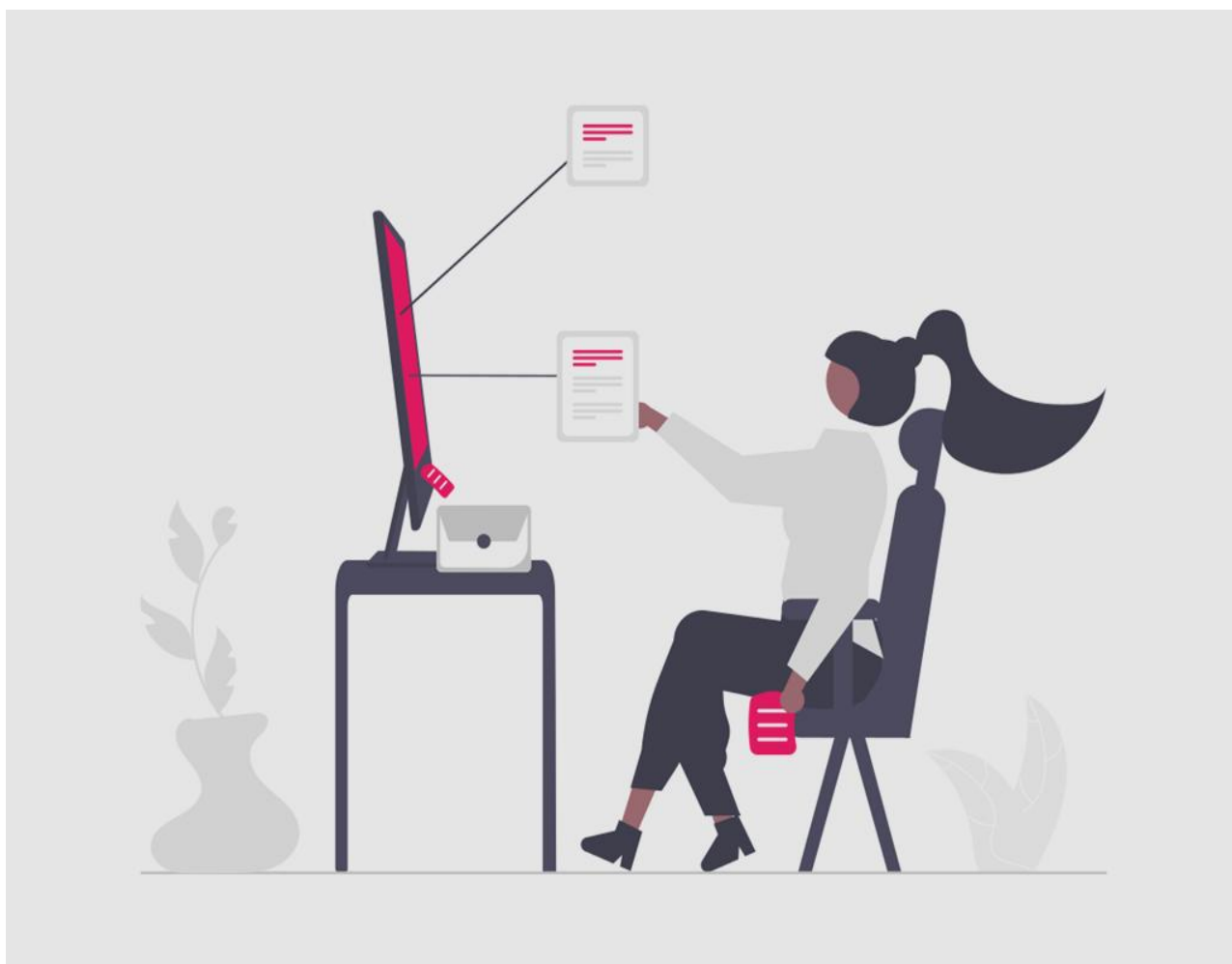


**PETIT
Marie-Camille**

Compte rendu

Mediatek86

Application en C#



Sommaire

Accès à l'application.....	3
Contexte.	4
Présentation des missions.....	5
Mission 1 (optionnelle) : Gérer les documents.....	5
Mission 2 : Gérer les commandes.....	5
Mission 3 : Gérer le suivi de l'état des documents.	6
Mission 4 : Mettre en place des authentifications.	6
Mission 5 : Qualité, test et documentation technique.	6
Mission 6 : Création d'une vidéo de présentation.	6
Ressources logicielles utilisées.	7
Mission 0 : Préparer l'environnement de travail.	8
Mission 2 : Gérer les commandes.	11
Création de la table 'suivi'.	12
Création des onglets livres et DVD et implémentations des zones de recherches et d'affichages d'informations.	13
Mise en place des boutons pouvant permettre de gérer les commandes (onglet livre et DVD).	19
Onglet Abonnement à une Revue.	26
Création d'un trigger pour ajouter les exemplaires automatiquement.	29
Création d'un trigger pour contrôler la contrainte de partition de l'héritage sur commande.	30
Création de la fenêtre d'alerte de fin d'abonnements.	30
Bilan de la mission 2.	33
Mission 4 : Mettre en place des authentifications.....	34
Création des tables 'Utilisateur' et 'Service'.	34
Création de la fenêtre d'authentification et accès aux différents services.	35
Bilan de la mission 4.	40
Mission 5 : Qualité, tests et documentation technique.....	41
Contrôle de qualité avec SonarLint.....	41
Création d'un test fonctionnel avec SpecFlow.....	42
Journalisation.	44
Documentation technique.....	45
Bilan.....	46
Déploiement de l'application.....	46
Mission 6 : Créer une vidéo de démonstration.	46
Bilan Final.	47
Compétences mobilisées.	48

Accès à l'application.

1. Installation.

- Télécharger le fichier 'Mediatek86.zip' depuis mon site portfolio.
- Installer l'application avec l'installateur 'Mediatek86 Installer.msi'.

1. Compte utilisateur.

Utilisateur	Mot de passe
admin	admin
serviceadmin	serviceadmin
pret	pret
culture	culture

2. Accès à la base de données.

- **adresse :** mediatek86projet.mysql.database.azure.com
- **Utilisateur :** AdminMediatek
- **Mot de passe :** BTSSioCNED86

Le script de la base de données est récupérable sur le dépôt 'GitHub'.

Contexte.

La médiathèque Mediatek86 possède une application qui permet de gérer différents documents (livres, DVD, Revues). Pour le moment l'application permet seulement de consulter le catalogue grâce aux recherches et aux filtrages. Chaque document possède des informations (auteur, ISBN, rayons, etc..) en fonction du type de document.

Dans cet atelier, je vais m'occuper des besoins formulés afin de faire évoluer cette application. Cet atelier est volumineux, il est conçu pour être réalisé par plusieurs personnes.

J'ai réalisé cet atelier seule, dans ce cas deux missions sont optionnelles :

- Mission 1 = Gérer les documents ;
- Mission 3 = Gérer le suivi de l'état des documents.

Présentation des missions.

Mission 1 (optionnelle) : Gérer les documents.

La première mission demandée était d'ajouter des boutons dans les onglets existant dans l'application de base qui permettent d'ajouter, modifier ou de supprimer. Il y avait quelques contraintes à respecter :

- Il n'est pas possible de supprimer un document si un exemplaire ou une commande existe ;
- On ne doit pas pouvoir modifier l'identifiant d'un document ;

Mission 2 : Gérer les commandes.

La deuxième mission consiste à ajouter des fonctionnalités pour permettre de commander des livres / DVD ou de souscrire à des abonnements de revues. On doit créer différents onglets pour chaque cas et permettre la recherche d'un document pour lequel on souhaite passer commande ou souscrire et qui affiche également des commandes ou abonnements déjà existants. Les onglets doivent ressembler à ceux déjà existants. Il faut pouvoir trier sur chaque colonne des listes. Toutes les fonctionnalités doivent être sécurisées.

Il doit être possible de créer de nouvelles commandes pour les livres et DVD, de modifier l'état de suivie de chaque document. Il faut donc créer une table 'Suivi' dans la base de données, qui doit être reliée à la table 'commandedocument' et qui contient tout les statuts de suivi possible.

Il existe une contrainte sur le passage d'un état (suivi), il n'est pas possible de revenir en arrière quand une commande est livrée ou réglée. Il n'est pas possible de supprimer une commande qui a été livrée et il est impossible de supprimer une commande non livrée.

Il faut créer un trigger qui ajoute automatiquement un exemplaire à la base de données quand le document est livré.

Pour les abonnements, on a la possibilité d'ajouter et de supprimer un abonnement. Et il est impossible de supprimer un abonnement quand un exemplaire y est rattaché. Il est demandé de créer une méthode 'ParutionDansAbonnement' qui vérifie cela, en comparant la date de parution avec les dates de début et de fin des abonnements et de faire un test unitaire sur cette méthode.

Pour finir cette mission, il faut créer une fenêtre d'alerte qui retourne les abonnements qui expirent dans moins de 30 jours. Cette fenêtre s'affiche dès l'ouverture de l'application.

Mission 3 : Gérer le suivi de l'état des documents.

La mission 3 doit permettre de gérer l'état des documents et donc des exemplaires. Tous les documents doivent être dans l'état 'neuve' et l'application doit permettre de changer l'état (« neuf », « usagé », « détérioré », « inutilisable »). Il doit également être possible de supprimer un exemplaire.

Mission 4 : Mettre en place des authentifications.

La quatrième mission est la mise en place d'un système d'authentification pour accéder à l'application. Il existe différents services et chacun d'entre eux ont des accès différents :

- Service « Administrateur » : Toutes les applications ;
- Service « Service administratif » : Toutes les applications
- Service « Prêts » : Consultation des documents ;
- Service « Culture » : Aucun accès.

Mission 5 : Qualité, test et documentation technique.

La cinquième mission demande de faire un contrôle de qualité de code avec SonarLint, créer des tests fonctionnels avec SpecFlow et gérer les logs dans les blocs de try/catch. Il faut générer la documentation avec des commentaires normalisés.

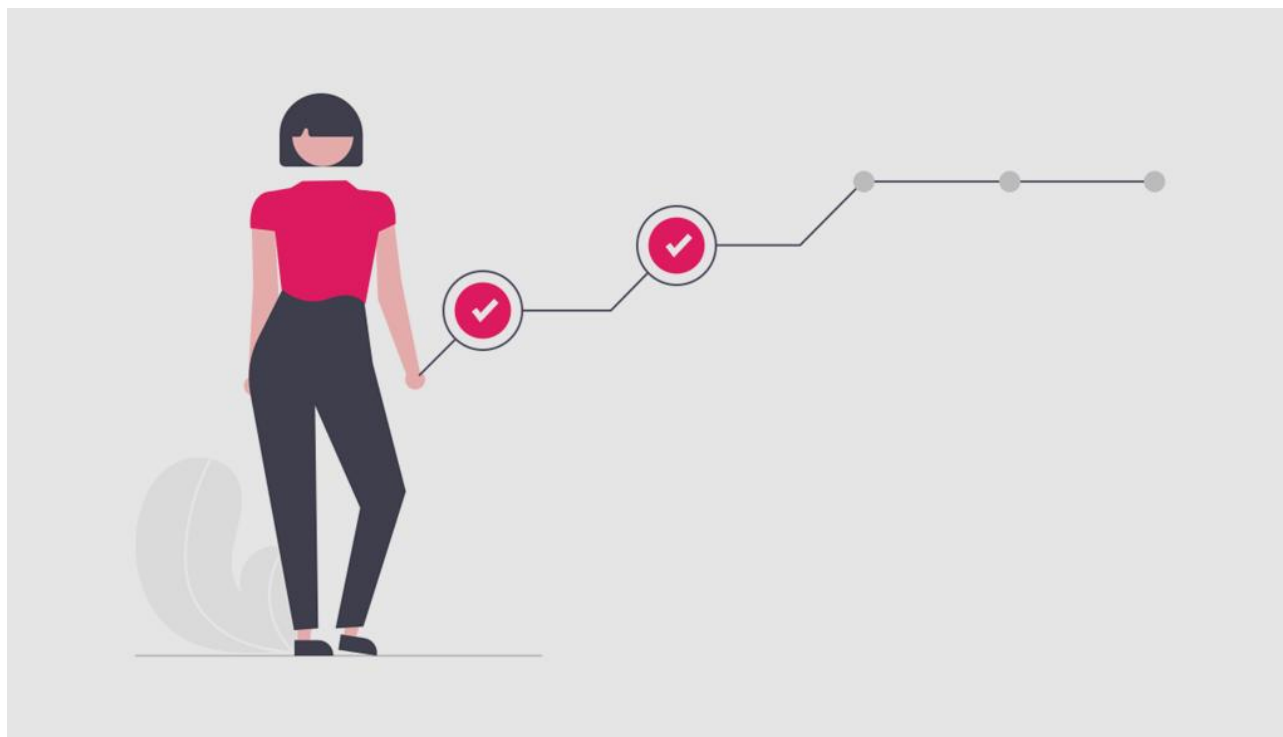
Mission 6 : Création d'une vidéo de présentation.

La sixième mission est une vidéo de moins de 10 minutes qui présente toutes les fonctionnalités de l'application.

Ressources logicielles utilisées.

Le logiciel à utiliser les ressources suivantes :

- L'application de départ est en C# ;
- L'IDE utilisé est Visual Studio 2022 avec les extensions SonarLint, GitHub, Serilog pour la journalisation et les tests fonctionnels avec SpecFlow ;
- Pendant le développement j'ai utilisé WampServer avec PhpMyAdmin ;
- La base de données est en MySQL ;
- Le versioning est fait sur GitHub ;
- J'ai créé des issues sur GitHub pour faire le suivi du projet ;
- Pour la génération de la documentation technique, j'ai utilisé SandCastle ;



Mission 0 : Préparer l'environnement de travail.

Pour commencer le projet, il faut mettre en place l'environnement du projet et tous les outils nécessaires. Les travaux demandés pendant la préparation sont les suivants :

- Avoir WampServer et l'IDE Visual Studio ;
- Récupérer le dossier documentaire et se familiariser avec celle-ci ;
- Récupérer le code source et l'ouvrir sur l'IDE ainsi que tester l'application ;

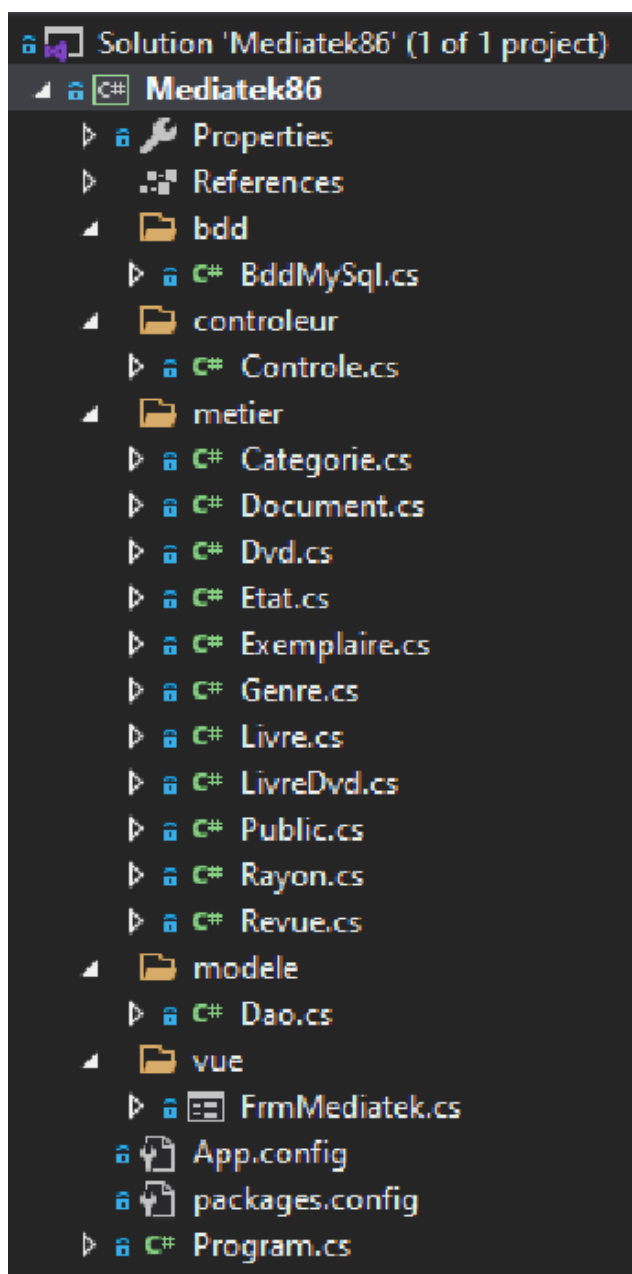


Image 1 : Projet initial sur Visual Studio

- Récupérer le script de la base de données et l'exécuter dans PhpMyAdmin ;

The screenshot shows the 'Structure' tab of a MySQL database named 'mediatek86'. It lists 13 tables with their respective actions (Browse, Structure, Search, Insert, Empty, Drop) and summary statistics (Rows, Type, Collation, Size, Overhead).

Table	Action	Rows	Type	Collation	Size	Overhead
abonnement	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
commande	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
commandedocument	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
document	Browse Structure Search Insert Empty Drop	41	InnoDB	utf8mb4_general_ci	64.0 KiB	-
dvd	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
etat	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
exemplaire	Browse Structure Search Insert Empty Drop	14	InnoDB	utf8mb4_general_ci	32.0 KiB	-
genre	Browse Structure Search Insert Empty Drop	19	InnoDB	utf8mb4_general_ci	16.0 KiB	-
livre	Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_general_ci	16.0 KiB	-
livres_dvd	Browse Structure Search Insert Empty Drop	30	InnoDB	utf8mb4_general_ci	16.0 KiB	-
public	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
rayon	Browse Structure Search Insert Empty Drop	15	InnoDB	utf8mb4_general_ci	16.0 KiB	-
revue	Browse Structure Search Insert Empty Drop	11	InnoDB	utf8mb4_general_ci	16.0 KiB	-
13 tables	Sum	168	MyISAM	utf8_unicode_ci	304.0 KiB	0 B

Image 2 : Base de données initiale sur PhpMyAdmin

- Créer un dépôt sur GitHub et faire le premier commit;

The screenshot shows a GitHub repository page for a user named 'MarieCamillePetit'. It displays a list of files and folders, all of which were committed for the first time ('Premier Commit') 5 days ago.

File/Folder	Commit Message	Time
Properties	Premier Commit	5 days ago
bdd	Premier Commit	5 days ago
controleur	Premier Commit	5 days ago
metier	Premier Commit	5 days ago
modele	Premier Commit	5 days ago
vue	Premier Commit	5 days ago
.gitattributes	Premier Commit	5 days ago
.gitignore	Premier Commit	5 days ago
App.config	Premier Commit	5 days ago
Mediatek86.csproj	Premier Commit	5 days ago
Mediatek86.sln	Premier Commit	5 days ago
Program.cs	Premier Commit	5 days ago
packages.config	Premier Commit	5 days ago

Image 3 : Premier commit

- Mettre en place un suivi de projet.

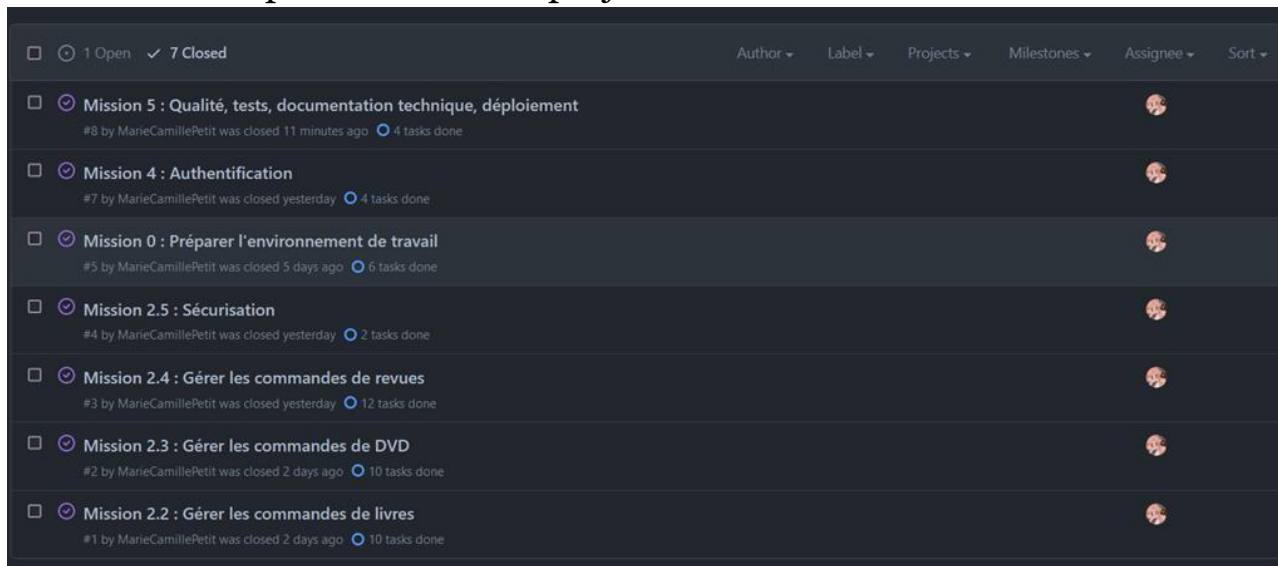
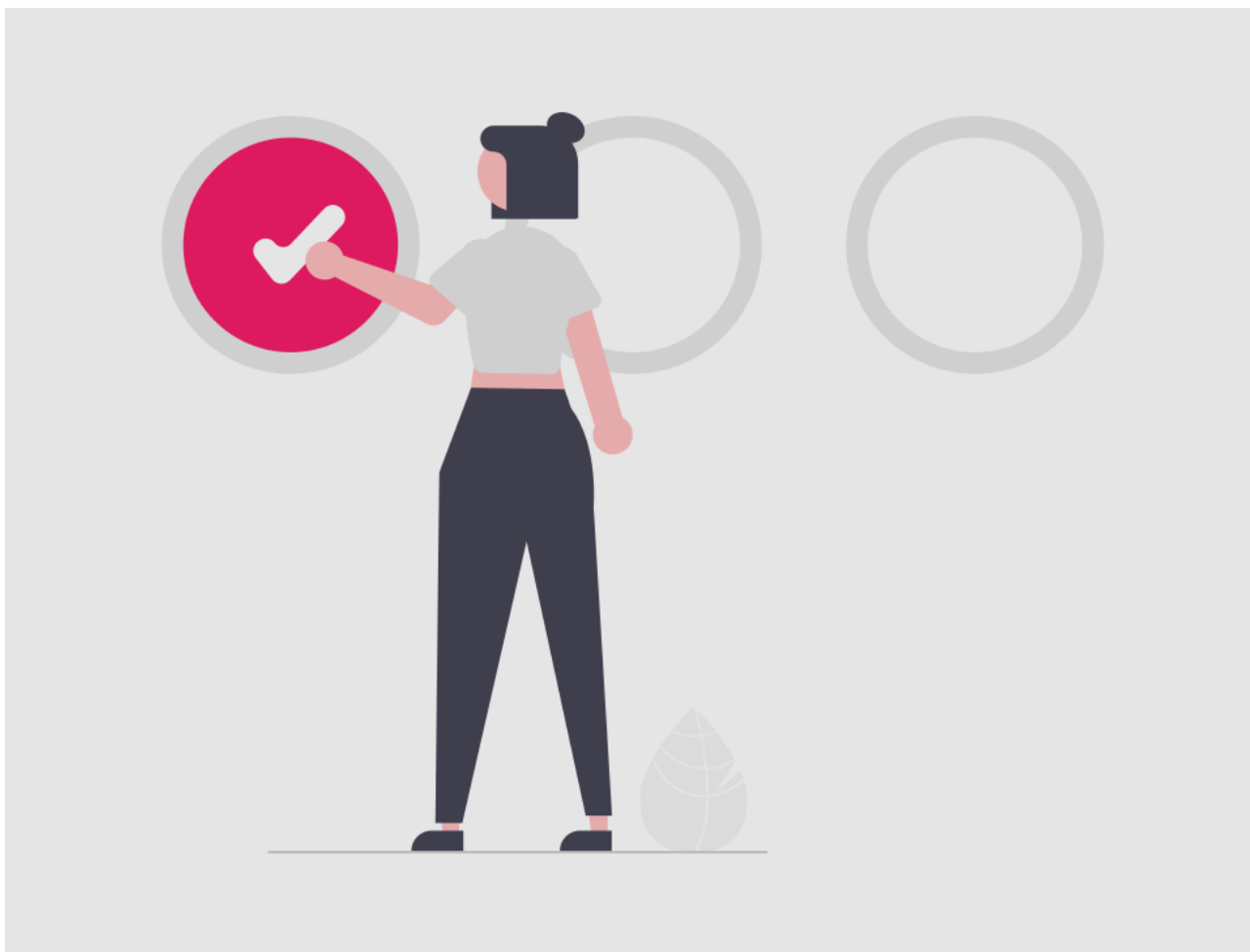


Image 4 : Suivi de projet sur GitHub.



Mission 2 : Gérer les commandes.

Les tâches suivantes m'ont été confiées pour la mission 2 :

Création d'une table 'Suivi' dans la base de données avec différents état de suivi et la relier avec CommandeDocument :

- « En cours » ;
- « Relancée » ;
- « Livrée » ;
- « Réglée » ;

Créer un onglet pour les commandes de livres :

- Dans cet onglet, il faut pouvoir retrouver un livre en fonction de son numéro ;
- Afficher toutes les informations du livre ;
- Afficher la liste des commandes de livres (date de commande, montant, nombres d'exemplaires, étape de suivi) :
- Pouvoir trier la liste ;
- Pouvoir saisir et enregistrer une nouvelle commande ;
- Changer l'état de suivi de la commande (une nouvelle commande passe à l'état « En cours ») ;
- Création de trigger pour l'ajout automatique d'exemplaires lorsque la commande passe à l'état « Livrée » ;

Créer un onglet pour les commandes de DVD avec les mêmes fonctionnalités que l'onglet commandes de livres ;

Créer un onglet de gestion d'abonnement aux revues :

- Affichage des informations et des abonnements d'une revue en fonction de son numéro avec la possibilité de tri sur les colonnes ;
- Pouvoir saisir et enregistrer un nouvel abonnement ;
- Supprimer un abonnement si aucun exemplaire n'est rattaché ;

Permettre le tri sur les listes de commandes ou abonnements ;

Création de trigger pour contrôler la contrainte de partition de l'héritage sur commande ;

Création de procédure stockée dans 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, la fenêtre d'alerte apparaît avec la liste d'abonnement se finissant dans moins de 30 jours avec la possibilité de trier sur la date dans l'ordre chronologique.

Création de la table 'suivi'.

Pour commencer cette mission, j'ai créé la table 'Suivi' avec les champs :

- id ;
- libellé ;

```
CREATE TABLE IF NOT EXISTS `suivi` (
  `id` int(3) NOT NULL,
  `libelle` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Image 5 : Création de la table 'Suivi' dans la base de données

J'ai créé une table 'SuiviCommandeDoc' avec une clé primaire de 'idSuivi' et 'idCommande' pour pouvoir suivre l'état de suivi d'une commande.

```
CREATE TABLE IF NOT EXISTS `suivicommandedoc` (
  `idsuivi` int(3) NOT NULL,
  `idcommande` varchar(5) NOT NULL,
  PRIMARY KEY (`idsuivi`,`idcommande`),
  KEY `suivicommandedoc_fk2` (`idcommande`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Image 6 : Création de la table 'suivicommandedoc' dans la base de données

J'ai ensuite rempli la table 'Suivi' et créé les contraintes nécessaires.

```
INSERT INTO `suivi` (`id`, `libelle`) VALUES
(1, 'En cours'),
(2, 'Relancée'),
(3, 'Livrée'),
(4, 'Réglée');
```

Image 7 : Remplissage de la table 'suivi' dans

```
ALTER TABLE suivi
  ADD PRIMARY KEY (id);

ALTER TABLE suivicommandedoc
  ADD PRIMARY KEY (idsuivi, idcommande);

ALTER TABLE suivicommandedoc
  ADD CONSTRAINT suivicommandedoc_fk1 FOREIGN KEY (idsuivi) REFERENCES suivi (id),
  ADD CONSTRAINT suivicommandedoc_fk2 FOREIGN KEY (idcommande) REFERENCES commandedocument (id);
```

Image 8 : Création des « Key » et des contraintes.

Création des onglets livres et DVD et implémentations des zones de recherches et d'affichages d'informations.

Les onglets « Commande d'un livre » et « Commande d'un DVD » se ressemblent fortement, donc je vais regrouper les deux onglets dans ce compte rendu car les fonctions restent les mêmes (sauf avec 2 ou 3 changements de nom). J'ai également créé l'onglet « Abonnement à une Revue » car la méthode de recherche ressemble à celle utilisé pour les autres onglets.

The screenshot shows a web application titled "Gestion Médiathèque". The top navigation bar includes tabs: "Livres", "DVD", "Revue", "Parutions des revues", "Commande d'un livre" (selected), "Commande d'un DVD", and "Abonnement à une Revue".

The main content area is divided into three sections:

- Rechercher un livre:** Contains search fields for "Numéro du livre", "Titre", "Auteur(e)", "Collection", "Genre", "Public", "Rayon", and "Chemin de l'image". There is a "Rechercher" button and an "ISBN" field.
- Nouvelle commande:** Contains fields for "Numéro commande", "Date de commande" (set to 04/04/2022), "Exemplaire" (set to 1), and "Montant". Below these are "Enregistrer" and "Annuler" buttons.
- Gestions des commandes:** Contains five buttons: "Ajouter", "Relancer", "Confirmer", "Régler", and "Supprimer".

Image 9 : Onglet de « commande d'un livre ».

Gestion Médiathèque

Livres DVD Revues Parutions des revues Commande d'un livre **Commande d'un DVD** Abonnement à une Revue

Informations détaillées

Numéro de DVD : Rechercher Durée :

Titre :

Réalisateur(trice) :

Synopsis :

Genre :

Public :

Rayon :

Chemin de l'image :

Commandes :

Nouvelle commande

Numéro commande : Exemple : 1

Date de commande : 04/04/2022 Montant :

Enregistrer

Annuler

Gestions des commandes

Ajouter Relancer Confirmer Régler Supprimer

Image 10 : Onglet de « commande d'un DVD ».

Gestion Médiathèque

Livres | DVD | Revues | Parutions des revues | Commande d'un livre | Commande d'un DVD | **Abonnement à une Revue**

Recherche revue

Numéro revue : **Rechercher** Empruntable : ☐

Titre :

Périodicité :

Délai mise à dispo :

Genre :

Public :

Rayon :

Chemin de l'image :

Abonnements :

Détails de l'abonnement

Numéro commande : Début abonnement : 04/04/2022

Montant : Fin abonnement : 04/04/2022

Gestions des commandes

Image 11 : Onglet d' « Abonnement à une Revue ».

Les méthodes pour remplir les informations sont très similaires à celle des onglets déjà existants.

```

/// <summary>
/// Recherche d'un livre à partir du numéro et affichage les informations
/// </summary>txbCommandeLivresNumeroCommande
1 référence
private void CommandeLivresRechercher()
{
    if (!txbCommandeLivreNumero.Text.Equals(""))
    {
        Livre livre = lesLivres.Find(x => x.Id.Equals(txbCommandeLivreNumero.Text.Trim()));
        if (livre != null)
        {
            AfficheCommandeLivresInfos(livre);
        }
        else
        {
            MessageBox.Show("Numéro introuvable");
            txbCommandeLivreNumero.Text = "";
            txbCommandeLivreNumero.Focus();
            VideCommandeLivresInfos();
        }
    }
    else
    {
        VideCommandeLivresInfos();
    }
}

```

Image 12: Méthode pour afficher les informations d'un livre.

Il a également fallu créer une méthode GetCommandeDocument (contrôleur et Dao) pour récupérer les commandes de la base de données.

```

/// <summary>
/// récupère les commandes d'un Livre
/// </summary>
/// <param name="idDocument">Identifiant du document concerné</param>
/// <returns>Collection d'objets de type CommandeDocument</returns>
public List<CommandeDocument> GetCommandeDocument(string idDocument)
{
    return Dao.GetCommandeDocument(idDocument);
}

```

Image 13: Appel de la méthode GetCommandeDocument du Dao.


```

/// <summary>
/// Retourne les commandes d'un livre à partir de la BDD
/// </summary>
/// <param name="idDoc">Identifiant du livre</param>
/// <returns>Retourne la liste d'objets CommandeDocument</returns>
1 référence
public static List<CommandeDocument> GetCommandeDocument(string idDoc)
{
    List<CommandeDocument> lesCommandes = new List<CommandeDocument>();
    string req = "Select c.id, c.dateCommande, c.montant, cd.nbExemplaire, cd.idLivreDvd, scd.idSuivi, s.libelle ";
    req += "from commande c join commandedocument cd on c.id=cd.id ";
    req += "join suivicommandedoc scd on c.id=scd.idcommande ";
    req += "join suivi s on scd.idsuivi = s.id ";
    req += "where cd.idLivreDvd = @id ";
    req += "order by c.dateCommande DESC";
    Dictionary<string, object> parameters = new Dictionary<string, object>
    {
        { "@id", idDoc }
    };

    BddMySQL curs = BddMySQL.GetInstance(connectionString);
    curs.ReqSelect(req, parameters);

    while (curs.Read())
    {
        string id = (string)curs.Field("id");
        DateTime dateCommande = (DateTime)curs.Field("datecommande");
        double montant = (double)curs.Field("montant");
        int nbExemplaire = (int)curs.Field("nbExemplaire");
        string idLivreDvd = (string)curs.Field("idLivreDvd");
        int idSuivi = (int)curs.Field("idSuivi");
        string libelleSuivi = (string)curs.Field("libelle");
        CommandeDocument commandeDocument = new CommandeDocument(id, dateCommande, montant, nbExemplaire, idLivreDvd, idSuivi, libelleSuivi);
        lesCommandes.Add(commandeDocument);
    }
    curs.Close();
    return lesCommandes;
}

```

Image 14: Récupération des commandes (Dao).

Pour remplir la liste des commandes j'ai utilisé la fonction suivante :

```

/// <summary>
/// Remplissage du datagrid avec la collection CommandeDocument
/// </summary>
/// <param name="lesCommandeDocument">Collection de CommandeDocument</param>
private void RemplirCommandeLivresListe(List<CommandeDocument> lesCommandeDocument)
{
    bdgCommandesLivresListe.DataSource = lesCommandeDocument;
    dgvCommandeLivresListe.DataSource = bdgCommandesLivresListe;
    dgvCommandeLivresListe.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
    dgvCommandeLivresListe.Columns["Id"].Visible = false;
    dgvCommandeLivresListe.Columns["IdSuivi"].Visible = false;
    dgvCommandeLivresListe.Columns["IdLivreDvd"].Visible = false;
    dgvCommandeLivresListe.Columns["DateCommande"].DisplayIndex = 0;
    dgvCommandeLivresListe.Columns[5].HeaderCell.Value = "Date";
    dgvCommandeLivresListe.Columns[5].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dgvCommandeLivresListe.Columns["Montant"].DisplayIndex = 1;
    dgvCommandeLivresListe.Columns[6].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dgvCommandeLivresListe.Columns[6].DefaultCellStyle.Format = "c2";
    dgvCommandeLivresListe.Columns[6].DefaultCellStyle.FormatProvider = CultureInfo.GetCultureInfo("fr-FR");
    dgvCommandeLivresListe.Columns[0].HeaderCell.Value = "Exemplaires";
    dgvCommandeLivresListe.Columns[0].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dgvCommandeLivresListe.Columns[2].HeaderCell.Value = "Etat";
    dgvCommandeLivresListe.Columns[2].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
}

```

Image 15: Remplissage de la liste de commandes.

J'ai inséré manuellement une commande de livres pour voir le résultat. Après avoir adapté les méthodes (qui étaient similaires à celles utilisées dans l'onglet des « Parutions des revues ») pour permettre les recherches d'un livre ou d'un DVD ainsi que l'affichage des commandes. J'ai inséré manuellement une commande de livre pour voir le résultat. J'ai ensuite pu tester l'application :


Gestion Médiathèque

Livres DVD Revues Parutions des revues **Commande d'un livre** Commande d'un DVD Abonnement à une Revue

Rechercher un livre

Numéro du livre : 00001 **Rechercher** ISBN : 1234569877896
 Titre : Quand sort la recluse
 Auteur(e) : Fred Vargas
 Collection : Commissaire Adamsl
 Genre : Policier
 Public : Adultes
 Rayon : Policiers français étrangers
 Chemin de l'image : C:\Users\petit\OneDrive\Bureau\ATELIERDEPRO2\Atelier3\images\Quand-sort
 Commandes :

Date	Montant	Exemplaires	Etat
04/04/2022	100,00 €	1	En cours



Nouvelle commande

Numéro commande : Exemple : 1
 Date de commande : 04/04/2022 Montant :
 Enregistrer
 Annuler

Gestions des commandes

Ajouter Relancer Confirmer Régler Supprimer

Image 16 : Affichage des informations et commande d'un livre.

Mise en place des boutons pouvant permettre de gérer les commandes (onglet livre et DVD).

Pour la sécurité de l'application j'ai mis en place un système d'activation et de désactivation des différents boutons.

```

/// <summary>
/// Événement clic sur le bouton d'ajout de commande
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnCommandeLivresAjouter_Click(object sender, EventArgs e)
{
    AccesDetailsCommandeLivres(true);
    AccesModificationCommandeLivres(true);
}

/// <summary>
/// Activation de la zone détails d'une nouvelle commande
/// </summary>
/// <param name="acces"></param>
5 références
private void AccesDetailsCommandeLivres(bool acces)
{
    VideDetailsCommandeLivres();
    grpCommandeLivres.Enabled = acces;
    txbCommandeLivresNumeroCommande.Enabled = acces;
    datepickCommandeLivresDateCommande.Enabled = acces;
    numCommandeLivresExemplaires.Enabled = acces;
    txbCommandeLivresMontant.Enabled = acces;
    btnCommandeLivresEnregistrer.Enabled = acces;
    btnCommandeLivresAnnuler.Enabled = acces;
    btnCommandeLivresAjouter.Enabled = !acces;
}

```

Image 17 : Activation ou désactivation de bouton en fonction de la situation.

Cependant il fallait gérer les contraintes au niveau du suivi d'une commande, désactiver certains boutons en fonction de demandes du dossier documentaires :

```

/// <summary>
/// Active/Désactive les boutons de gestion de commande en fonction de l'état de suivi
/// </summary>
/// <param name="commandeDocument">CommandeDocument concernée</param>
1 référence
private void AccesBtnModificationCommandeLivres(CommandeDocument commandeDocument)
{
    string etatSuivi = commandeDocument.LibelleSuivi;
    switch (etatSuivi)
    {
        case "En cours":
        case "Relancée":
            btnCommandeLivresRelancer.Enabled = true;
            btnCommandeLivresConfirmer.Enabled = true;
            btnCommandeLivresRegler.Enabled = false;
            btnCommandeLivresSupprimer.Enabled = true;
            break;
        case "Livrée":
            btnCommandeLivresRelancer.Enabled = false;
            btnCommandeLivresConfirmer.Enabled = false;
            btnCommandeLivresRegler.Enabled = true;
            btnCommandeLivresSupprimer.Enabled = false;
            break;
        case "Réglée":
            AccesModificationCommandeLivres(false);
            break;
    }
}

```

Image 18 : Activation ou désactivation des boutons en fonction du suivi de la commande.

J'ai également créé trois nouvelles classes métiers dans l'application pour Commande, Commandedocument et Suivi.

```

▷ C# Commande.cs
▷ C# CommandeDocument.cs

```

Image 19 : Classe métier 'Commande' et 'Commandedocument'.

```

▷ C# Suivi.cs

```

Image 20: Classe métier 'Suivi'.

Ensuite, pour créer les différentes interactions avec la base de données, j'ai récupéré tous les 'Suivi' dans la méthode « GetAllSuivi ». Cette méthode était similaire à celle pour récupérer GetAllGenres, GetAllPublics et GetAllRayon.

```

/// <summary>
/// Retourne tous les stades de suivi à partir de la BDD
/// </summary>
/// <returns>Collection d'objets Suivi</returns>

public static List<Suivi> GetAllSuivis()
{
    List<Suivi> lesSuivis = new List<Suivi>();
    string req = "Select * from suivi";

    BddMySQL curs = BddMySQL.GetInstance(connectionString);
    curs.ReqSelect(req, null);

    while (curs.Read())
    {
        Suivi leSuivi = new Suivi((int)curs.Field("id"), (string)curs.Field("libelle"));
        lesSuivis.Add(leSuivi);
    }
    curs.Close();
    return lesSuivis;
}

```

Image 21: Méthode 'GetAllSuivis'.

J'ai également du créer des propriétés 'lesSuivis' pour le type Liste de Suivi dans le contrôleur et la vue.

```

/// <summary>
/// récupère les suivis
/// </summary>
2 références
public List<Suivi> GetAllSuivis()
{
    return lesSuivis;
}

```

Image 22: Création de la liste Suivi contenant tous les suivis.

Pour ajouter une nouvelle commande j'ai créé la méthode événementielle qui se déclenche lors d'un clic sur le bouton d'enregistrement, avec les méthodes correspondantes dans le contrôleur et le Dao.

```

/// <summary>
/// Événement clic sur le bouton 'enregistrer'
/// Création d'une nouvelle commande
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnCommandeLivresEnregistrer_Click(object sender, EventArgs e)
{
    if (txbCommandeLivresNumeroCommande.Text == "" || txbCommandeLivresMontant.Text == "")
    {
        MessageBox.Show("Vous devez remplir tous les champs.", "Information");
        return;
    }

    String id = txbCommandeLivresNumeroCommande.Text;
    DateTime dateCommande = datepickCommandeLivresDateCommande.Value;
    int nbExemplaires = (int)numCommandeLivresExemplaires.Value;
    string idLivreDvd = txbCommandeLivreNumero.Text.Trim();
    int idSuivi = lesSuivis[0].Id;
    string libelleSuivi = lesSuivis[0].Libelle;
    String montantSaisie = txbCommandeLivresMontant.Text.Replace(',', '.');

    // validation du champ montant
    if (!Double.TryParse(montantSaisie, out double montant))
    {
        MessageBox.Show("Le montant doit être numérique.", "Information");
        txbCommandeLivresMontant.Text = "";
        txbCommandeLivresMontant.Focus();
        return;
    }

    CommandeDocument laCommandeDocument = new CommandeDocument(id, dateCommande, montant, nbExemplaires, idLivreDvd, idSuivi, libelleSuivi);
    if (txbCommandeLivresNumeroCommande.Text.Length <= 5)
    {
        if (controle.CreerCommandeDocument(laCommandeDocument))
        {
            AfficheCommandeDocumentLivre();
            int addedRowIndex = -1;
            DataGridViewRow row = dgvCommandeLivresListe.Rows
                .Cast<DataGridViewRow>()
                .First(r => r.Cells["id"].Value.ToString().Equals(id));
            addedRowIndex = row.Index;
            dgvCommandeLivresListe.Rows[addedRowIndex].Selected = true;
        }
        else
        {
            MessageBox.Show("Le numéro de commande existe déjà.", "Information");
            txbCommandeLivresNumeroCommande.Text = "";
            txbCommandeLivresNumeroCommande.Focus();
        }
        AccesDetailsCommandeLivres(false);
        AccesGestionCommandeLivres(true);
    }
    else
    {
        MessageBox.Show("Le numéro de commande ne doit pas dépasser 5 caractères.", "Information");
        txbCommandeLivresNumeroCommande.Text = "";
        txbCommandeLivresNumeroCommande.Focus();
    }
}

```

Image 23 : Méthode événementielle d'enregistrement d'une commande.

Dans le même principe j'ai pu créer la méthode événementielle, ainsi que les méthodes correspondantes dans le contrôleur et le Dao, pour gérer la suppression d'une commande et également celle pour modifier l'état du Suivi.

```

/// <summary>
/// Suppression d'une commande
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnCommandeLivresSupprimer_Click(object sender, EventArgs e)
{
    if (ConfirmationSupprCommande())
    {
        CommandeDocument commandeDocument = (CommandeDocument)bdgCommandesLivresListe.Current;
        if (controle.SupprCommandeDocument(commandeDocument.Id))
        {
            AfficheCommandeDocumentLivre();
        }
        else
        {
            MessageBox.Show("Une erreur s'est produite.", "Erreur");
        }
    }
}

```

Image 24 : Méthode événementielle de suppression d'une commande.

```

/// <summary>
/// Modifie l'état de la commande à : relancée
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnCommandeLivresRelancer_Click(object sender, EventArgs e)
{
    CommandeDocument commandeDocument = (CommandeDocument)bdgCommandesLivresListe.List[bdgCommandesLivresListe.Position];
    Suivi nouveauSuivi = lesSuivis.Find(suivi => suivi.Libelle == "Relancée");
    ModifEtatSuiviCommandeDocumentLivre(commandeDocument.Id, nouveauSuivi);
}

/// <summary>
/// Modifie l'état de la commande à : livrée
/// Notifie la création des exemplaires
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnCommandeLivresConfirmerLivraison_Click(object sender, EventArgs e)
{
    CommandeDocument commandeDocument = (CommandeDocument)bdgCommandesLivresListe.List[bdgCommandesLivresListe.Position];
    Suivi nouveauSuivi = lesSuivis.Find(suivi => suivi.Libelle == "Livrée");
    if (ModifEtatSuiviCommandeDocumentLivre(commandeDocument.Id, nouveauSuivi))
    {
        MessageBox.Show("Les exemplaires ont été ajoutés dans la base de données.", "Information");
    }
}

/// <summary>
/// Modifie l'état de la commande à : réglée
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnCommandeLivresRegler_Click(object sender, EventArgs e)
{
    CommandeDocument commandeDocument = (CommandeDocument)bdgCommandesLivresListe.List[bdgCommandesLivresListe.Position];
    Suivi nouveauSuivi = lesSuivis.Find(suivi => suivi.Libelle == "Réglée");
    ModifEtatSuiviCommandeDocumentLivre(commandeDocument.Id, nouveauSuivi);
}

```

Image 25 : Méthodes événementielles pour les changements d'état.

```

/// <summary>
/// Demande de modification de l'état de suivi au contrôleur après validation utilisateur
/// </summary>
/// <param name="idCommandeDocument">identifiant du document concerné</param>
/// <param name="nouveauSuivi">nouvel état de suivi</param>
/// <returns>True si modification a réussi</returns>
3 références
private bool ModifEtatSuiviCommandeDocumentLivres(string idCommandeDocument, Suivi nouveauSuivi)
{
    if (ConfirmationModifSuiviCommande(nouveauSuivi.Libelle))
    {
        if (controle.ModifSuiviCommandeDocument(idCommandeDocument, nouveauSuivi.Id))
        {
            AfficheCommandeDocumentLivres();
            return true;
        }
        else
        {
            MessageBox.Show("Une erreur s'est produite.", "Erreur");
            return false;
        }
    }
    return false;
}

```

Image 26 : Demande de changements d'état de suivi au contrôleur.

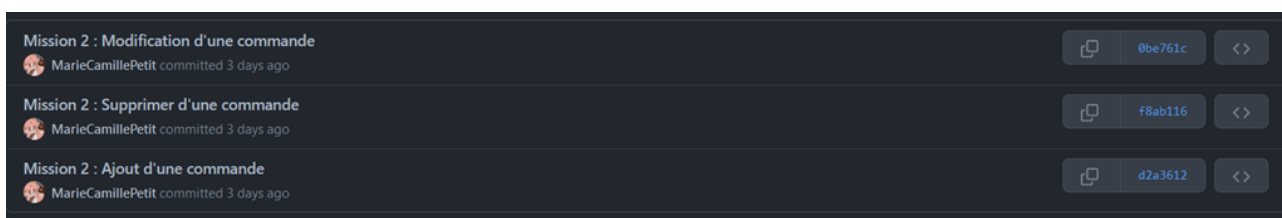


Image 27 : Commit de la partie commande de livre.

En recopiant et en modifiant les fonctions, je les ai adaptées pour pouvoir correspondre à l'onglet DVD.

Gestion Médiathèque

Livres | DVD | Revues | Parutions des revues | Commande d'un livre | Commande d'un DVD | Abonnement à une Revue

Informations détaillées

Numéro de DVD : Durée :

Titre :

Réalisateur(trice) :

Synopsis :

Genre :

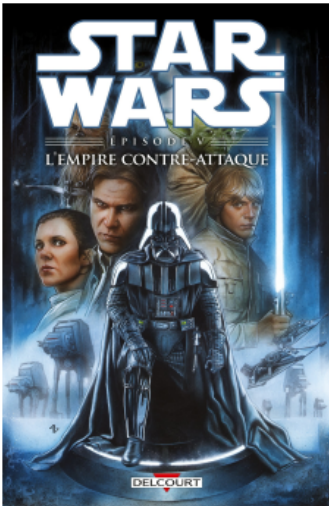
Public :

Rayon :

Chemin de l'image :

Commandes :

Date	Montant	Exemplaires	Etat
02/04/2022	55,00 €	1	En cours
02/04/2022	66,00 €	1	Réglée
01/04/2022	66,66 €	1	Réglée
01/04/2022	10,00 €	3	Réglée



Nouvelle commande

Numéro commande : Exemple :

Date de commande : Montant :

Gestions des commandes

Image 28 : Onglet DVD avec les différentes fonctionnalités.

Onglet Abonnement à une Revue.

L'onglet abonnement ressemble également aux onglets précédents. Il reprend les mêmes fonctions pour l'affichage des informations (avec quelques différences au niveau du code).

Gestion Médiathèque

Livres DVD Revue Parutions des revues Commande d'un livre Commande d'un DVD Abonnement à une Revue

Recherche revue

Numéro revue : 10006 Rechercher Empruntable : ☐

Titre : Le Monde

Périodicité : QT

Délai mise à dispo : 5

Genre : Actualités

Public : Adultes

Rayon : Presse quotidienne

Chemin de l'image : C:\Users\petit\OneDrive\Bureau\ATELIERDEPRO2\Atelier3\images\lemonde.jpg

Abonnements :

Date commande	Montant	Date fin abonnement

Détails de l'abonnement

Numéro commande : Montant : Début abonnement : 04/04/2022 Fin abonnement : 04/04/2022

Enregistrer Annuler

Gestions des commandes

Ajouter Supprimer

Image 29 : Onglet d'abonnement à une revue avec l'affichage des informations.

Pour créer un nouvel abonnement j'ai créé une méthode événementielle sur le bouton d'enregistrement, cette méthode ressemblait également à celle des onglets précédents mais il fallait vérifier les dates de commandes et de fin d'abonnements.

```

/// <summary>
/// Clic sur "enregistrer" permet la création d'un nouveau abonnement
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void btnAboRevueEnregistrer_Click(object sender, EventArgs e)
{
    if (txbAboRevueNumeroDetails.Text == "" || txbAboRevueNumeroMontant.Text == "")
    {
        MessageBox.Show("Veuillez remplir tous les champs.", "Information");
        return;
    }

    if (DateTime.Compare(dtpAboRevueDateCommande.Value, dtpAboRevueFinCommande.Value) >= 0)
    {
        MessageBox.Show("La date de fin d'abonnement n'est pas valide.", "Information");
        dtpAboRevueFinCommande.Value = DateTime.Now.AddYears(1);
        dtpAboRevueFinCommande.Focus();
        return;
    }

    String id = txbAboRevueNumeroDetails.Text;
    DateTime dateCommande = dtpAboRevueDateCommande.Value;
    DateTime finAbonnement = dtpAboRevueFinCommande.Value;
    string idRevue = txbCommandeRevueNumero.Text.Trim();
    String montantSaisie = txbAboRevueNumeroMontant.Text.Replace('.', ',');

    if (!Double.TryParse(montantSaisie, out double montant))
    {
        MessageBox.Show("Le montant doit être numérique.", "Information");
        txbAboRevueNumeroMontant.Text = "";
        txbAboRevueNumeroMontant.Focus();
        return;
    }

    Abonnement abonnement = new Abonnement(id, dateCommande, montant, finAbonnement, idRevue);
    if (txbAboRevueNumeroDetails.Text.Length <= 5)
    {
        if (controle.CreerAbonnement(abonnement))
        {
            AfficheAbonnementRevue();

            int addedRowIndex = -1;
            DataGridViewRow row = dgvAboRevueListe.Rows
                .Cast<DataGridViewRow>()
                .First(r => r.Cells["id"].Value.ToString().Equals(id));
            addedRowIndex = row.Index;
            dgvAboRevueListe.Rows[addedRowIndex].Selected = true;

            AccesDetailsAbonnementRevue(false);
            AccesGestionAbonnementRevue(true);
        }
        else
        {
            MessageBox.Show("Ce numéro d'abonnement existe déjà.", "Information");
            txbAboRevueNumeroDetails.Text = "";
            txbAboRevueNumeroDetails.Focus();
        }
    }
    else
    {
        MessageBox.Show("Le numéro d'abonnement ne doit pas dépasser 5 caractères.", "Information");
        txbAboRevueNumeroDetails.Text = "";
        txbAboRevueNumeroDetails.Focus();
    }
}

```

Image 30 : Enregistrement d'un nouvel abonnement.

Pour supprimer un abonnement, il faut respecter la demande du dossier document. Un abonnement ne peut être supprimé s'il y a des exemplaires rattachés. J'ai donc créé une fonction qui va comparer 3 dates différentes :

- La date de commande ;
- La date de fin d'abonnement ;
- La date de parution.

Pour pouvoir récupérer il a fallu créer deux méthodes qui récupèrent les informations de la vue et de la base de données, dans le contrôleur.

```

/// <summary>
/// Vérification si la suppression d'un abonnement est possible
/// seulement possible quand l'abonnement n'est lié à aucun exemplaire
/// </summary>
/// <param name="abonnement">Abonnement concerné</param>
/// <returns>True si la suppression est possible</returns>
1 référence
public bool CheckSupprAbonnement(Abonnement abonnement)
{
    List<Exemplaire> lesExemplaires = GetExemplairesRevue(abonnement.IdRevue);
    bool suppression = false;

    foreach (Exemplaire exemplaire in lesExemplaires.Where(ex => ParutionDansAbonnement(abonnement.DateCommande, abonnement.DateFinAbonnement, ex.DateAchat)))
    {
        suppression = true;
    }
    return !suppression;
}

/// <summary>
/// Teste si dateParution est compris entre dateCommande et dateFinAbonnement
/// </summary>
/// <param name="dateCommande">Date de commande d'un abonnement</param>
/// <param name="dateFinAbonnement">Date de fin d'un abonnement</param>
/// <param name="dateParution">Date de parution d'un exemplaire</param>
/// <returns>True si la date est comprise</returns>
1 référence
public bool ParutionDansAbonnement(DateTime dateCommande, DateTime dateFinAbonnement, DateTime dateParution)
{
    return (DateTime.Compare(dateCommande, dateParution) < 0 && DateTime.Compare(dateParution, dateFinAbonnement) < 0);
}

```

Image 31 : Vérification des dates.

Création d'un trigger pour ajouter les exemplaires automatiquement.

Le premier trigger à créer était quand une commande passe à l'étape « Livrée », ce trigger doit automatiquement ajouter les exemplaires à la base de données avec l'état « neuf » et le numéro d'exemplaire doit être séquentiel au livre concerné.

```
DROP TRIGGER IF EXISTS `updateSuiviCommandeDoc`;
DELIMITER $$
CREATE TRIGGER `updateSuiviCommandeDoc` AFTER UPDATE ON `suivicommandedoc` FOR EACH ROW BEGIN
    DECLARE idDoc VARCHAR(10);
    DECLARE nombre INTEGER;
    DECLARE date DATE;
    DECLARE numeroExemplaire INTEGER;
    /* trigger uniquement en cas de passage à l'étape 3 (livrée) */
    IF new.idSuivi = 3 THEN

        SELECT idLivreDvd, nbExemplaire, dateCommande INTO idDoc, nombre, date
        FROM suivicommandedoc scd
        JOIN commandedocument cd ON scd.idcommande = cd.id
        JOIN commande c ON cd.id = c.id
        WHERE scd.idcommande = old.idcommande;

        /* récupération du numéro d'exemplaire le plus élevé du document concerné */
        SELECT MAX(numero) INTO numeroExemplaire FROM exemplaire e WHERE e.id = idDoc;

        IF numeroExemplaire IS NULL THEN
            SET numeroExemplaire = 1;
        ELSE
            SET numeroExemplaire = numeroExemplaire + 1;
        END IF;

        /* ajout des exemplaires */
        REPEAT
            INSERT INTO exemplaire (id, numero, dateAchat, photo, idEtat) VALUES(idDoc, numeroExemplaire, date, "", "00001");
            SET numeroExemplaire = numeroExemplaire + 1;
            SET nombre = nombre - 1;
        UNTIL nombre = 0 END REPEAT;

    END IF;
END
$$
DELIMITER ;
```

Image 32 : Création d'un Trigger pour gérer les exemplaires.

Création d'un trigger pour contrôler la contrainte de partition de l'héritage sur commande.

Il a fallu créer un trigger qui vérifie si le numéro de commande est unique et qu'il ne correspond à aucune autre commande.

```
DROP TRIGGER IF EXISTS `insCommande`;
DELIMITER $$
CREATE TRIGGER `insCommande` BEFORE INSERT ON `commandedocument` FOR EACH ROW BEGIN
    DECLARE countId INTEGER;
    SELECT COUNT(*) INTO countId FROM abonnement WHERE id = new.id;
    IF countId > 0 THEN
        SIGNAL SQLSTATE "45000"
        SET MESSAGE_TEXT = "Un abonnement avec cet identifiant existe déjà.";
    END IF;
END
$$
DELIMITER ;
```

Image 33 : Création d'un Trigger vérifier que le numéro de commande est unique.

Création de la fenêtre d'alerte de fin d'abonnements.

Pour cette tâche, il était demandé de créer une procédure stockée dans la base de données pour récupérer les abonnements qui se terminent dans moins de 30 jours :

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `abonnementsFin30` () BEGIN
    SELECT a.dateFinAbonnement, a.idRevue, d.titre
        FROM abonnement a
        JOIN revue r ON a.idRevue = r.id
        JOIN document d ON r.id = d.id
        WHERE DATEDIFF(a.dateFinAbonnement, CURDATE()) < 30
        ORDER BY a.dateFinAbonnement ASC;
END$$
```

Image 34 : Procédure pour récupérer les abonnements qui expirent dans moins de 30 jours.

J'ai donc créé une fenêtre d'alerte qui se lancera dès le début de l'application, après la connexion.

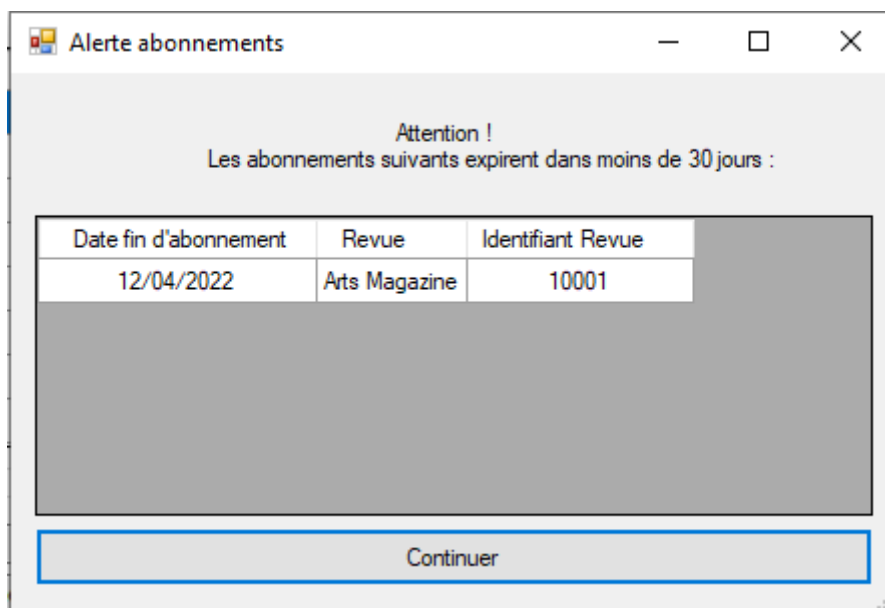


Image 35 : Fenêtre d'alerte.

Pour afficher les différentes commandes j'ai du créer une classe métier « FinAbonnement », une méthode GetFinAbonnement dans le contrôleur et le dao. Puis gérer l'affichage de la liste dans le DataGridView de la fenêtre d'alerte.

```
namespace Mediatek86.vue
{
    /// <summary>
    /// Class pour l'alerte
    /// </summary>
    4 références
    public partial class FrmAlerteFinAbonnements : Form
    {
        private readonly BindingSource bdgAlerteAbonnements = new BindingSource();

        private readonly List<FinAbonnement> lesFinAbonnement;

        /// <summary>
        /// Affichage des abonnements dans la fenêtre d'alerte
        /// </summary>
        /// <param name="controle"></param>
        1 référence
        internal FrmAlerteFinAbonnements(Controle controle)
        {
            InitializeComponent();
            lesFinAbonnement = controle.GetFinAbonnement();
            bdgAlerteAbonnements.DataSource = lesFinAbonnement;
            dgvAlerteFinAbonnements.DataSource = bdgAlerteAbonnements;
            dgvAlerteFinAbonnements.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
            dgvAlerteFinAbonnements.Columns["idRevue"].DisplayIndex = 2;
            dgvAlerteFinAbonnements.Columns[0].HeaderCell.Value = "Date fin d'abonnement";
            dgvAlerteFinAbonnements.Columns[1].HeaderCell.Value = "Identifiant Revue";
            dgvAlerteFinAbonnements.Columns[2].HeaderCell.Value = "Revue";
            btnAlerteFinAbonnements.Focus();
        }

        /// <summary>
        /// Selection d'un abonnement
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 référence
        private void dgvAlerteFinAbonnements_SelectionChanged(object sender, EventArgs e)
        {
            dgvAlerteFinAbonnements.ClearSelection();
        }

        /// <summary>
        /// Bouton pour quitter la fenêtre d'alerte
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 référence
        private void btnAlerteFinAbonnements_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Image 36 : La vue de la fenêtre de fin d'abonnement.

Bilan de la mission 2.

Cette mission était longue à mettre en place, mais pas vraiment difficile car il a fallu reprendre et modifier des fonctions déjà existantes. J'ai eu quelques difficultés au niveau de la récupération des données. Mais après avoir bien observé la base de données, j'ai compris certaines erreurs car je ne récupérais pas le bon 'id'.

La sécurisation des différents onglets s'est faite grâce à l'activation et la désactivation des boutons. J'ai également pris soin de vérifier le nombre de caractères possibles au niveau du numéro de commande.

```
Abonnement abonnement = new Abonnement(id, dateCommande, montant, finAbonnement, idRevue);
if (txbAboRevueNumeroDetails.Text.Length <= 5)
{
    if (controle.CreerAbonnement(abonnement))
    {
        AfficheAbonnementRevue();

        int addedRowIndex = -1;
        DataGridViewRow row = dgvAboRevueListe.Rows
            .Cast<DataGridViewRow>()
            .First(r => r.Cells["id"].Value.ToString().Equals(id));
        addedRowIndex = row.Index;
        dgvAboRevueListe.Rows[addedRowIndex].Selected = true;

        AccesDetailsAbonnementRevue(false);
        AccesGestionAbonnementRevue(true);
    }
    else
    {
        MessageBox.Show("Ce numéro d'abonnement existe déjà.", "Information");
        txbAboRevueNumeroDetails.Text = "";
        txbAboRevueNumeroDetails.Focus();
    }
}
else
{
    MessageBox.Show("Le numéro d'abonnement ne doit pas dépasser 5 caractères.", "Information");
    txbAboRevueNumeroDetails.Text = "";
    txbAboRevueNumeroDetails.Focus();
}
```

Image 37 : Sécurisation du numéro de commande.

J'ai pu mettre à jour mon suivi de projet.





<input type="checkbox"/>	<div> <div>Mission 2.5 : Sécurisation</div> <div>#4 by MarieCamillePetit was closed 2 days ago 2 tasks done</div> </div>	
<input type="checkbox"/>	<div> <div>Mission 2.4 : Gérer les commandes de revues</div> <div>#3 by MarieCamillePetit was closed 2 days ago 12 tasks done</div> </div>	
<input type="checkbox"/>	<div> <div>Mission 2.3 : Gérer les commandes de DVD</div> <div>#2 by MarieCamillePetit was closed 2 days ago 10 tasks done</div> </div>	
<input type="checkbox"/>	<div> <div>Mission 2.2 : Gérer les commandes de livres</div> <div>#1 by MarieCamillePetit was closed 3 days ago 10 tasks done</div> </div>	

Image 38 : Update suivi projet.

Mission 4 : Mettre en place des authentifications.

Les différentes tâches de la mission 4 sont les suivantes :

- Ajouter une table Utilisateur et une table service dans la base de données ;
- Ajouter une fenêtre d'authentification ;
- Empêcher l'accès à certaines fonctionnalités en fonction du service.

Création des tables 'Utilisateur' et 'Service'.

J'ai commencé cette nouvelle mission en créant les deux tables dans la base de données.

```
CREATE TABLE IF NOT EXISTS `utilisateur` (
  `NOM` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `ID` int(2) NOT NULL,
  `MDP` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Image 39 : Création table 'Utilisateur'.

```
CREATE TABLE IF NOT EXISTS `service` (
  `ID` int(2) NOT NULL,
  `LIBELLE` varchar(30) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
--
-- Déchargement des données de la table `service`
--
```

```
INSERT INTO `service` (`ID`, `LIBELLE`) VALUES
(4, 'culture'),
(3, 'prêt'),
(2, 'services administratifs'),
(1, 'admin');
```

Image 40 : Création table 'Service' et implémentations des différents services.

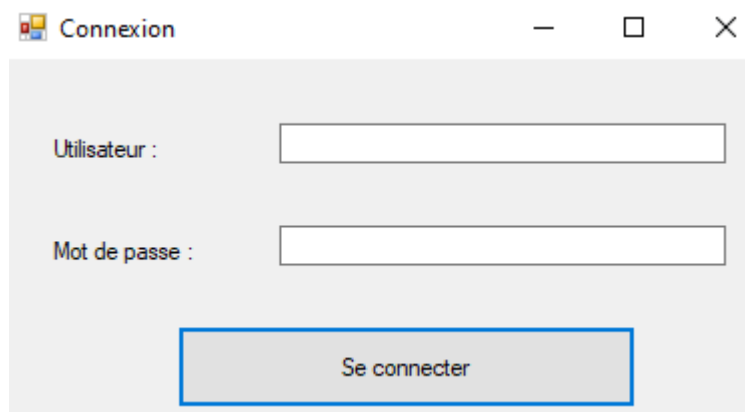
Étant donné que les accès aux différentes parties sont accessibles par un groupe de personnes, j'ai créé des comptes en fonction des différents services. Pour sécuriser ces comptes, ils seront stockés avec un hash de mots de passe (MD5).

```
INSERT INTO `utilisateur` (`NOM`, `ID`, `MDP`) VALUES
('admin', 1, '21232f297a57a5a743894a0e4a801fc3'),
('serviceadmin', 2, 'b186f841d646eb03b34761000b630c6f'),
('pret', 3, '5e18cf11d8566f62ce38f2a00fb8410e'),
('culture', 4, '3f7039a836c00d92ecf87fd7d338c4db');
```

Image 41 : Implémentations de la table Utilisateur avec hashage de mot de passe.

Création de la fenêtre d'authentification et accès aux différents services.

Pour commencer la partie authentification j'ai d'abord créé la fenêtre qui s'affichera lors du lancement de l'application. La fenêtre de connexion ressemble à :



The image shows a standard Windows-style dialog box titled "Connexion". It has a light gray background and a title bar with a minimize button, a maximize button (disabled), and a close button. Inside the dialog, there are two labels: "Utilisateur :" and "Mot de passe :". Each label is followed by a white rectangular text input field. Below these fields, centered, is a button with a blue border and a light gray fill, labeled "Se connecter".

Image 42 : Fenêtre de connexion.

```

/// <summary>
/// Frm Authentification
/// </summary>
/// <param name="controle"></param>
1 référence
public FrmAuthentification(Controle controle)
{
    InitializeComponent();
    this.controle = controle;
}

/// <summary>
/// Connexion et renvoie un message
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
2 références
private void btnConnexion_Click(object sender, EventArgs e)
{
    string utilisateur = txbUser.Text.Trim();
    string mdp = txbMDP.Text.Trim();
    Service userService = controle.Authentification(utilisateur, mdp);
}

```

Image 43 : Vue de l'authentification.

Il fallait modifier le constructeur du contrôleur pour prendre en compte la nouvelle fenêtre :

```

/// <summary>
/// Ouverture de la fenêtre
/// </summary>
2 références
public Controle()
{
    lesLivres = Dao.GetAllLivres();
    lesDvd = Dao.GetAllDvd();
    lesRevue = Dao.GetAllRevue();
    lesGenres = Dao.GetAllGenres();
    lesRayons = Dao.GetAllRayons();
    lesPublics = Dao.GetAllPublics();
    lesSuivis = Dao.GetAllSuivis();
    FrmAuthentification authentification = new FrmAuthentification(this);
    Application.Run(authentification);
    if (authentification.onSuccessAuth)
    {
        FrmMediatek frmMediatek = new FrmMediatek(this);
        Application.Run(frmMediatek);
    }
}

```

Image 44 : Vue d'authentification dans le contrôleur.

J'ai ensuite ajouté la nouvelle classe métier 'Service'.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Mediatek86.metier
{
    /// <summary>
    /// Class Service
    /// </summary>

    public class Service
    {
        private readonly int id;
        private readonly string libelle;
        /// <summary>
        /// Valorisation des propriété de la classe
        /// </summary>
        /// <param name="id"></param>
        /// <param name="libelle"></param>

        public Service(int id, string libelle)
        {
            this.id = id;
            this.libelle = libelle;
        }
        /// <summary>
        /// get Id
        /// </summary>

        public int Id => id;
        /// <summary>
        /// get Libelle
        /// </summary>
        3 références
        public string Libelle => libelle;
    }
}
```

Image 45 : La nouvelle classe métier 'Service'.

```

/// <summary>
/// Retourne le service de l'utilisateur
/// </summary>
/// <param name="utilisateur">Utilisateur</param>
/// <param name="mdp">Mot de passe</param>
/// <returns>Retourne le service de l'utilisateur</returns>
1 référence
public static Service Authentification(string utilisateur, string mdp)
{
    Service service = null;
    string req = "select s.ID, s.LIBELLE from utilisateur u join service s on u.ID = s.ID where u.NOM = @login and u.MDP = @pwd;";
    Dictionary<string, object> parameters = new Dictionary<string, object>
    {
        { "@login", utilisateur },
        { "@pwd", mdp }
    };
    BddMySQL curs = BddMySQL.GetInstance(connectionString);
    curs.ReqSelect(req, parameters);

    while (curs.Read())
    {
        service = new Service((int)curs.Field("id"), (string)curs.Field("libelle"));
    }
    curs.Close();
    return service;
}

```

Image 46 : Méthode « Authentification » dans le Dao.

Le mot de passe doit être hashé, j'ai donc du créer la méthode MD5 pour pouvoir hasher le mot de passe dans le contrôleur.

```

/// <summary>
/// hash de mot de passe
/// </summary>
/// <param name="mdp">Mot de passe</param>
/// <returns>Mot de passe haché</returns>
1 référence
public string hashMD5(string mdp)
{
    using (MD5 md5 = MD5.Create())
    {
        byte[] inputBytes = Encoding.ASCII.GetBytes(mdp);
        byte[] hashBytes = md5.ComputeHash(inputBytes);

        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < hashBytes.Length; i++)
        {
            sb.Append(hashBytes[i].ToString("X2"));
        }
        return sb.ToString();
    }
}

```

Image 47 : Méthode de hashage de mot de passe.

Pour pouvoir gérer les différents accès à l'application j'ai complété la méthode événementielle du clic sur le bouton Connexion.

```

/// <summary>
/// Connexion et renvoie un message
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
2 références
private void btnConnexion_Click(object sender, EventArgs e)
{
    string utilisateur = txtUser.Text.Trim();
    string mdp = txtMDP.Text.Trim();
    Service userService = controle.Authentification(utilisateur, mdp);

    if (userService != null)
    {
        if (userService.Libelle == "culture")
        {
            MessageBox.Show("Accès réservé aux services administratifs et au service de prêt.", "Information");
            VideTbxConnexion();
        }
        else
        {
            onSuccessAuth = true;
            Close();
        }
    }
    else
    {
        MessageBox.Show("Vérifiez vos identifiants de connexion, ils sont incorrects..", "Information");
        VideTbxConnexion();
    }
}

```

Image 48 : Événement lors du clic sur le bouton 'Connexion'.

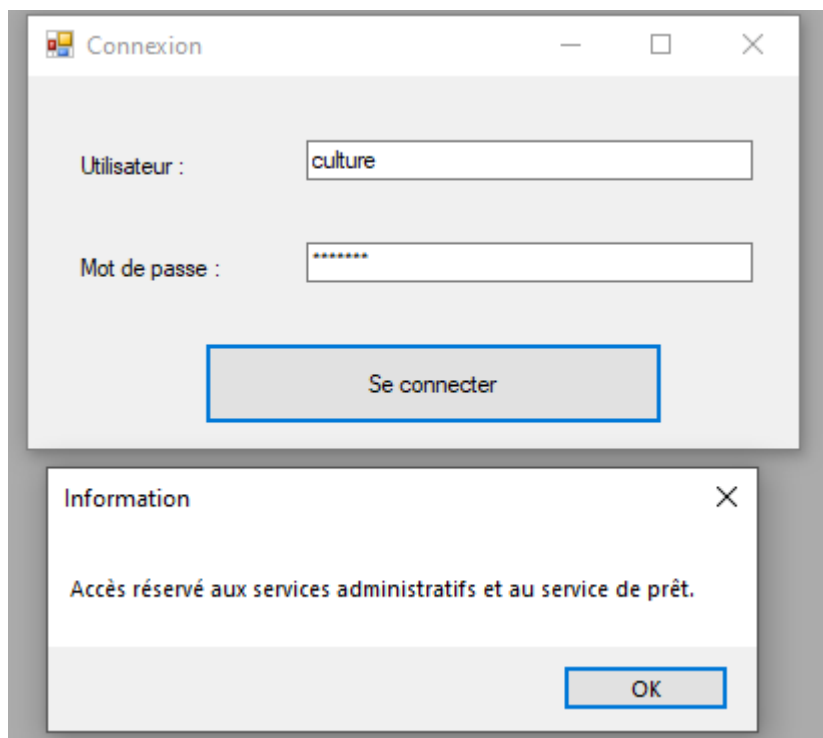


Image 49 : Accès refusé pour le service 'Culture'.

Pour gérer les accès aux différents onglets en fonction des services j'ai modifié la vue 'FrmMediatek' en y ajoutant la condition d'accès.

```

/// <summary>
/// Permet de rendre invisible des parties de l'application en fonction du service
/// </summary>
/// <param name="controle"></param>
2 références
public FrmMediatek(Controle controle)
{
    InitializeComponent();
    this.controle = controle;
    if (controle.userService.Libelle == "prêt")
    {
        tabOngletsApplication.TabPages.Remove(tabCommandeLivre);
        tabOngletsApplication.TabPages.Remove(tabCommandeDVD);
        tabOngletsApplication.TabPages.Remove(tabAboRevue);
        grpReceptionExemplaire.Visible = false;
    }
}

```

Image 50 : Conditions d'accès pour le service « Prêt ».

```

/// <summary>
/// Dès l'ouverture de l'application la vue d'alerte de fin d'abonnements s'ouvre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void FrmMediatek_Shown(object sender, EventArgs e)
{
    if (controle.userService.Libelle != "prêt")
    {
        FrmAlerteFinAbonnements alerteFinAbonnements = new FrmAlerteFinAbonnements(controle)
        {
            StartPosition = FormStartPosition.CenterParent
        };
        alerteFinAbonnements.ShowDialog();
    }
}

```

Image 51 : Ne pas montrer la fenêtre de fin d'abonnement au service « Prêt ».

Bilan de la mission 4.

La mission est sans difficulté, j'ai du faire quelques recherches sur internet pour comprendre comment hasher un mot de passe et pour le mettre dans la base de données.

J'ai testé toutes les fonctionnalités de l'application sans avoir de problème.

Mission 5 : Qualité, tests et documentation technique.

Contrôle de qualité avec SonarLint.

Pour cette tâche, j'ai lancé une analyse de code avec SonarLint dans Visual Studio. Il m'a retourné aucune erreur, aucun avertissement et beaucoup de messages d'informations.

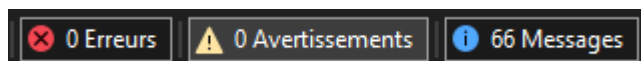


Image 52 : Liste des erreurs.

Tout au long de cet atelier, j'ai réparé les différents messages d'erreurs et d'avertissements. Les messages d'information restants étaient principalement de violation de règles de nommage.

IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : userService	Mediatek86	Controle.cs	28	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : hashMD5	Mediatek86	Controle.cs	256	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : tabRevue_Enter	Mediatek86	FrmMediatek.cs	138	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : cbxRevueTitreRecherche_TextChanged	Mediatek86	FrmMediatek.cs	206	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : cbxRevuePublics_SelectedIndexChanged	Mediatek86	FrmMediatek.cs	296	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : cbxRevueRayons_SelectedIndexChanged	Mediatek86	FrmMediatek.cs	315	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : dgvRevueListe_SelectionChanged	Mediatek86	FrmMediatek.cs	335	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : btnRevueAnnulPublics_Click	Mediatek86	FrmMediatek.cs	360	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : btnRevueAnnulRayons_Click	Mediatek86	FrmMediatek.cs	370	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : btnRevueAnnulGenres_Click	Mediatek86	FrmMediatek.cs	380	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : dgvRevueListe_ColumnHeaderMouseClick	Mediatek86	FrmMediatek.cs	412	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : tabDvd_Enter	Mediatek86	FrmMediatek.cs	779	Actif
IDE1006	Violation des règles de nommage : Les mots suivants doivent commencer par des lettres majuscules : btnDvdNumRecherche_Click	Mediatek86	FrmMediatek.cs	811	Actif

Image 53 : Liste de message d'information lié aux noms des fonctions.

Je n'ai pas vraiment prêté attention à ces messages, car je ne souhaite pas modifier les fonctions automatiques créées par Visual Studio.

J'ai également eu des messages concernant l'utilisation des initialiseurs d'objets :

IDE0017	L'initialisation de l'objet peut être simplifiée	Mediatek86	FrmMediatek.cs	1243	Actif
---------	--	------------	----------------	------	-------

Image 54 : Liste de message d'information lié aux initialiseurs d'objets.

```
AlerteFinAbonnements alerteFinAbonnements = new AlerteFinAbonnements
    (controle);
alerteFinAbonnements.StartPosition = FormStartPosition.CenterParent;
alerteFinAbonnements.ShowDialog();
```

Image 55 : Fonction à modifier.

Pour résoudre ce message, j'ai utilisé les conseils de SonarLint :

```
FrmAlerteFinAbonnements alerteFinAbonnements = new FrmAlerteFinAbonnements(controle)
{
    StartPosition = FormStartPosition.CenterParent
};
alerteFinAbonnements.ShowDialog();
```

Image 56 : Modifications.

Création d'un test fonctionnel avec SpecFlow.

Pour la création d'un test fonctionnel, j'ai utilisé SpecFlow, et créer un nouveau projet dans la solution.

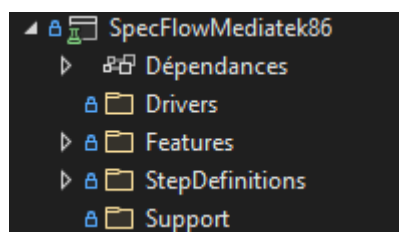


Image 57 : Création du test fonctionnel.

J'ai créé mon premier test dans les fichiers créés. Ce test est fait pour tester la fonctionnalité de recherche d'un livre en tapant le numéro d'un document.

```
Feature: RechercheLivreNumero
    Test sur la recherche d'un document par son numéro

    @rechercheLivreNumero
    Scenario: Chercher un livre par son numero
        Given Je saisis la valeur 00025
        When Je clic sur le bouton Rechercher
        Then Les informations détaillées doivent afficher le titre L'archipel du danger
```

Image 58 : Création du scénario du test fonctionnel.

```

using Mediatek86.vue;
using System.Windows.Forms;
using TechTalk.SpecFlow;
using Mediatek86.controleur;
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace SpecFlowMediatek86.Steps
{
    [Binding]
    0 références
    public class RechercheLivreNumeroSteps
    {
        private readonly FrmMediatek frmMediatek = new FrmMediatek(new Controle());

        [Given(@"Je saisis la valeur (.*)")]
        0 références
        public void GivenJeSaisisLaValeur(string valeur)
        {
            TextBox TxtValeur = (TextBox)frmMediatek.Controls["tabOngletsApplication"].Controls["tabLivres"].Controls["grpLivresRecherche"].Controls["txbLivresNumRecherche"];
            frmMediatek.Visible = true;
            TxtValeur.Text = valeur;
        }

        [When(@"Je clique sur le bouton Rechercher")]
        0 références
        public void WhenJeCliqueSurLeBoutonRechercher()
        {
            Button BtnRechercher = (Button)frmMediatek.Controls["tabOngletsApplication"].Controls["tabLivres"].Controls["grpLivresRecherche"].Controls["btnLivresNumRecherche"];
            frmMediatek.Visible = true;
            BtnRechercher.PerformClick();
        }

        [Then(@"Les informations détaillées doivent afficher le titre (.*)")]
        0 références
        public void ThenLesInformationsDetailleesDoiventAfficherLeTitre(string titreAttendu)
        {
            TextBox TxtTitre = (TextBox)frmMediatek.Controls["tabOngletsApplication"].Controls["tabLivres"].Controls["grpLivresInfos"].Controls["txbLivresTitre"];
            string titreObtenu = TxtTitre.Text;
            Assert.AreEqual(titreAttendu, titreObtenu);
        }
    }
}

```

Image 59 : Création des steps du test fonctionnel.

Le test fonctionnel a réussi.

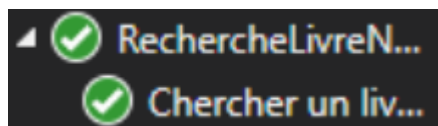


Image 60 : Réussite du test.

Journalisation.

Pour faire la journalisation et ainsi pouvoir créer des logs, j'ai utilisé l'extension Serilog Analyser.

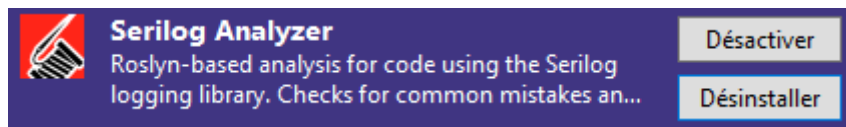


Image 61 : Extension Serilog Analyser.

J'ai ensuite ajouté les fonctionnalités qu'offre Serilog dans les méthodes qui devaient avoir des logs.

```
Log.Logger = new LoggerConfiguration()
    .MinimumLevel.Information()
    .WriteTo.File(new JsonFormatter(), "logs/log.txt",
        rollingInterval: RollingInterval.Day)
    .CreateLogger();
```

Image 62: Initialisation du logger.

```
/// <summary>
/// Constructeur privé pour créer la connexion à la BDD et l'ouvrir
/// </summary>
/// <param name="stringConnect">chaîne de connexion</param>
1 référence
private BddMySQL(string stringConnect)
{
    try
    {
        connection = new MySqlConnection(stringConnect);
        connection.Open();
    }
    catch (MySqlException e)
    {
        Log.Error("BddMySQL.BddMySQL_catch **** stringConnect = " + stringConnect + " **** MySqlException = " + e.Message);
        ErreurGraveBddNonAccessible(e);
    }
}
```

Image 63 : Création de log en cas de problème provenant de la BDD.

Documentation technique.

Pour la génération de la documentation technique j'ai utilisé SandCaste. Mais Avant de procéder à cette génération j'ai vérifié que les commentaires ont été insérés partout et qu'ils soient clairs et pertinents. J'ai généré une première documentation en XML, cette documentation n'est pas très lisible, c'est pourquoi j'ai utilisé SandCastle pour obtenir une documentation accessibles en fichier HTML Compilé.

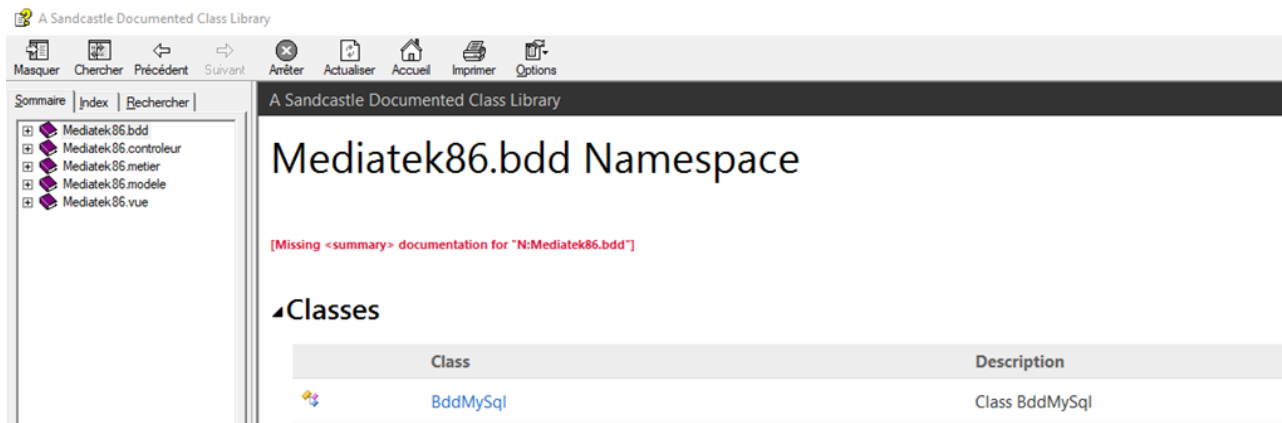


Image 64 : Documentation technique.

Bilan.

La mission s'est globalement bien passée, cependant j'ai eu des difficultés de compréhension avec SpecFlow. Je trouve que la prise en main est compliquée et j'ai eu de nombreux problèmes pendant l'utilisation de ces tests fonctionnels.

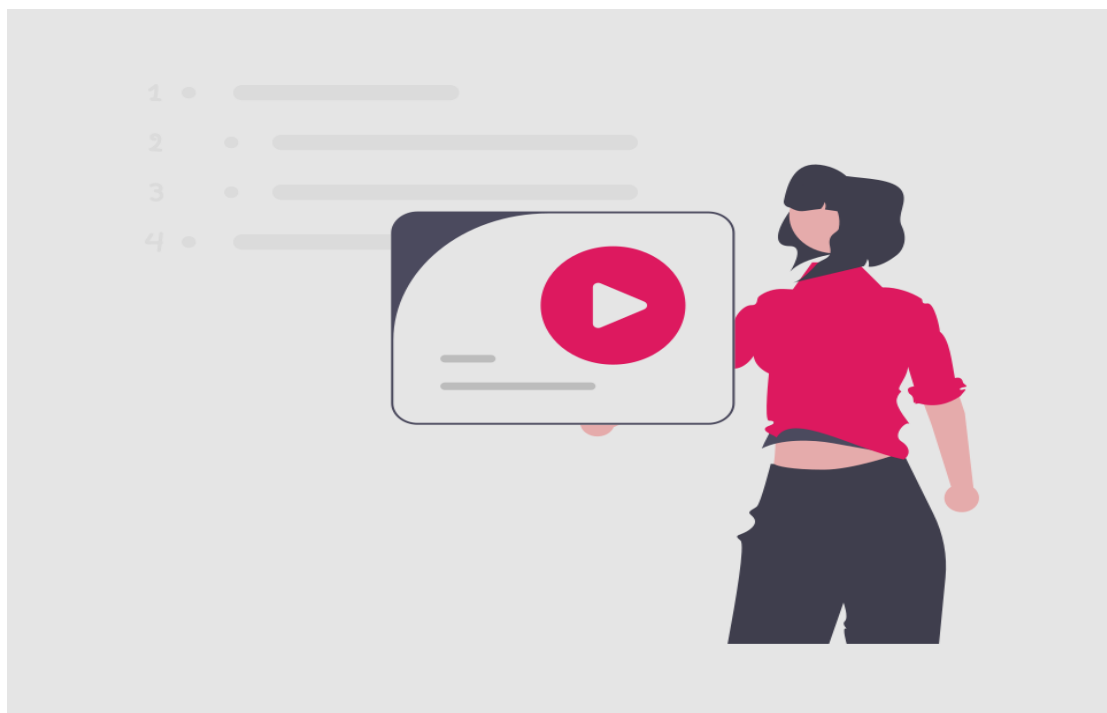
J'ai également eu un problème lors de la documentation technique, car celle-ci ne se crée pas totalement dans SandCastle. C'est pourquoi j'ai du prendre le format Html compilé.

Déploiement de l'application.

Pour pouvoir utiliser l'application j'ai du mettre en ligne la base de données sur Azure. J'ai choisi Azure car pendant nos cours, nous avons utilisé cette plateforme à de nombreuses reprises.

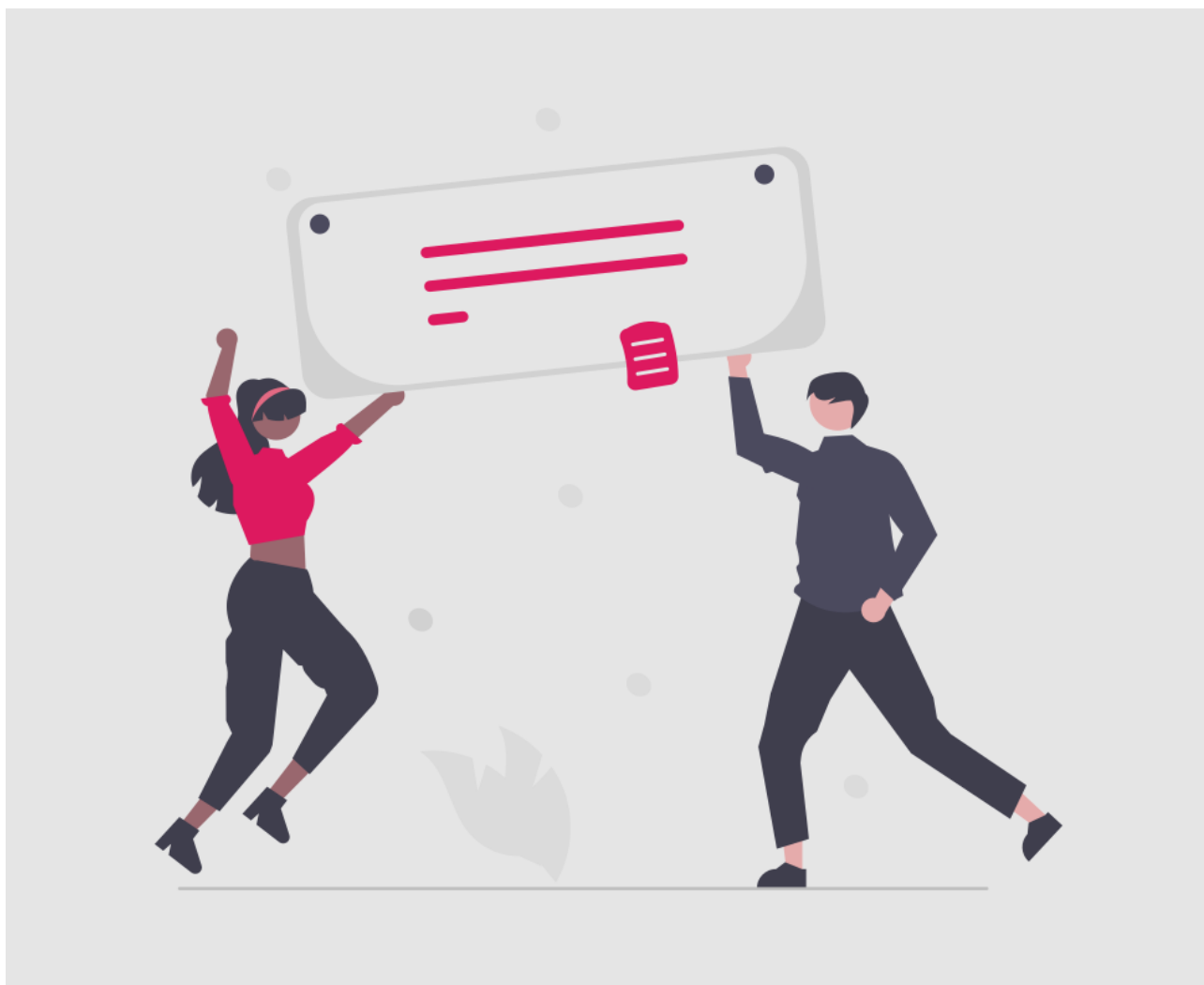
Mission 6 : Créer une vidéo de démonstration.

Pour la mission 6, il faut créer une vidéo de maximum 10 minutes, faisant une démonstration de toutes les fonctionnalités de l'application. Vous pourrez retrouver cette vidéo sur mon portfolio.



Bilan Final.

Cet atelier était le plus important et complexe. Projet que j'ai pu réaliser pendant les deux années de BTS SIO. Il reprend les compétences acquises dans les cours du Bloc 2 'Conception et développement d'applications', du bloc 1 'Support et mise à disposition de services informatiques' et du bloc 3 'Cyber sécurité des services informatiques'. Je suis assez satisfaite du résultat, cependant je trouve avoir perdu beaucoup de temps sur certaines solutions. Toutes les fonctionnalités demandées dans les missions obligatoires du dossier documentaire ont été réalisées et fonctionnent. L'hébergement de la base de données en ligne, était compliqué car j'ai cherché plusieurs possibilités pour l'héberger et pour comprendre comment faire. En reprenant certains cours et grâce à des recherches sur internet, j'ai réussi à l'héberger sur un serveur Flexible d'Azure pour MySql.



Compétences mobilisées.

Les compétences officielles couvertes	
Bloc 1	Répondre aux incidents et aux demandes d'assistance et d'évolution.
	Travailler en mode projet.
	Mettre à disposition des utilisateurs un service informatique :
Bloc 2	Concevoir et développer une solution applicative.
	Assurer la maintenance corrective ou évolutive d'une solution applicative.
	Gérer les données.
Bloc 3	Sécuriser les équipements et les usages des utilisateurs.
	Assurer la cybersécurité d'une solution applicative et de son développement.