



Compte rendu d'activité 3

Evolution de l'application bureau de gestion documentaire de Mediatek86

Table des matières

1Rappel du contexte	4
2Rappel des missions à effectuer.....	5
2.1Mission 1 : Gestion des commandes.....	5
2.2Mission 2 : Mettre en place des authentifications.....	6
2.3Mission 3 : qualités, test, documentation technique et déploiement.....	7
3Présentation des outils utilisés :.....	8
4Réalisations des missions	8
4.1Mission 1 : : Gestion des commandes.....	8
4.1.1Etape 1 : Changement BDD	8
4.1.2Etape 2 : Ajout d'un nouvel onglet : Commande de livres.....	8
4.1.3Etape 3 : Ajout d'objets graphiques dans l'onglet commande de livres.....	9
4.1.4Etape 4 : : Ajout d'une condition pour la modification d'une commande de livres	9
4.1.5Etape 5 : Ajout d'une condition pour la suppression d'une commande de livres	10
4.1.6Etape 6 : Création d'un trigger dans la BDD.....	10
4.1.7Etape 7 : Ajout d'un nouvel onglet : Commande de Dvd	10
4.1.8Etape 8 : Ajout d'un nouvel Onglet : Commande de Revues	11
4.1.9Etape 9 : Ajout d'objets graphiques dans l'onglet commande de revues.....	12
4.1.10Etape 10 : Ajout d'une condition sur la suppression d'un abonnement.....	12
4.1.11Etape 11 : Ajout d'une condition sur la modification d'un abonnement et test unitaire...	13
4.1.12Etape 12 : Sécurité et optimisation de l'interface utilisateur	14
4.1.13Etape 13 : Création d'un trigger pour respecter la contrainte de partition.....	15
4.1.14Etape 14 : Récupération des abonnements arrivant à expiration	15
4.2Mission 2 : Mettre en place des authentifications.....	16
4.2.1Etape 1 : Changement dans la BDD.....	16
4.2.2Etape 2 : Ajout d'une fenêtre d'authentification	16
4.2.3Etape 3 et 4 : Mise en place des niveaux d'habilitations	17
4.2.4Etape 5 : Modification de la fenêtre d'alerte	17
4.3Mission 3 : qualités, test, documentation technique et déploiement.....	18
4.3.1Etape 1 : SonarLint	18
4.3.2Etape 2 : Ajout de tests unitaires	18
4.3.3Etape 3 : Ajout de logs.....	19
4.3.4Etape 4 : Création de la documentation technique	20
4.3.5Etape 5 : Configuration du cloud pour l'hébergement de la BDD.....	20
4.3.6Etape 6 : Génération du MSI	20

5Bilan.....	21
6Listes des compétences couvertes.....	21
6.1B1 : Support et mise à disposition de services informatiques	21
6.2B2 : BLOC 2 – SLAM – Conception et développement d’applications.....	22
6.3B3 : SLAM - Cybersécurité des services informatiques – 2e année	23

1 Rappel du contexte

InfoTech Services 86 (ITS 86), est une Entreprise de Services Numériques (ESN) spécialisée dans le développement informatique (applications de bureau, web, mobile), l'hébergement de site web, l'infogérance, la gestion de parc informatique et l'ingénierie système et réseau. Elle répond régulièrement à des appels d'offres en tant que société d'infogérance et prestataire de services informatiques.

ITS 86 est une équipe composée de 32 collaborateurs, administratifs, ingénieurs et techniciens dont les activités s'organisent autour de 2 pôles : le pôle développement et le pôle système et réseaux.

La principale activité du pôle Développement consiste à proposer des solutions d'hébergements sur des serveurs dédiés. Les développeurs possèdent également une expertise dans l'intégration de services, le développement logiciel et la gestion/création de bases de données.

Les équipes de développement accompagne les clients dans différentes étapes de leurs projets :

- Analyse de la problématique, rédaction du cahier des charges, assistance à maîtrise d'ouvrage (AMO) ;
- Développement d'applications de bureau en C# et en Java pour répondre à des besoins spécifiques en monoposte ou multipostes ;
- Développement d'applications mobiles (Android, iOS) ;
- Développement d'applications web : HTML, PHP, JavaScript, CSS, Ajax, Symfony, Angular JS. Les applications sont multi-plateformes (Windows, Linux, Mac) et multi-navigateurs (Edge, Mozilla, Firefox, Opera, Chrome, Safari) et répondent aux standards du W3C ;
- Administration de base de données : Intégration et interconnexion avec les systèmes existants, fourniture de l'ensemble des documentations liées à ses applications.



2 Rappel des missions à effectuer

2.1 Mission 1 : Gestion des commandes

Objectif de la mission :

Etape 1 : Dans la base de données, créer la table 'Suivi' qui contient les différentes étapes de suivi d'une commande de document de type livre ou dvd. Relier cette table à CommandeDocument.

Etape 2 : Créer un onglet pour gérer les commandes de livres. Dans l'onglet, permettre la sélection d'un livre par son numéro, afficher les informations du livre ainsi que la liste des commandes, triée par date (ordre inverse de la chronologie). La liste doit comporter les informations suivantes : date de la commande, montant, nombre d'exemplaires commandés et l'étape de suivi de la commande.

Etape 3 : Créer un Groupbox qui permet de saisir les informations d'une nouvelle commande et de l'enregistrer. Lors de l'enregistrement de la commande, l'étape de suivi doit être mise à "en cours".

Etape 4 : Permettre de modifier l'étape de suivi d'une commande en respectant certaines règles (une commande livrée ou réglée ne peut pas revenir à une étape précédente : en cours ou relancée, une commande ne peut pas être réglée si elle n'est pas livrée).

Etape 5 : Permettre de supprimer une commande uniquement si elle n'est pas encore livrée.

Etape 6 : Créer le trigger qui se déclenche si une commande passe à l'étape "livrée" et qui crée autant de tuples dans la table "Exemplaire" que nécessaires, en valorisant la date d'achat avec la date de commande et en mettant l'état de l'exemplaire à "neuf". Le numéro d'exemplaire doit être séquentiel par rapport au livre concerné.

Etape 7 : Créer un onglet pour gérer les commandes de DVD en suivant la même logique que pour les commandes de livres.

Etape 8 : Créer un onglet pour gérer les commandes de revues : une commande représente un nouvel abonnement ou le renouvellement d'un abonnement. Dans le cas d'un nouvel abonnement, la revue sera préalablement créée dans l'onglet Revues. Donc, dans l'onglet des commandes de revues, il n'y a pas de distinction entre un nouvel abonnement et un renouvellement. Permettre la sélection d'une revue par son numéro, afficher les informations de la revue ainsi que la liste des commandes (abonnements), triée par date (ordre inverse de la chronologie). La liste doit comporter les informations suivantes : date de la commande, montant et date de fin d'abonnement.

Etape 9 : Créer un GroupBox qui permet de saisir les informations d'une nouvelle commande (nouvel abonnement ou renouvellement, le principe est identique) et de l'enregistrer.

Etape 10 : Une commande de revue peut être supprimée, si aucun exemplaire n'est rattaché (donc, en vérifiant la date de l'exemplaire, comprise entre la date de la commande et la date de fin d'abonnement). Pour cela, créer et utiliser la méthode 'ParutionDansAbonnement' qui reçoit en paramètre 3 dates (date commande, date fin abonnement, date parution) et qui retourne vrai si la date de parution est entre les 2 autres dates. Créer le test unitaire sur cette méthode.

Etape 11 : La présentation de chaque onglet de gestion des commandes doit être similaire à l'onglet "Parutions des revues". Dans toutes les listes, permettre le tri sur les colonnes.

Etape 12 : Toutes les sécurités seront mises en place pour éviter des erreurs de manipulation.

Etape 13 : Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Commande.

Etape 14 : Créer une procédure stockée qui permet d'obtenir la liste des revues dont l'abonnement se termine dans moins de 30 jours. Dès l'ouverture de l'application, ouvrir une petite fenêtre d'alerte rappelant la liste de ces revues (titre et date de fin abonnement) triée sur la date dans l'ordre chronologique.

2.2 Mission 2 : Mettre en place des authentifications

Objectif de la mission :

Etape 1 : dans la base de données, ajouter une table Utilisateur et une table Service, sachant que chaque utilisateur ne fait partie que d'un service. Pour réaliser les tests, remplir les tables d'exemples.

Etape 2 : ajouter une première fenêtre d'authentification. Faire en sorte que le contrôleur ouvre cette fenêtre en premier.

Etape 3 : suivant le type de personne authentifiée, empêcher certains accès en rendant invisibles ou inactifs certains onglets ou objets graphiques.

Etape 4 : dans le cas du service Culture qui n'a accès à rien, afficher un message précisant que les droits ne sont pas suffisants pour accéder à cette application.

Etape 5 : faire en sorte que l'alerte de fin d'abonnement n'apparaisse que pour les personnes concernées (qui gèrent les commandes).

2.3 Mission 3 : qualités, test, documentation technique et déploiement

Objectif de la mission :

Etape 1 : contrôler la qualité du code avec SonarLint.

Etape 2 : créer des tests fonctionnels (avec SpecFlow ou directement avec les NUnit).

Etape 3 : ajouter des logs dans les catches qui contiennent des affichages consoles et enregistrer ces logs dans un fichier texte.

Etape 4 : générer la documentation technique de l'application complète.

Etape 5 : mettre en ligne la BDD (par exemple directement dans le Cloud ou une VM sur Azure ou AWS), dans l'application modifier l'accès à la BDD et tester.

Etape 6 : créer un installateur de l'application

3 Présentation des outils utilisés :

IDE : Visual Studio Code 2019

Base de données : MariaDB 10.2.38

Hébergement : Machine Ec2 proposé par AWS

Qualité de code : SonarLint 4.35

Logiciel de versionning : GitHub

Logiciel pour générer la documentation : Vsdocman 11.2

Logiciel pour l'ajout des logs dans un fichier : Serilog 2.10




4 Réalisations des missions


4.1 Mission 1 : : Gestion des commandes

4.1.1 Etape 1 : Changement BDD

Explication du travail réalisé : Pour créer la table suivie, il a fallu s'aider du gestionnaire de base de données phpMyAdmin, cette table comporte 2 champs : l'identifiant et le libelle. Pour relier cette table à la table CommandeDocument il a fallu ajouter une clé étrangère à la table CommandeDocument qui fait référence à l'identifiant de la table suivi.

Bilan du travail réalisés :

	#	Nom	Type
<input type="checkbox"/>	1	id 	varchar(5)
<input type="checkbox"/>	2	nbExemplaire	int
<input type="checkbox"/>	3	idLivreDvd 	varchar(10)
<input type="checkbox"/>	4	idsuivi 	varchar(5)

	#	Nom	Type
<input type="checkbox"/>	1	id 	varchar(5)
<input type="checkbox"/>	2	libelle	varchar(20)

4.1.2 Etape 2 : Ajout d'un nouvel onglet : Commande de livres

Explication du travail réalisé : Pour créer l'onglet de gestion des commandes de livres, il a fallu s'aider de l'interface graphique de Windpws Form et ajouter un évènement sur le bouton de recherche afin de pouvoir afficher les informations d'un livre ainsi que ses commandes. La méthode pour rechercher un livre avait déjà été codé il suffisait de l'appeler une nouvelle fois et de trier les commandes par ordre alphabétiques en triant au préalable la liste qui valorisait la DataGriedView. Pour récupérer les commandes il fallait coder dans le modèle la fonction getAllCommande() et la classe métier Commande.

Bilan du travail réalisés :

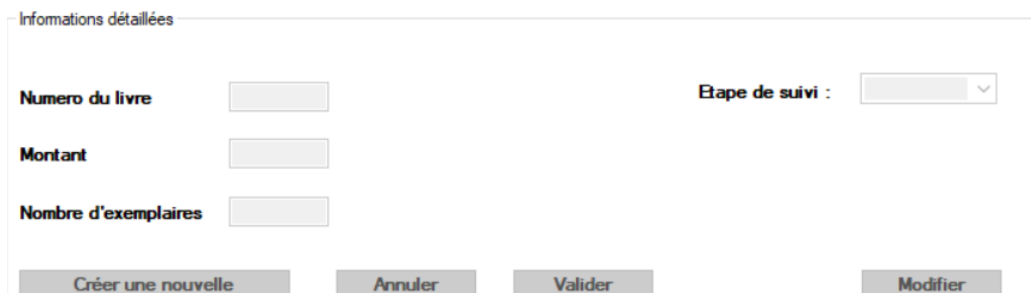


4.1.3 Etape 3 : Ajout d'objets graphiques dans l'onglet commande de livres

Explication du travail réalisé :

Pour créer le GroupBox il a fallu s'aider de l'interface graphique de Windows Form, cette interface graphique permet de créer une nouvelle commande, pour l'enregistrer il a fallu coder une méthode dans le modèle : creerCommandeDocument(Commande commande). Cette méthode doit au préalable appeler la méthode getLastIdCommande() pour récupérer le dernier identifiant, c'est cette identifiant qui permettra d'insérer dans la BDD la nouvelle commande.

Bilan du travail réalisé :



4.1.4 Etape 4 : Ajout d'une condition pour la modification d'une commande de livres

Explication du travail réalisé :

Une fois qu'une commande est créée sa statue est mise à « En cours », il est possible de le modifier suivant certaine règle pour cela la méthode remplirInfoCommande() i inclut une structure conditionnelle qui permet de respecter la règle . Si la règle est respectée la méthode qui permet de modifier une commande est appelée.

Bilan du travail réalisés :

```
if (unSuivi.Id == "EC" || unSuivi.Id == "REL")
{
    lstSuivisModif.Remove(lesSuivis.Find(x => x.Id.Equals("REG")));
}
else if (unSuivi.Id == "REG" || unSuivi.Id == "LI")
{
    lstSuivisModif.Remove(lesSuivis.Find(x => x.Id.Equals("REL")));
    lstSuivisModif.Remove(lesSuivis.Find(x => x.Id.Equals("EC")));
    if (unSuivi.Id == "REG")
    {
        lstSuivisModif.Remove(lstSuivisModif.Find(x => x.Id.Equals("LI")));
    }
}
```

4.1.5 Etape 5 : Ajout d'une condition pour la suppression d'une commande de livres

Explication du travail réalisé : Pour supprimer une commande, il faut au préalable vérifier grâce à une structure conditionnelle si la commande n'a pas été livrée, si c'est le cas la méthode qui permet de supprimer une commande est appelé dans la procédure événementielle en lien avec le bouton supprimer.

Bilan du travail réalisé :

```
if (commande.Suivi.Id != "LI")
{
    zoneNewCmdEnable(false);
    controle.deleteCmdLivre(commande);
    remplirLstCmdLivres(controle.GetAllCommandes(), txtNumLivreCmd.Text);
}
```

4.1.6 Etape 6 : Création d'un trigger dans la BDD

Explication du travail réalisé : Le trigger a été créé dans la table CommandeDocument et s'exécute avant une insertion, il faut au préalable récupéré le dernier numéro séquentiel de l'exemplaire, s'il n'y en a pas il faut l'initialiser.

4.1.7 Etape 7 : Ajout d'un nouvel onglet : Commande de Dvd

Explication du travail réalisé :

La méthode pour créer l'onglet des commandes de Dvd est similaire à celles employé pour l'onglet des commandes de Livres, il suffit donc de refaire les étapes 1 à 6 à quelques détails près.

Bilan du travail réalisés :

Recherches

Saisir un numéro de dvd :

Informations détaillées

Numero du dvd Etape de suivi :

Montant

Nombre d'exemplaires

4.1.8 Etape 8 : Ajout d'un nouvel Onglet : Commande de Revues

Explication du travail réalisé : La première partie de cette étape est similaire à l'étape 2 : la méthode qui permet de rechercher une revue et d'afficher ses informations est déjà codée il suffit de la réutiliser, pour récupérer les abonnements, il faut créer la méthode `getAllAbonnements()` et la classe métier abonnement dans le modèle. Une fois que la liste des abonnements est valorisée en passant par le contrôleur, celle-ci est affichée dans la DataGridView.

Bilan du travail réalisé :

Recherches

Saisir un numéro de revue :

Id	Titre	Periodicite	DelaiMiseADispo	Genre	Public
10002	Alternatives Economiques	MS	52	Presse Economique	Adultes

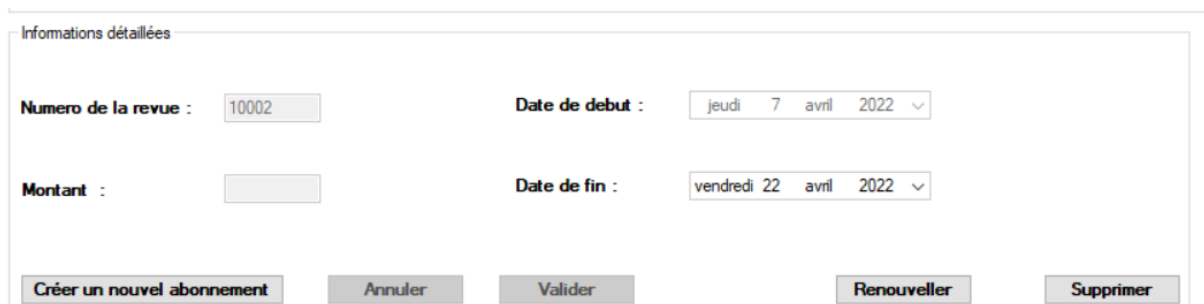
Id	DateCommande	Montant	DateFinAbonnement	IdRevue
51	07/04/2022	231	22/04/2022	10002
52	07/04/2022	100	24/04/2022	10002
54	07/04/2022	123	09/04/2022	10002
69	08/04/2022	232	13/04/2022	10002
70	10/04/2022	450	30/04/2022	10002
71	10/04/2022	450	30/04/2022	10002

4.1.9 Etape 9 : Ajout d'objets graphiques dans l'onglet commande de revues

Explication du travail réalisé :

Pour créer le GroupBox la méthode est similaire que pour les 2 autres onglets il faut s'aider de l'interface de Windows Form puis coder la procédure événementielle sur le bouton « créer un nouvel abonnement ». Lors du clic, si toutes les conditions sont respectées, la méthode qui permet de créer un nouvel abonnement est appelée via le constructeur.

Bilan du travail réalisé :



4.1.10 Etape 10 : Ajout d'une condition sur la suppression d'un abonnement

Explication du travail réalisé :

Dans la procédure événementielle qui est appelé lors du click sur le bouton supprimer, une fonction qui inclut une structure conditionnelle permet de vérifier si la date de chaque exemplaire n'est pas comprise entre la date de commande et la date de fin d'abonnement. Si oui la commande est supprimée. Pour vérifier que la fonction fonctionne correctement un test unitaire à été créé : 3 variables sont utilisées : la date d'aujourd'hui, la date d'aujourd'hui avec 1 jours supplémentaires et la date d'aujourd'hui avec 2 jours supplémentaires. En alternant chacune de ces valeurs toute les types de valeur d'entrées possibles sont testés.

Bilan du travail réalisé :

```
public bool isDeleteRevuePossible(DateTime dateDebut, DateTime dateFin, DateTime dateParution)
{
    // references | 0 modification | 0 auteur, 0 modification
    public void isDeleteRevuePossibleTest()
    {
        DateTime dateDebut = DateTime.Now;
        DateTime dateFin = DateTime.Now.AddDays(2);
        DateTime dateMilieu = DateTime.Now.AddDays(1);
        Assert.AreEqual(true, frmMediatek.isDeleteRevuePossible(dateDebut, dateFin, dateMilieu));
        Assert.AreEqual(false, frmMediatek.isDeleteRevuePossible(dateDebut, dateDebut, dateMilieu));
        Assert.AreEqual(true, frmMediatek.isDeleteRevuePossible(dateDebut, dateFin, dateDebut));
        Assert.AreEqual(false, frmMediatek.isDeleteRevuePossible(dateFin, dateFin, dateMilieu));
        Assert.AreEqual(true, frmMediatek.isDeleteRevuePossible(dateDebut, dateFin, dateFin));
        Assert.AreEqual(true, frmMediatek.isDeleteRevuePossible(dateDebut, dateFin, dateMilieu));
    }
}
```

4.1.11 Etape 11 : Ajout d'une condition sur la modification d'un abonnement et test unitaire

Explication du travail réalisé :

Pour gérer la modification d'un abonnement, il faut vérifier grâce à une structure conditionnelle si la date entrée n'est pas inférieure ou égale à la date initiale puis appeler la méthode qui permet de modifier un abonnement en passant par le contrôleur.

Pour permettre le tri sur les colonnes, il faut ajouter pour chaque onglet une procédure événementielle « Colum sur l'évènement : ColumHeaderMouseClick », la liste est ensuite triée suivant la colonne cliquée grâce à un switch :

Bilan du travail réalisé :

```
switch (titreColonne)
{
    case "Id":
        sortedList = lesCommandes.OrderBy(o => o.Id).ToList();
        break;
    case "DateCommande":
        sortedList = lesCommandes.OrderBy(o => o.DateCommande).ToList();
        break;

    case "Montant":
        sortedList = lesCommandes.OrderBy(o => o.Montant).ToList();
        break;
    case "NbExemplaire":
        sortedList = lesCommandes.OrderBy(o => o.NbExemplaire).ToList();
        break;
    case "IdLivreDvd":
        sortedList = controle.GetAllCommandes();
        break;
    case "Suivi":
        sortedList = controle.GetAllCommandes();
        break;
}
```

4.1.12 Etape 12 : Sécurité et optimisation de l'interface utilisateur

Explication du travail réalisé : Pour gérer les éventuelles erreurs, toute les entrées ont été entouré d'un « try catch » et l'utilisation de paramètres dans le modèle permet d'éviter les injections SQL . De plus pour faciliter l'utilisation du système par l'utilisateur et éviter les erreurs de manipulations les boutons sont disponibles uniquement quand cela est possible, il existe plusieurs situations possibles mais les plus importantes sont :

- Situation 1 : L'utilisateur n'a pas cliqué sur le bouton « créer une nouvelle commande/abonnement » donc le bouton « créer une nouvelle commande/abonnement », modifier et supprimer sont disponibles s'il existe des commandes/abonnement dans la Datagriedview .

- Situation 2 : L'utilisateur a cliqué sur le bouton « créer une nouvelle commande/abonnement » dans ce cas le bouton annuler et valider sont les seuls disponibles.

Bilan du travail réalisé :

La fonction suivante concerne les commandes de livres la situation 1 correspond au cas où le booléen entré est « false » et la situation 2 correspond au cas où le booléen entré est « true », il existe d'autres possibilités dans le cas où par exemple la datagridview est vide.

```
private void zoneNewCmdRevueEnable(bool saisie)
{
    if(txtCmdMontantRevue.Enabled = true)
    {
        viderNouvelleAbonnement();
    }
    txtCmdMontantRevue.Enabled = saisie;
    txtNumCmdRevue.Enabled = false;
    dateDebutCmdRevue.Enabled = false;
    btnAnnulerCmdRevue.Enabled = saisie;
    btnValiderCmdRevue.Enabled = saisie;
    btnRenouvelerCmdRevue.Enabled = !saisie;
    btnSupprimerCmdRevue.Enabled = !saisie;
    btnNewCmdRevue.Enabled = !saisie;
    dateFinCmdRevue.Enabled = true;
    if (dgvCmdRevue.CurrentCell == null)
    {
        btnRenouvelerCmdRevue.Enabled = false;
        dateFinCmdRevue.Enabled = saisie;
    }
}
```

4.1.13 Etape 13 : Création d'un trigger pour respecter la contrainte de partition

Explication du travail réalisé :

2 triggers ont été codés pour respecter la contrainte de partition, un dans CommandeDocument et l'autre dans Abonnement. Avant une insertion, si l'identifiant de la commande est déjà présent dans l'une ou l'autre table alors la contrainte de partition n'est pas respectée et le trigger renvoie une erreur.

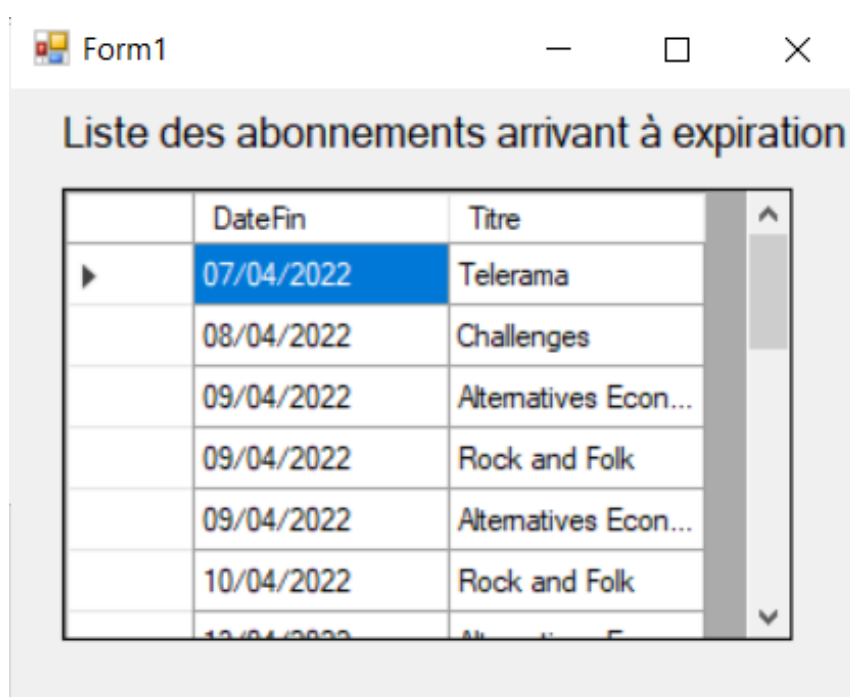
4.1.14 Etape 14 : Récupération des abonnements arrivant à expiration

Explication du travail réalisé :

La procédure stockée inclut une structure conditionnelle qui permet de récupérer les abonnements qui arrivent à expiration si la date de fin d'abonnement est inférieure à la date d'aujourd'hui + 30 jours. La fonction est appelée à l'aide de la méthode CALL() de MySQL à partir du modèle. Les informations sont ensuite récupérées via le contrôleur dans une liste qui valorisera la datagridview

La nouvelle Fenêtre est appelée dans une procédure événementielle sur l'évènement « Enter » de la « Tabcontrol » de cette façon elle s'affiche après la fenêtre principale

Bilan du travail réalisé :



	DateFin	Titre
▶	07/04/2022	Telerama
	08/04/2022	Challenges
	09/04/2022	Alternatives Econ...
	09/04/2022	Rock and Folk
	09/04/2022	Alternatives Econ...
	10/04/2022	Rock and Folk
	12/04/2022	Alternatives Econ...

4.2 Mission 2 : Mettre en place des authentifications

4.2.1 Etape 1 : Changement dans la BDD

Explication du travail réalisé :

Une fois la table Utilisateur créée il a fallu lier l'utilisateur à un service grâce à une clé étrangère dans la table Utilisateur qui fait référence à l'identifiant d'un service. Pour les tests des utilisateurs de chaque service ont été créés avec des mots de passe identiques : « password » hashée selon l'algorithme SHA256.

Bilan du travail réalisé :

id	login	password	idservice
1	culture	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a...	4
2	admin	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a...	1
3	administratif	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a...	2
4	pret	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a...	3

4.2.2 Etape 2 : Ajout d'une fenêtre d'authentification

Explication du travail réalisé :

L'application a été modifiée pour que la fenêtre d'authentification s'ouvre dès le démarrage de l'application, pour contrôler l'authentification une méthode a été créée dans le modèle qui renvoie 1 dans le cas où le hash du mot de passe entrée et le login correspondent au hash et au login dans la base de données.

Bilan du travail réalisé :

```
string req = "select * from utilisateur";  
req += " where login=@login and password=SHA2(@pwd, 256);";
```

4.2.3 Etape 3 et 4 : Mise en place des niveaux d'habilitations

Explication du travail réalisé :

Grâce à la méthode dans le modèle on récupère l'id du service, le numéro 1 correspond à l'admin, 2 correspond au service administratif, 3 correspond au service prêt et 4 au service culture. Ces données sont récupérées dans le contrôleur, dans le cas où le service est le 4 un message d'erreur indique que les droits d'accès ne sont pas suffisants, sinon le numéro de service est envoyé à la fenêtre principale, dans le cas où le service est le 3, les onglets commandes de dvd, livres et revues sont supprimées. Le service 2 et 3 ont tous les droits.

Bilan du travail réalisé :

```

if ((service = Dao.ControleAuthentification(login, pwd)) != 0)
{
    if (service == 3 || service == 2 || service == 1) {
        if (instance != 1)
        {
            frmAuthentification.Hide();
            instance = 1;
            frameMediatek = new FrmMediatek(this, service);
            frameMediatek.ShowDialog();
        }
    }
    else
    {
        return 1;
    }
}

```

4.2.4 Etape 5 : Modification de la fenêtre d'alerte

Explication du travail réalisé :

Pour que la fenêtre d'alerte n'apparaisse que pour le service 1 et 2 une condition permet de vérifier le niveau d'accès, si l'accès est suffisant la fenêtre apparaît.

Bilan du travail réalisé :

```

private void TabCtrl_Enter(object sender, EventArgs e)
{
    if(niveau != 3)
    {
        (new Alerte(contrôle)).Show();
    }
}

```


4.3 Mission 3 : qualités, test, documentation technique et déploiement

4.3.1 Etape 1 : SonarLint

Explication du travail réalisé :

SonarLint permet d'effectuer l'analyse statique du code, en s'aidant des informations dans la fenêtre ci-dessous on peut corriger les erreurs qui affectent la qualité du code.

Bilan du travail réalisé :

⚠ S2589	Change this condition so that it does not always evaluate to 'true'.	Mediatek86	FrmMediatek.cs	2090	Actif
⚠ CS0162	Code inaccessible détecté	Mediatek86	Controle.cs	161	Actif
⚠ CS0162	Code inaccessible détecté	Mediatek86	Controle.cs	180	Actif
⚠ CS0162	Code inaccessible détecté	Mediatek86	Controle.cs	185	Actif
⚠ S1121	Extract the assignment of 'txtCmdMontantRevue.Enabled' from this expression.	Mediatek86	FrmMediatek.cs	2090	Actif
⚠ CS0665	L'assignation dans une expression conditionnelle est toujours constante ; voulez-vous utiliser == au lieu de = ?	Mediatek86	FrmMediatek.cs	2090	Actif
⚠ CS0169	Le champ 'Controle.frmAuthentification' n'est jamais utilisé	Mediatek86	Controle.cs	20	Actif

4.3.2 Etape 2 : Ajout de tests unitaires

Explication du travail réalisé :

Les tests unitaires ont été créés avec Nunit et certaines fonctions du modèle ont été testées. Lors des accès à la BDD, les tests qui modifient la BDD sont des transactions pour cela on place au début et à la fin des fonctions qui permettent de respecter l'architecture d'une transaction de cette façon les tests ne modifient pas la BDD.

Bilan du travail réalisé :

```
public void deleteCmdLivreTest()
{
    BeginTransaction();
    List<Commande> lesCommandes = Dao.GetAllCommandes();
    int nbBeforeDelete = lesCommandes.Count;
    if (nbBeforeDelete > 0)
    {
        Commande commande = lesCommandes[0];
        Dao.deleteCmdLivre(commande);
        lesCommandes = Dao.GetAllCommandes();
        Commande delCommande = lesCommandes.Find(x => x.Id.Equals(commande.Id));
        Assert.IsNull(delCommande, "Devrait réussir si commande supprimé");
        int nbAfterDelete = lesCommandes.Count;
        Assert.AreEqual(nbBeforeDelete - 1, nbAfterDelete, "Devrait réussir : une commande en moins");
    }
    EndTransaction();
}
```

2 références | 0 modification | 0 auteur, 0 modification

```
private void BeginTransaction()  
{  
    curs.ReqUpdate("SET AUTOCOMMIT=0", null);  
}
```

2 références | 0 modification | 0 auteur, 0 modification

```
private void EndTransaction()  
{  
    curs.ReqUpdate("ROLLBACK", null);  
}
```

4.3.3 Etape 3 : Ajout de logs

Explication du travail réalisé :

Les logs ont été ajoutés grâce à l'outil Serilog proposé par VSCode , pour cela on initialise le Logger (ici par défaut) et on place dans les try catch les logs en fonctions du niveau d'alerte. Les logs sont ensuite disponibles dans le fichier mentionné.

Bilan du travail réalisé :

```
Log.Logger = new LoggerConfiguration().WriteTo.File("logs/log.txt").CreateLogger();  
catch  
{  
    MessageBox.Show("Le montant doit etre numerique ", "Information");  
    Log.Information("Le montant saisi n'etait pas numérique ");  
}
```

4.3.4 Etape 4 : Création de la documentation technique

Explication du travail réalisé :













La documentation technique a été réalisé grâce à JavaDoc

4.3.5 Etape 5 : Configuration du cloud pour l'hébergement de la BDD

Explication du travail réalisé :

La base de données est hébergée à partir d'une machine Amazon-linux 2, un utilisateur a été créée pour permettre d'accéder à distance à la BDD et le pare feu à été configurée de sortes à accepter les connexions sur le port 3306

Bilan du travail réalisé :

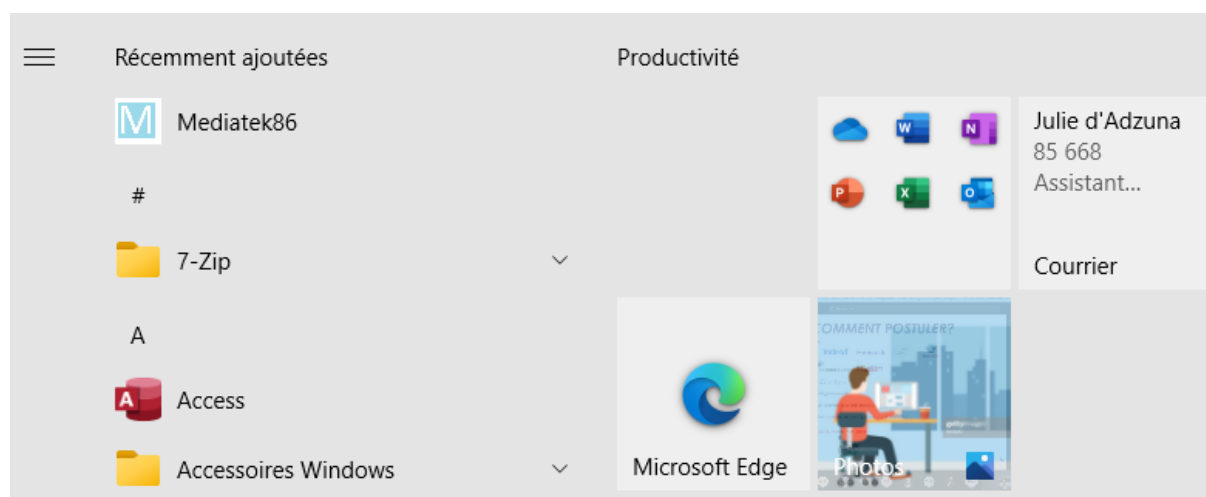
<input type="checkbox"/>	root	127.0.0.1	global	ALL PRIVILEGES	Oui	 Éditer les privilèges	 Exporte
<input type="checkbox"/>	root	:::1	global	ALL PRIVILEGES	Oui	 Éditer les privilèges	 Exporte
<input type="checkbox"/>	root	ip-172-31-85-115.ec2.internal	global	ALL PRIVILEGES	Oui	 Éditer les privilèges	 Exporte
<input type="checkbox"/>	root	localhost	global	ALL PRIVILEGES	Oui	 Éditer les privilèges	 Exporte
<input type="checkbox"/>	roots	%	global	ALL PRIVILEGES	Oui	 Éditer les privilèges	 Exporte
			spécifique à cette base de données	ALL PRIVILEGES	Non	 Éditer les privilèges	 Exporte

4.3.6 Etape 6 : Génération du MSI

Explication du travail réalisé :

Un icône a été ajouté au générateur de l'exécutable et le MSI et l'exécutable ont été générés grâce à l'extension Microsoft Visual Studio Installer Projects, une fois l'application installée 2 raccourcis seront ajoutés : l'un dans le bureau et l'autre dans le menu démarrer.

Bilan du travail réalisé :



5 Bilan

L'Administrateur du site et les utilisateurs du service administratif peuvent désormais accéder aux onglets commande de livres, commande de dvd et commande de revues, ils peuvent consulter la liste des commandes passées, les modifier et les supprimer. Ils peuvent également créer une nouvelle commande ou abonnement. Les utilisateurs du service prêt conservent la même fenêtre et mais doivent au préalable s'authentifier tandis que les utilisateurs du service culture n'ont plus accès à l'application.

6 Listes des compétences couvertes

6.1 B1 : Support et mise à disposition de services informatiques

6.1.1 B1.1

- ✓ Recenser et identifier les ressources numériques
- ✓ Mettre en place et vérifier les niveaux d'habilitation associés à un service
- ✓ Exploiter des référentiels, normes et standards adoptés par le prestataire informatique

6.1.2 B1.2

- ✓ Traiter des demandes concernant les applications
- ✓ Traiter des demandes concernant les services réseau et système, applicatifs
- ✓ Collecter, suivre et orienter des demandes

6.1.3 B1.3

- ✓ Participer à l'évolution d'un site Web exploitant les données de l'organisation

6.1.4 B1.4

- ✓ Analyser les objectifs et les modalités d'organisation d'un projet
- ✓ Planifier les activités

6.1.5 B1.5

- ✓ Déployer un service
- ✓ Réaliser les tests d'intégration et d'acceptation d'un service
- ✓ Accompagner les utilisateurs dans la mise en place d'un service
- ✓ Mettre en place et vérifier les niveaux d'habilitation associés à un service

6.2 B2 : BLOC 2 – SLAM – Conception et développement d'applications

6.2.1 B2.1

- ✓ Analyser un besoin exprimé et son contexte juridique.
- ✓ Modéliser une solution applicative.
- ✓ Participer à la conception de l'architecture d'une solution applicative.
- ✓ Rédiger des documentations techniques et d'utilisation d'une solution applicative.
- ✓ Identifier, développer, utiliser ou adapter des composants logiciels.

- ✓ Intégrer en continu les versions d'une solution applicative.
- ✓ Réaliser les tests nécessaires à la validation ou à la mise en production d'éléments adaptés ou développés.
- ✓ Exploiter les fonctionnalités d'un environnement de développement et de tests.
- ✓ Utiliser des composants d'accès aux données

6.2.2 B2.2

- ✓ Recueillir, analyser et mettre à jour les informations sur une version d'une solution applicative.
- ✓ Analyser et corriger un dysfonctionnement.
- ✓ Mettre à jour des documentations technique et d'utilisation d'une solution applicative.
- ✓ Évaluer la qualité d'une solution applicative.
- ✓ Élaborer et réaliser les tests des éléments mis à jour.

6.2.3 B2.3

- ✓ Exploiter des données à l'aide d'un langage de requêtes.
- ✓ Concevoir ou adapter une base de données.

6.3 B3 : SLAM - Cybersécurité des services informatiques – 2e année

6.3.1 B3.3

- ✓ Gérer les accès et les privilèges appropriés.
- ✓ Identifier les menaces et mettre en œuvre les défenses appropriées.
- ✓ Vérifier l'efficacité de la protection.

6.3.2 B3.5

- ✓ Participer à la vérification des éléments contribuant à la sûreté d'un développement informatique
- ✓ Prendre en compte la sécurité dans un projet de développement d'une solution applicative

- ✓ Mettre en œuvre et vérifier la conformité d'une solution applicative et de son développement à un référentiel, une norme ou un standard de sécurité
- ✓ Prévenir les attaques
- ✓ Analyser les connexions (logs)
- ✓ Analyser les incidents de sécurité, proposer et mettre en œuvre des contre-mesures