Application bureautique Mediatek86 de gestion pour médiathèque (C#)

SOMMAIRE

1. Contexte

La société InfoTech Services 86 (ou *ITS 86*) est constituée d'une équipe de 32 collaborateurs, administratifs, ingénieurs et techniciens. Elle organise ses activités autour de deux pôles : développement et Systèmes et réseau. Ses clients sont des TPE et des PME.

Le pôle développement propose des solutions d'hébergements sur des serveurs dédiés, intégration de services et développement logiciel avec gestion et création de bases de données.

ITS 86 a récemment obtenu la gestion du parc informatique du réseau des médiathèques de la Vienne, *MediaTek86*. Les médiathèques mettent à disposition de leurs adhérents des documents sur divers supports (papier, cédérom, livre numérique, DVD, etc...).

Dans chaque médiathèque, il existe plusieurs types d'employés :

- Ceux du service Administratif, qui gèrent notamment les achats de livres
- Ceux du service Prêt, qui s'occupent des prêts et du rangement des documents dans la médiathèque
- Ceux du service Médiation culturelle, qui communiquent sur les activités de la médiathèque

MediaTek86 gère le portail web unique, qui permet aux adhérents de réserver et d'emprunter plus de 200 000 documents (livres, journeaux, CD, DVD, jeux), revues et ressources numériques. *Mediatek86* est un réseau qui regroupe les responsables de chaque médiathèque, un chef de projet numérique, une chargée de communication numérique et un infographiste. Les projets numériques sont pilotés par le chef de projet numérique qui est chargé de la maîtrise d'ouvrage auprès des entreprises de services numériques auxquelles *MediaTek86* fait appel.

Dans le cadre du développement de la médiathèque numérique pour l'ensemble des médiathèques du département, MediaTek86 a fait appel à ITS 86 pour développer une application pour les médiathèques, permettant le visionnage et la gestion des documents et des commandes ainsi que le suivi de l'état des documents. Cette application sera installée sur plusieurs postes de la médiathèque pour accéder à la même base de données.

Un autre développeur s'est chargé de la construction de la base de données et certaines fonctionnalités de l'application. Ce développeur a ajouté les fonctionnalités de recherche et d'affichage d'informations sur les documents de la médiathèque. Il a également ajouté la gestion de la réception de nouveaux numéros de revues.

Les évolutions demandées sont les suivantes :

1. Permettre la gestion des documents : possibilité d'ajouter, de modifier ou de supprimers des documents, sous certaines conditions.
2. Permettre la gestion des commandes :
3. Permettre la gestion du suivi de l'état des documents
4. Mettre en place des authentifications
5. Contrôler la qualité du code, ajouter des tests, générer la documentation technique et déployer l'application
6. Réaliser une vidéo de démonstration d'utilisation de l'application

Moi et un autre développeur avons fait équipe pour réaliser ces missions.

2. Outils utilisés

Les outils utilisés pour cette situation professionnelle sont les suivants :

- IDE : **Visual Studio 2019 Entreprise**
- Analyse de la qualité du code : **SonatLint** (extension pour Visual Studio)
- Logiciel de gestion de versions : **GitHub**
- Outil de suivi de projet collaboratif en ligne : **Trello**
- Outil de communication instantanée : **Discord**

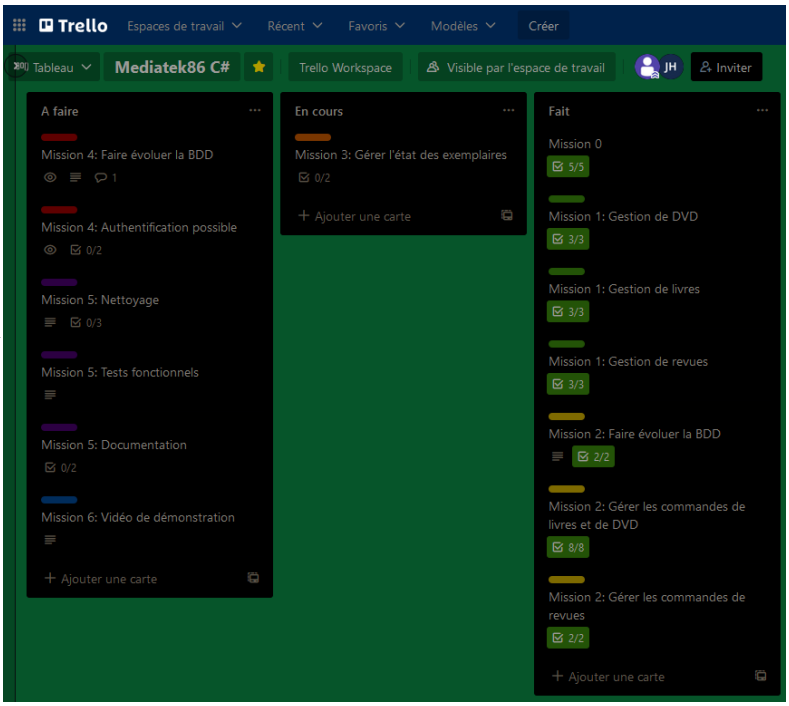
3. Réalisation des missions

Au début du développement, moi et mon coéquipier avons commencé par décider de répartir les tâches d'une même mission de façon à travailler ensemble pour nous entre-aider et perdre moins de temps en cas de problème.

Pour la gestion des versions avec GitHub, nous avons travaillé tout au long du développement avec une branche dev chacun. Lorsqu'une tâche était terminée, nous fusionnions notre branch dev individuelle vers la branche main afin d'ajouter nos fonctionnalités à l'application principale.

Nous avons également utilisé l'outil de suivi de projet Trello pour suivre les différentes étapes de réalisation des missions.

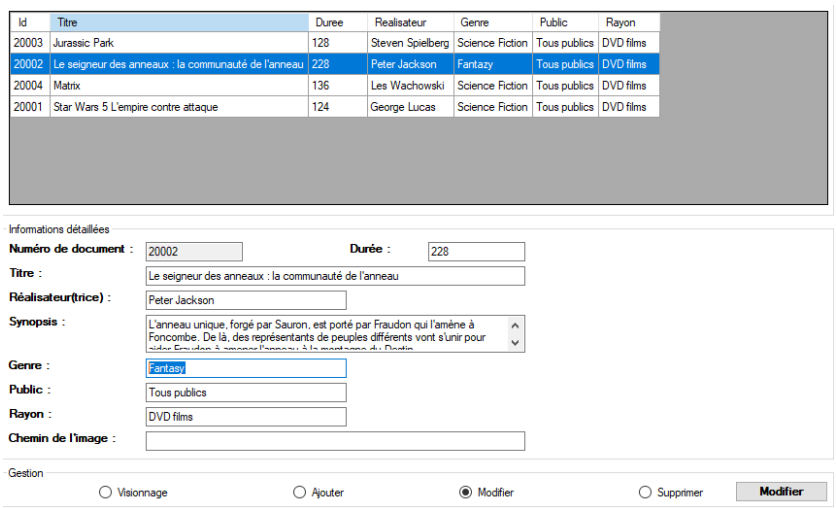
Chaque carte correspond à une mission et contient une checklist avec un ensemble de tâches à compléter. Cela nous a permis d'avoir une idée très claire de l'avancée de notre travail tout au long du développement.



3.1. Gestion des documents

La gestion des documents demandée consistait en l'évolution de l'application, de façon à permettre, dans les onglets "Livres", "Dvd", et "Revues" préexistants, de permettre l'ajout, la modification et la suppression de documents. Les modifications peuvent pas porter sur un numéro (qui sert d'identifiant aux documents), et la suppression n'est possible que si le document ne possède aucun exemplaire ni commande.

Pour répondre à ces besoins, j'ai développé la partie gestion (ajout, modification et suppression) des DVD et géré l'uniformisation visuelle des objets graphiques entre les différents onglets. L'autre développeur s'est chargé de développer les fonctions d'ajout, de modification et de suppression communes aux différents onglets dans le controleur, puis la partie gestion de livre et de revues.



Pour l'affichage, nous avons fini par opter pour des boutons radio permettant d'alterner entre "visionnage", "ajout", "modifier" et "supprimer". Ce système permet la gestion des documents en réutilisant les zones de saisies préexistantes, et d'éviter les répétitions des informations concernant les documents dans la fenêtre de l'application.

Une fonction par onglet se charge de rassembler les traitements liés à la gestion de documents. Par exemple, pour la gestion des Dvd, chaque clic sur un bouton radio redirige vers cette fonction :

```
private void GestionRadioLivre(string action)
{
    switch (action)
    {
        case "visionnage":
            // lecture seule
            [...]
            break;
        case "ajouter":
            // modification des informations possible
            [...]
            break;
        case "modifier":
            // modification des informations possible
            [...]
            break;
        case "supprimer":
            // lecture seule
            [...]
            break;
    }
}
```

Au terme de cette mission, les objectifs suivants ont été atteints :

- Groupe permettant la gestion des documents dans les onglets Livres, Dvd et Revues
- La suppression d'un document n'est possible que si celui-ci n'a pas de commande ou d'exemplaire associé.
- Des triggers contrôlant les contraintes de partition des héritages sur les tables Document et LivresDvd ont été ajoutés à la base de données.

3.2. Gestion des commandes

La mission de gestion des commandes est constituée de 2 grandes parties distinctes à développer :

- Les onglets de commandes de livres ou de DVD, qui sont très similaires
- L'onglet d'abonnements aux revues, qui nécessite du code qui lui est propre

Pour cela, il fallait pouvoir gérer les différents stades d'une commande de document dans l'application. La mission consistait donc en l'ajout des stades à la base de données, l'ajout de la classe Suivi à l'application pour récupérer les stades des commandes, et enfin, l'ajout des onglets de commandes pour les livres, les DVD et les revues. Lorsqu'une commande de livres ou de DVD est "livrée", les exemplaires concernés doivent être générés automatiquement dans la base de données. De plus, les commandes ne doivent pas pouvoir être supprimées si elles ont été livrées. Dans le cas des commandes de revues, un abonnement à une revue ne peut être supprimé que si aucun exemplaire n'y est lié, donc si aucun exemplaire de cette revue n'a été ajouté pendant la période d'abonnement. Enfin, l'étape de suivi des commandes ne doit pas pouvoir retourner en arrière : une commande livrée ou réglée ne peut pas passer à l'étape "en cours" ou "relancée", et une commande ne peut être réglée que si elle a été livrée.

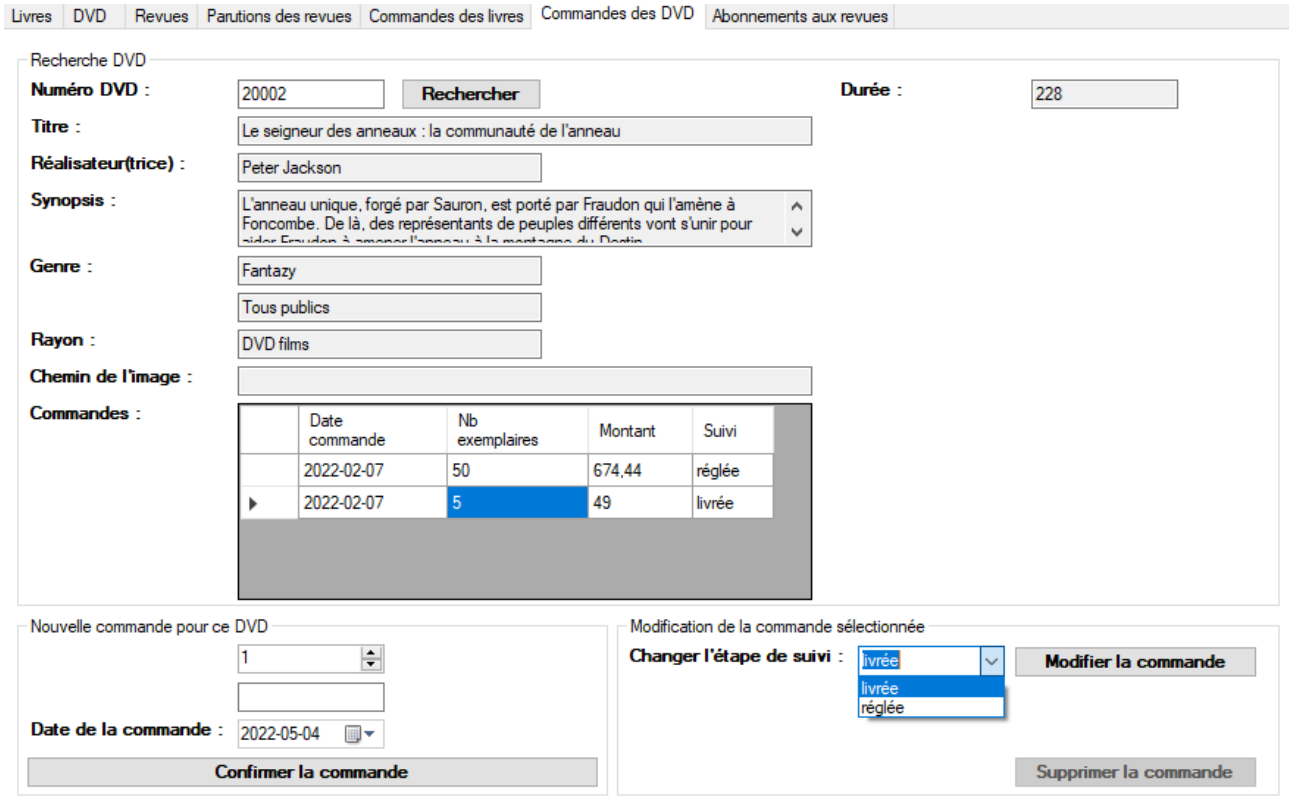
Pour la répartition des tâches, j'ai commencé par coder la partie graphique des commandes de DVD pendant que l'autre développeur codait les commandes de livres et abonnements aux revues. J'ai ensuite rassemblé les fonctions similaires des trois onglets en des fonctions communes, et uniformisé la présentation des nouveaux onglets avec ceux préexistants. Nous avons ensuite ajouté les conditions de suppression pour les commandes de livres et de DVD, ainsi que pour les abonnements aux revues.

La gestion des différentes étapes de suivi disponibles lors de la modifications d'une commande pour les livres et les DVD est donc gérée dans une seule fonction, *GetSuivisAffiches* :

```
private List<Suivi> GetSuivisAffiches(CommandeDocument commandeDocument)
{
    Suivi suivi = lesSuivis.Find(x => x.Id == commandeDocument.IdSuivi);
    List<Suivi> suivisAffiches = new List<Suivi>();
    suivisAffiches.AddRange(lesSuivis);
    string libelle = suivi.Libelle;
    if (libelle == "livrée" || libelle == "réglée")
    {
        suivisAffiches.RemoveAll(x => x.Libelle.Length > 6);
    }
    else
    {
        suivisAffiches.RemoveAt(suivisAffiches.FindIndex(x => x.Libelle == "réglée"));
    }
    suivisAffiches.Remove(suivi);
    suivisAffiches.Insert(0, suivi);
    return suivisAffiches;
}
```

Pour gérer les commandes livrées, nous avons ajouté un trigger à la base de données, pour ajouter automatiquement des exemplaires à un document lorsque une commande pour celui-ci est livrée. La contrainte de partition de l'héritage sur la table Commande a également été gérée avec un trigger dans la base de données.

Enfin, une procédure stockée a été ajoutée à la base de données pour obtenir la liste des revues dont l'abonnement se termine dans moins de 30 jours. Dès l'ouverture de l'application, un message d'alerte utilise cette procédure pour afficher la liste de ces revues.



Au terme de cette mission, les objectifs suivants ont été atteints :

- Gestion des commandes de livres et de DVD dans deux nouveaux onglets
- Ajout de la table Suivi à la base de données et intégration des étapes de suivi aux commandes
- Possibilité de modifier les étapes de suivi d'une commande selon des conditions précises
- Possibilité de supprimer une commande uniquement si elle n'a pas été livrée
- Trigger ajoutant automatiquement des exemplaires lorsqu'une commande est livrée
- Gestion des abonnements aux revues dans un nouvel onglet
- Possibilité de supprimer un abonnement à une revue si aucun exemplaire n'y est rattaché
- Trigger contrôlant la contrainte de partition de l'héritage sur la table Commande
- La présentation de tous les onglets est similaire

3.3. Gestion du suivi de l'état des documents

L'objectif de la mission est de pouvoir changer les états des exemplaires de livres, de DVD et de revues (donc les parutions). Les états possibles sont les suivantes : "neuf", "usagé", "détérioré", "inutilisable". Il doit aussi être possible de supprimer un exemplaire.

Pour la répartition des tâches, j'ai codé la partie parutions des revues et l'autre développeur s'est chargé des livres et des DVD. Je me suis ensuite occupé de l'aspect des nouveaux objets graphiques pour qu'ils soient similaires au reste de l'application, ainsi que de l'optimisation du code pour éviter les répétitions. La gestion du suivi de l'état des documents a nécessité un agrandissement de la fenêtre pour ajouter la liste des exemplaires en bas des onglets Livres et Dvd.

Au terme de cette mission, les objectifs suivants ont été atteints :

- Gestion des exemplaires de livres et de DVD
- Gestion des parutions de revues

3.4. Mise en place des authentifications

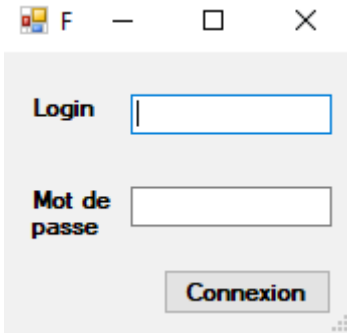
La mise en place des authentifications doit être gérée directement dans la base de données. Il faut donc la modifier pour permettre l'authentification des employés des différents services de MediaTek86. Il est aussi demandé d'ajouter une fenêtre d'authentification qui doit s'ouvrir dès le lancement de l'application. Enfin, selon le type de personne authentifiée, certains accès doivent être bloqués :

- L'administrateur et le service administratif ont accès à toutes les fonctionnalités de l'application
- Les employés du service Prêts ont accès en consultation seulement au catalogue (livre, DVD, revues)
- Les employés du service Culture n'ont pas accès à l'application

Au début de la mission, nous avons ajouté les deux tables Utilisateur et Service pour permettre l'authentification.
Ensuite, nous avons réparti les tâches de la façon suivante :

- L'autre développeur a créé la fenêtre d'authentification et s'est occupé de la gestion de l'authentification pour l'administrateur et le service administratif
- Je me suis occupé de la gestion de l'authentification pour les employés du service Prêts et ceux du service Culture.

Pour l'administrateur et le service administratif, ceux-ci ont accès à toutes les fonctionnalités de l'application, il n'y avait donc qu'à coder les fonctions d'authentification.



Pour les employés du service Culture, ceux-ci n'ayant pas accès à l'application, il a suffit de mettre un message d'alerte indiquant à l'utilisateur que ses droits ne sont pas suffisants pour accéder à l'application. Ensuite, l'application se ferme automatiquement.

Pour les employés du service Prêts, ceux-ci n'ayant accès au catalogue qu'en consultation seule, j'ai ajouté la fonction *AffichageApplication* permettant de retirer les onglets de gestion, et rendre invisible les groupes de gestion :

```
public void AffichageApplication(string role)
{
    if (role == "prêts")
    {
        // retire les onglets de gestion
        while (tabOngletsApplication.TabCount > 3)
        {
            tabOngletsApplication.TabPages.RemoveAt(3);
        }
        // impossible de gérer le catalogue
        grpLivreGestion.Visible = false;
        grpDvdGestion.Visible = false;
        string[] controlesLectureExemplaires = { dgvLivreExemplairesListe.Name,
            lblLivreExemplaires.Name, pcbLivreExemplaireImage.Name,
            dgvDvdExemplairesListe.Name, lblDvdExemplaires.Name,
            pcbDvdExemplaireImage.Name };
        CacheControle(grpLivreExemplaire, controlesLectureExemplaires);
        CacheControle(grpDvdExemplaire, controlesLectureExemplaires);
        grpRevueGestion.Visible = false;
```

```
        // remise en place des groupes affichant les exemplaires en conséquence
        grpLivreExemplaire.Location = new Point(grpLivreExemplaire.Location.X,
            grpLivreExemplaire.Location.Y - (grpLivreGestion.Height + 6));
        grpDvdExemplaire.Location = new Point(grpDvdExemplaire.Location.X,
            grpDvdExemplaire.Location.Y - (grpDvdGestion.Height + 6));
        // réduction de la taille de la fenêtre principale de l'application
        int reductionHauteur = 110;
        grpDvdExemplaire.Size = new Size(grpDvdExemplaire.Width,
            grpDvdExemplaire.Height - reductionHauteur);
        grpLivreExemplaire.Size = new Size(grpLivreExemplaire.Width,
            grpLivreExemplaire.Height - reductionHauteur);
        this.Size = new Size(this.Width, 936);
    }
}
```

Par ailleurs, juste après l'authentification, il a fallu ajouter un test pour n'afficher l'alerte de fin d'abonnement qu'à l'administrateur ou aux employés du service administratif.

Au terme de cette mission, les objectifs atteints étaient les suivants :

- Ajout des tables Utilisateur et Service dans la base de données
- Ajout d'une fenêtre d'authentification à l'ouverture de l'application
- Certains accès sont bloqués selon le type de personne authentifiée.

3.5. Contrôle de la qualité du code, tests, documentation technique et déploiement

Pour cette mission, il est demandé de nettoyer le code, d'ajouter les logs, des tests fonctionnels, et générer la documentation technique .

Pour répartir les tâches, je me suis occupé du contrôle de la qualité du code en m'aidant de SonarLint, et du nettoyage du code manuellement car SonarLint détecte mal ou pas les problèmes de redondance de code ou de variables mal nommées. J'ai également mis en ligne la base de données en ligne et modifié l'accès à la base de données dans l'application en conséquence.

L'autre développeur s'est occupé d'ajouter des logs dans les parties *catch* contenant des affichages consoles, et de paramétrer ces logs pour qu'ils s'enregistrent dans un fichier texte. Il s'est également chargé de générer la documentation technique avec VSdocman.

J'ai ensuite créé un installateur de l'application pour pouvoir la déployer.

Les tests fonctionnels n'ont pas pu être réalisés, SpecFlow ne fonctionnant correctement avec une application Windows Form sur aucune de nos installations.

Au terme de cette mission, les objectifs suivants ont été atteints :

- Contrôle de la qualité du code avec SonarLint
- Nettoyage du code
- Ajout de logs dans les *catch* avec affichages consoles
- Documentation technique générée à l'aide de VSdocman
- Déploiement de l'application avec base de données en ligne

4. Bilan

L'application MediaTek86 permet maintenant de gérer l'ajout, la modification et la suppression des documents de la médiathèque. La suppression d'un document n'est possible que si aucune commande n'est liée à celui-ci.

Les fonctionnalités liées aux commandes ont été ajoutées : il est possible de visionner les commandes pour chaque document, et d'ajouter, de modifier ou de supprimer des commandes.

La gestion des exemplaires a également été ajoutée, permettant de voir et modifier l'état des exemplaires commandés pour chaque document.

Toute l'application est protégée par un système d'authentification, il faut donc être un utilisateur reconnu et avec les droits suffisants pour pouvoir y accéder.

La base de données a évolué en conséquence, avec l'ajout des tables Suivi, Utilisateur et Service. Des triggers ont également été ajoutés pour gérer les contraintes de partition des héritages des tables Document, LivreDvd et Commande. Un autre trigger gère l'ajout automatique d'exemplaires à la base de données lorsqu'une commande est livrée.

Enfin, un nettoyage général du code de l'application a été effectué à l'aide de SonarLint, des logs ont été ajoutés et la documentation technique a été générée. L'application a ensuite été deployée.