construire une collection de tests dans Postman pour contrôler les fonctionnalités de l'API d'accès à la BDD.

Tâche 2 : créer les documentations techniques (1h)

Générer les documentations techniques correspondant à l'application C# et à l'API REST.

Tâche 3 : créer la documentation utilisateur en vidéo (2h)

Créer une vidéo de 10mn maximum qui présente l'ensemble des fonctionnalités de l'application C#.

Mission 7 : déployer et gérer les sauvegardes des données (4h)

Tâche 1 : déployer le projet (3h)

Mettre en ligne l'API et la base de données (chez un hébergeur).
Créer un installeur de l'application C# pour qu'elle puisse s'installer facilement.

Tâche 2 : gérer les sauvegardes des données (1h)

Automatiser une tâche de sauvegarde quotidienne de la BDD.