

ATELIER DE PROFESSIONNALISATION.

Application de bureau (C#) exploitant une base de données relationnelle (MySQL) à travers une API REST.



**MediaTek86**

Des formations pour tous  
sur des outils numériques

Brevet de technicien supérieur - Services informatiques aux organisations 2023-2024

**CNED**

# Table des matières

Préambule.....	5
Mission 0 : préparer l'environnement de travail.....	5
Mission 1 : gérer les documents.....	6
Ajouter d'un Livre.....	8
VIEW.....	9
CONTROLLER.....	14
DAL.....	14
API.....	14
Modifier un Livre.....	16
VIEW.....	16
CONTROLLER.....	18
DAL.....	19
API.....	19
Supprimer un livre.....	20
VIEW.....	21
CONTROLLER.....	22
DAL.....	22
API.....	22
Ajouter un DVD.....	23
VIEW.....	24
CONTROLLE.....	25
DAL.....	26
API.....	26
Modifier un DVD.....	27
VIEW.....	27
CONTROLLE.....	29
DAL.....	29
API.....	29
Supprimer un dvd.....	30
VIEW.....	31
CONTROLLE.....	31
DAL.....	31
API.....	32
Ajouter une Revue.....	33
VIEW.....	33
CONTROLLE.....	35
DAL.....	35
API.....	35
Modifier une revue.....	36
VIEW.....	36
CONTROLLE.....	38
DAL.....	38
API.....	39
Supprimer une revue.....	40
VIEW.....	40
CONTROLLER.....	41
DAL.....	41
API.....	41
Mission 2 : gérer les commandes.....	42
Tâche 1 : gérer les commandes de livres ou de DVD.....	42
Création de la table suivi.....	43

Classes nécessaire aux commandes.....	45
Charte graphique.....	46
Commande de Livres.....	47
Recherche d'un livre.....	47
Ajouter une commande de livre.....	48
VIEW.....	48
CONTROLLER.....	50
DAL.....	51
API.....	51
DataGridView des commande de livre.....	53
Modifier une commande de livre.....	54
VIEW.....	54
CONTROLLER.....	55
DAL.....	56
API.....	56
Supprimer une commande de livre.....	58
VIEW.....	58
CONTROLLE.....	59
DAL.....	59
Ajouter une commande de dvd.....	59
VIEW.....	59
CONTROLLER.....	61
DAL.....	61
API.....	61
Modifier une commande DVD.....	62
VIEW.....	62
CONTROLLER.....	62
DAL.....	63
API.....	63
Supprimer une commande de DVD.....	64
VIEW.....	64
CONTROLLER.....	64
DAL.....	64
Tâche 2 : gérer les commandes de revues.....	64
Classes nécessaire aux commandes.....	65
Commande d'une revue.....	65
Recherche d'une revue.....	65
CONTROLLER.....	66
DAL.....	67
API.....	67
Supprimer une commande de revue.....	68
CONTROLLE.....	69
DAL.....	70
Test unitaire.....	70
Abonnement se terminant.....	72
Mission 3 : gérer le suivi de l'état des exemplaires.....	73
Exemplaires Livres.....	73
VIEW.....	73
CONTROLLE.....	77
DAL.....	77
API.....	77
Suppression.....	78

Exemplaires DVD.....	78
Exemplaires revues.....	80
Mission 4 : mettre en place des authentifications.....	82
Base de données.....	82
Model.....	84
VIEW.....	84
CONTROLER.....	86
DAL.....	86
API.....	87
Mission 5 : assurer la sécurité, la qualité et intégrer des logs.....	89
Tâche 1 : corriger des problèmes de sécurité.....	89
Problème 1.....	90
Problème 2.....	94
Tâche 2 : contrôler la qualité.....	96
Configuration SonarQube.....	96
CONFIGURATION JENKINS.....	100
Tâche 3 : intégrer des logs.....	103
Mission 6 : tester et documenter.....	105
Tâche 1 : gérer les tests.....	105
Tests unitaires.....	105
Tests fonctionnels.....	107
Tests sur Postman.....	108
Tâche 2 : créer les documentations techniques.....	110
Tâche 3 : créer la documentation utilisateur en vidéo.....	112
Mission 7 : déployer et gérer les sauvegardes de données.....	112
Tâche 1 : déployer le projet.....	112
En ligne.....	113
Tâche 2 : gérer les sauvegardes des données.....	118
Bilan.....	119

## Préambule.

Cette réalisation se fait dans le contexte suivant.

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.

Parmi les marchés gagnés récemment par ITS 86, on trouve celui de la gestion du parc informatique du réseau des médiathèques de la Vienne, MediaTek86, ainsi que l'informatisation de plusieurs activités internes des médiathèques ou en lien avec le public.

Je travail en tant que technicien junior développeur pour l'ESN InfoTech Services 86. Il m'est confié la réalisation d'une application de bureau de gestions de documents.

Cette application est déjà commencée, elle est réalisée en C#. Il m'est demandé de poursuivre son développement en gardant la structure actuelle et en suivant un cheminement se composant de sept missions qui constituent les parties majeures du process de développement de l'application.

## Mission 0 : préparer l'environnement de travail.

Pour cette réalisation, j'utilise l'IDE Visual Studio 2019 pour coder l'application et l'IDE Apache NetBeans pour coder l'API. Ainsi que tout le nécessaire pour travailler en local dans un premier temps Wampserver, Postman etc ... J'utilise la version de PHP 7,4,33 car l'API n'est pas compatible avec les dernières versions de PHP.

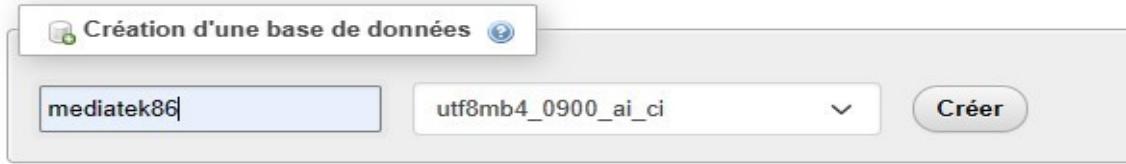
Je récupère l'API, le script de la base de données ainsi que le début de l'application sur les dépôts GitHub du CNED.

The screenshot shows a GitHub repository page for 'rest\_mEDIATEkdocuments'. The repository is public and has 1 branch and 0 tags. The master branch is selected. There are 5 commits listed:

Commit	Description	Date
Ajout du readme	Api d'accès à la BDD Mediatek86 + script de la BDD	6 months ago
nbproject	Etendue des possibilités des requêtes select	2 years ago
.htaccess	Etendue des possibilités des requêtes select	6 months ago
AccessBDD.php	Correction de l'erreur 400 quand une requête select retour...	6 months ago
ConnexionPDO.php		10 months ago

J'installe l'API dans le dossier www dédié à WAMP. Sur PhpMyAdmin, je crée la base de donnée mediatek86 et j'importe le script de la base de donnée.

# Bases de données



Je crée un dépôt sur mon GitHub qui va contenir mon application et toutes ses évolutions. Je l'initialise avec l'application récupérée depuis le GitHub du CNED.

```
PowerShell développeur
+ PowerShell développeur | ⌂ ⌂ | ⚙

PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git remote add origin https://github.com/hedi-k/Atelier3.git
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git branch -M main
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git push -u origin main
Enumerating objects: 46, done.
Counting objects: 100% (46/46), done.
Delta compression using up to 8 threads
Compressing objects: 100% (44/44), done.
Writing objects: 100% (46/46), 41.25 KiB | 2.17 MiB/s, done.
Total 46 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/hedi-k/Atelier3.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments>
```

Dans mon dépôt, je crée un projet Kanban et je le remplis de toutes les tâches prévues.

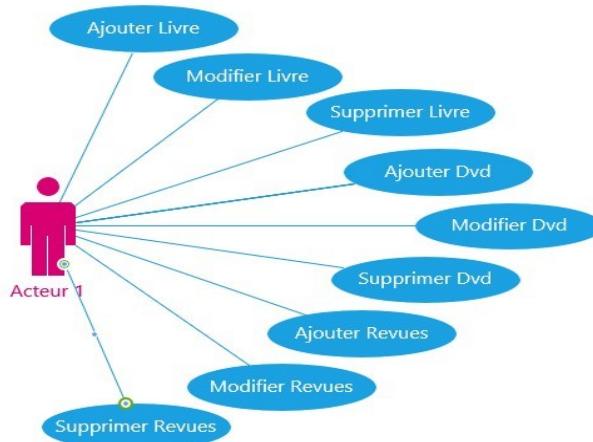
## Mission 1 : gérer les documents

### Tâches à réaliser

Dans les onglets actuels (Livres, Dvd, Revues), ajouter les fonctionnalités (boutons) qui permettent d'ajouter, de modifier ou de supprimer un document. Un document ne peut être supprimé que s'il n'a pas d'exemplaire rattaché, ni de commandes. La modification d'un document ne peut pas porter sur son id. Toutes les sécurités seront mises en place pour éviter des erreurs de manipulation.

Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Document, idem pour LivresDvd.

Après avoir mis à jour mon kaban sur GitHub, je fais le diagramme de cas d'utilisation de la mission.



Pour ces actions, je fais le choix d'ajouter trois boutons Ajouter, Modifier et Supprimer sur la vue FrmMediatek.

The screenshot shows a Windows application window titled "Gestion des documents de la médiathèque". The tab "Livres" is selected. The interface includes search fields for title/part of title, document number, genre, public, and rayon. Below is a grid of document records:

ID	Titre	Auteur	Collection	Genre	Public	Rayon
00017	Catastrophes au Brésil	Philippe Masson		Policier	Ados	Jeunesse romans
00007	Dans les coulisses du musée	Kate Atkinson		Roman	Tous publics	Littérature étrangère
00003	Et je danse aussi	Anne-Laure Bondoux		Comédie	Tous publics	Littérature française
00019	Guide Vert - Iles Canaries		Guide Vert	Voyages	Tous publics	Voyages
00020	Guide Vert - Irlande		Guide Vert	Voyages	Tous publics	Voyages
00008	Histoire du juif errant	Jean d'Ormesson		Roman	Adultes	Littérature française
00025	L'archipel du danger	Ayrolles - Masbou	De cape et de crocs	Bande dessinée	Adultes	BD Adultes
00004	L'armée furieuse	Fred Vargas	Commissaire Adamsberg	Policier	Adultes	Policiers français étrangers

Below the grid, detailed information for document ID 00017 is shown:

Numéro de document :	00017	Code ISBN :	
Titre :	Catastrophes au Brésil		
Auteur(e) :	Philippe Masson		
Collection :			
Genre :	Policier		
Public :	Ados		
Rayon :	Jeunesse romans		
Chemin de l'image :			

Buttons at the bottom: Ajouter, supprimer, and Modifier.

Les deux premiers boutons ont pour conséquence d'ouvrir une deuxième fenêtre FrmAjout. Qui propose tout le nécessaire à l'ajout ou la modification d'un document.

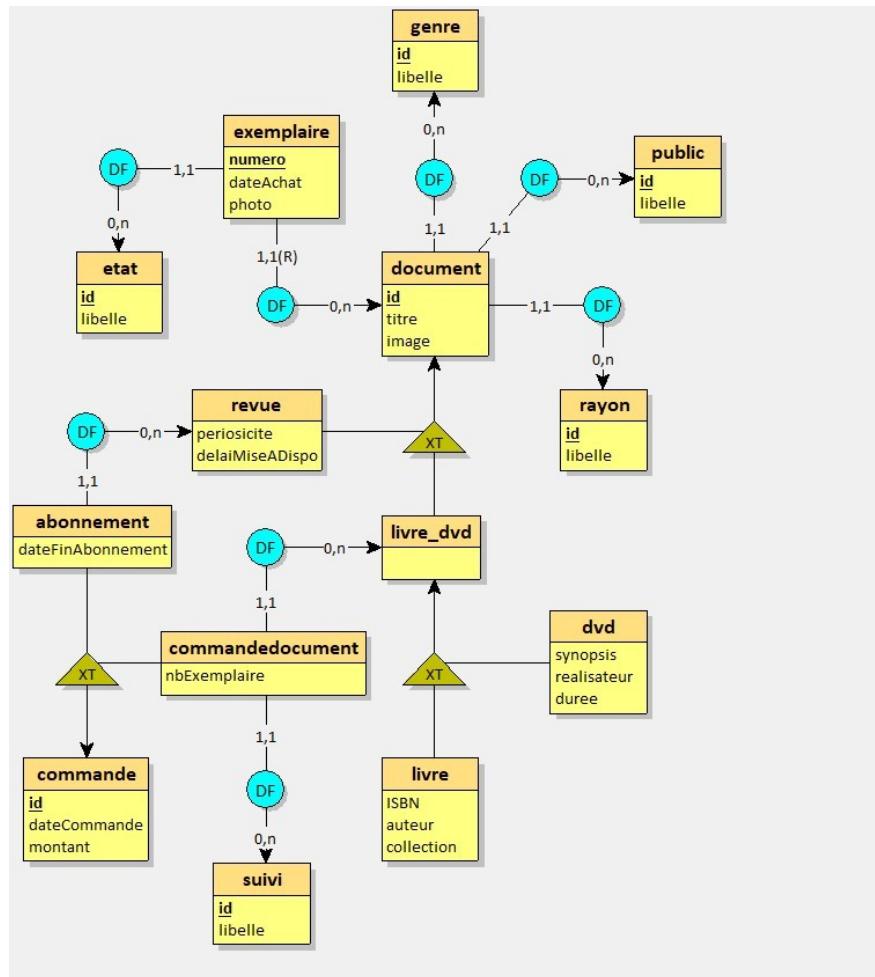
The screenshot shows a Windows application window titled "Form1". It contains fields for genre, public, rayon, document number, title, and image path. Buttons for Ajouter (Add), Modifier (Modify), and Annuler (Cancel) are present. A note at the bottom states: "\* Champs obligatoire pour ajouter un livre".

Genre*	Public*	Rayon*		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
Numéro de document *:	<input type="text"/>	<input type="text"/>	Ajouter	
Titre *:	<input type="text"/>			
Genre :	<input type="text"/>			
Public :	<input type="text"/>			
Rayon :	<input type="text"/>			
Chemin de l'image :	<input type="text"/>			Modifier
* Champs obligatoire pour ajouter un livre				Annuler

## Ajouter d'un Livre

Pour comprendre l'Addition d'un livre, il est nécessaire d'expliquer ce que représente l'objet Livre.  
 Pour ce faire nous disposons du modèle conceptuel de données (MCD) de l'application et de sa base de données associée.

MCD



On comprend depuis ce schéma que la table livre et dvd ont une partition sur héritage sur la table livre\_dvd. Donc tout les livre\_dvd sont soit des livres soit des Dvds.

De plus la table revue et livre\_dvd ont une partition sur héritage sur la table document. Donc tout les documents sont soit des revues soit des livre\_dvd.

De la on peut en déduire qu'ajouter, modifier ou supprimer un livre équivaut à modifier la table livre, la table livre\_dvd et la table document.

En étudiant dans le programme, le modèle [Livre.cs](#) on constate que les données concordent bien avec le MCD.

```

43 références | hedi-k, il y a 18 jours | 1 auteur, 1 modification
public class Livre : LivreDvd
{
    4 références | hedi-k, il y a 18 jours | 1 auteur, 1 modification
    public string Isbn { get; }
    5 références | hedi-k, il y a 18 jours | 1 auteur, 1 modification
    public string Auteur { get; }
    5 références | hedi-k, il y a 18 jours | 1 auteur, 1 modification
    public string Collection { get; }

    1 référence | hedi-k, il y a 18 jours | 1 auteur, 1 modification
    public Livre(string id, string titre, string image, string isbn, string auteur, string collection,
        string idGenre, string genre, string idPublic, string lePublic, string idRayon, string rayon)
        : base(id, titre, image, idGenre, genre, idPublic, lePublic, idRayon, rayon)
    {
        this.Isbn = isbn;
        this.Auteur = auteur;
        this.Collection = collection;
    }
}

```

La classe Livre hérite de la classe LivreDvd.

```

public abstract class LivreDvd : Document
{
    2 références | hedi-k, il y a 18 jours | 1 auteur, 1 modification
    protected LivreDvd(string id, string titre, string image, string idGenre, string genre,
        string idPublic, string lePublic, string idRayon, string rayon)
        : base(id, titre, image, idGenre, genre, idPublic, lePublic, idRayon, rayon)
    {
    }
}

```

Et la classe LivreDvd hérite de la classe Document.

Donc ajouter un livre dans la BDD consiste en la valorisation d'un objet de la classe Livre, transférer cet objet dans un format compréhensible à l'API qui va la transmettre à son tour à la BDD.

## VIEW

Je code l'action du bouton Ajouter de [FrmMediatek.cs](#). Son rôle est de ouvrir [FrmAjout.cs](#) avec divers paramètres qui vont affecter la seconde vue.

```

//Action du bouton Ajouter (livre)
//Le boolean est pour l'affichage des TextBoxs genre public rayon et Bouton modifier.
1 référence | hedi-k, il y a 7 jours | 1 auteur, 6 modifications
private void btnAjout_Click(object sender, EventArgs e)
{
    //Chaîne de caractère qui va influencer les structures conditionnelles de FrmAjout
    string ongletLivre = "livre";
    //Converti la liste de livres en liste d'objets
    List<Object> listeObjLivre = lesLivres.ConvertAll(Livre => (Object)Livre);
    //Lance frmAjout avec les paramètres nécessaires à l'ajout d'un livre
    FrmAjout frmAjout = new FrmAjout(bdgGenres, bdgPublics, bdgRayons, false, ongletLivre, listeObjLivre);
    frmAjout.Text = "Ajout d'un LIVRE ";
    frmAjout.Show();
    this.Hide();
}

```

La seconde vue va donc démarrer avec cette méthode.

```

6 références | hedi-k, il y a 11 jours | 1 auteur, 3 modifications
public FrmAjout(BindingSource bdgGenres2, BindingSource bdgPublics, BindingSource bdgRayons, bool affichage, string onglet,
    List<Object> leslistes = null, Object aModifier = null)
{
    InitializeComponent();
    //Remplit les ComboBox qui contiennent les Genres, Publics et Rayons.
    RemplirCbx(bdgGenres2, cbxGenres);
    RemplirCbx(bdgPublics, cbxPublics);
    RemplirCbx(bdgRayons, cbxRayons);
    //Gestion des affichages des TextBox et label
    Affichage(onglet, affichage);
    //Convertit la liste d'objets reçue en liste de livres, de DVDs ou de revues.
    ChargeListe(onglet, leslistes);
    //Méthode pour la modification d'un livre ou un dvd
    ChargeObjet(onglet, aModifier);
    //Affecte le comportement pour l'appel du contrôleur
    quelEnvoi = onglet;
}

```

D'ici toutes les fonctions vont s'adapter pour le traitement d'un livre. Procéder comme cela va me permettre d'utiliser les mêmes fonctions que ce soit pour l'ajout et la modification d'un livre ou d'un DVD.

RemplirCbx va remplir les ComboBox avec les BindingSource quelle reçoit en paramètre.

```

//Remplit la comboBox reçue en paramètre en fonction de la bdg reçue en paramètre
3 références | hedi-k, il y a 7 jours | 1 auteur, 3 modifications
private void RemplirCbx(BindingSource bdg, ComboBox cbx)
{
    cbx.DataSource = bdg;
    if (cbx.Items.Count > 0)
    {
        cbx.SelectedIndex = -1;
    }
}

```

Affichage() va modifier les noms des labels, l'accès des boutons et textBox en fonction des paramètres reçus, pour être cohérent avec l'opération voulu.

Dans le cas d'un ajout, on veut que le bouton modifier soit inaccessible et que tout les labels concernent un livre. Voilà pourquoi j'ai envoyé livre en chaîne de caractères dans la [FrmMediatek.cs](#) et un booléen à false.

```

//Gestion des affichages des TextBox et label
1 référence | hedi-k, il y a 7 jours | 1 auteur, 2 modifications
private void Affichage(string onglet, bool affichage)
{
    //Adapte le comportement des boutons
    btnAjouter.Enabled = !affichage;
    txbNumero.Enabled = !affichage;
    btnModifier.Enabled = affichage;
    txbGenre.Enabled = false;
    txbPublic.Enabled = false;
    txbRayon.Enabled = false;
    //Adapte les label
    switch (onglet)
    {
        case "livre":
            lblAuteurRealisateurPer.Text = "Auteur * :";
            lblCollectionSynopsisDel.Text = "Collection * :";
            lblIsbnDuree.Text = "ISBN * :";
            txbIsbnDuree.Enabled = true;
            break;
        case "dvd":
            lblAuteurRealisateurPer.Text = "Realisateur * :";
            lblCollectionSynopsisDel.Text = "Synopsis * :";
            lblIsbnDuree.Text = "Durée * :";
            txbIsbnDuree.Enabled = true;
            break;
        case "revue":
            lblAuteurRealisateurPer.Text = "Périodicité * :";
            lblCollectionSynopsisDel.Text = "Délai mise à dispo * :";
            lblIsbnDuree.Text = "";
            txbIsbnDuree.Enabled = false; // ne pas afficher pour les revues
            break;
    }
}

```

La fonction [ChargeListe\(\)](#) va me permettre de convertir une liste d'objets en liste de livres dans ce cas. Ce procédé me permet de ne pas avoir la méthode [FrmAjout\(\)](#) trop chargées de conditions en fonction du traitement d'un livre d'un dvd ou d'une revue.

```

//Convertit la liste d'objets reçue en liste de livres, de DVD ou de revues.
1 référence | hedi-k, il y a 7 jours | 1 auteur, 4 modifications
private void ChargeListe(string onglet, List<Object> uneListe)
{
    switch (onglet)
    {
        case "livre":
            //Convertit la liste d'objets en livres
            listeLivre = uneListe.ConvertAll(Object => (Livre)Object);
            break;
        case "dvd":
            listeDvd = uneListe.ConvertAll(Object => (Dvd)Object);
            break;
        case "revue":
            listeRevue = uneListe.ConvertAll(Object => (Revue)Object);
            break;
    }
}

```

L'intérêt d'avoir la liste des livres existants dans l'ajout d'un livre est pour le contrôle de l'id du livre. Car chaque livre possède un id unique qui sert d'identifiant dans la base de données. A ce niveau j'ai donc la seconde vue qui est lancé avec tout ce qui est nécessaire à l'ajout d'un livre. Il ne reste plus qu'à tout remplir comme dans l'exemple.

The screenshot shows a Windows application window titled "Ajout d'un LIVRE". The interface includes dropdown menus for "Genre" (set to "Comédie"), "Public" (set to "Tous publics"), and "Rayon" (set to "Presse quotidienne"). Below these are text input fields for "Numéro de document" (00052), "ISBN" (1111111111111), "Titre" (Les blagues de hedi), "Auteur" (Hédi), "Collection" (Classiques Hahette), and "Chemin de l'image". There are also empty dropdowns for "Genre", "Public", and "Rayon". At the bottom left is a note: "\* Champs obligatoire pour ajouter un livre". On the right side are buttons for "Ajouter" (highlighted in blue), "Modifier", and "Annuler".

L'action du bouton Ajouter va appeler la fonction [SuperLivre\(\)](#) qui va retourner un livre et l'envoyer au contrôleur.

```
//Action du bouton Ajouter. Envoi un livre, un dvd ou une revue dans la bdd
//référence | hedi-k, il y a 7 jours | 1 auteur, 7 modifications
private void btnAjouter_Click(object sender, EventArgs e)
{
    switch (quelEnvoi)
    {
        case "livre":
            Livre livre = SuperLivre();
            if (livre != null)
            {
                if (controller.EnvoiDocuments(livre)) // l'api return true si l'ajout a fonctionné
                {
                    MessageBox.Show("Livre ajouté");
                    Vide();
                }
            }
            break;
    }
}
```

[SuperLivre\(\)](#) a pour rôle d'appeler la fonction qui contient le constructeur du livre et de contrôler toutes les entrées de l'utilisateur. Elle retournera un message d'erreur à l'utilisateur si besoin. A noter que je limite les id des livres entre 0 et 2000 pour avoir une cohérence avec les id déjà présentes dans la base de données. Le tout dans un try catch pour essayer de contrôler des incohérences que l'utilisateur peut entrer.

```

//Contrôle des valeurs entrées
2 références | hedi-k, il y a 7 jours | 1 auteur, 4 modifications
private Livre SuperLivre()
{
    try
    {
        Livre livre = ValoriseLivre();
        //contrôle les ComboBox sauf si modification
        if (cbxGenres.Text != "" && cbxPublics.Text != "" && cbxRayons.Text != "" || livreModif != null)
        {
            //contrôle les TextBox
            if (txbTitre.Text != "" && txbIsbnDuree.Text != "" && txbAuteurRealisateurPer.Text != "" && txbCollectionSynopsisDel.Text != "" && txbNumero.Text != "")
            {
                Livre testId = listeLivre.Find(x => x.Id.Equals(txNumero.Text));
                //contrôle que l'id soit disponible sauf si modification
                if (testId == null || livreModif != null)
                {
                    //contrôle la valeur de l'ID
                    if ((int.Parse(txNumero.Text)) > 0 && (int.Parse(txNumero.Text)) < 2000)
                    {
                        //contrôle le format de l'ISBN
                        if (txIsbnDuree.Text.Length == 13)
                        {
                            return livre;
                        }
                        else { MessageBox.Show("ISBN incorrecte. (13 chiffres)"); return null; }
                    }
                    else { MessageBox.Show("Id incorrecte. (compris entre 00001 et 19999)"); return null; }
                }
                else { MessageBox.Show("Cet id est déjà utilisé."); return null; }
            }
            else { MessageBox.Show("Entrez un id, un titre, un auteur, un ISBN et une collection."); return null; }
        }
        else { MessageBox.Show("Sélectionnez un genre, un public et un rayon."); return null; }
    }
    catch (Exception ex) { return null; }
}

```

La fonction `ValoriseLivre()` va créer le livre avec les données entrées et le retourner. Le tout dans un contrôle d'erreur. Et se comporter différemment en cas de modification d'un livre

```

//Méthode pour la création/modification d'un livre
1 référence | hedi-k, il y a 11 jours | 1 auteur, 5 modifications
private Livre ValoriseLivre()
{
    try
    {
        //valorise tous les paramètres
        string id = FormaterId(txNumero.Text);
        string titre = txbTitre.Text;
        string image = txbImage.Text;
        string isbn = txIsbnDuree.Text;
        string auteur = txbAuteurRealisateurPer.Text;
        string collection = txbCollectionSynopsisDel.Text;
        string idGenre = GetIdGenre(cbxGenres.Text);
        string genre = txbGenre.Text;
        string idPublic = GetIdPublic(cbxPublics.Text);
        string Public = txbPublic.Text;
        string idRayon = GetIdRayon(cbxRayons.Text);
        string rayon = txbRayon.Text;
        //cas d'une modification de livre
        // les id pas valorisés ne sont pas des chaînes vides mais null (rappel)
        if (livreModif != null)...
        Livre livreValorise = new Livre(id, titre, image, isbn, auteur, collection, idGenre, genre = null, idPublic, Public = null, idRayon, rayon);
        return livreValorise;
    }
    catch (Exception ex) { return null; }
}

```

En résumé, `ValoriseLivre()` retourne un objet livre à `SuperLivre()` qui sera retourné à la méthode `btnAjouter_Click()`.

Ce livre sera envoyé au contrôleur et en cas de succès de l'enregistrement dans la BDD le contrôleur retourne un true ce qui affiche un message à l'utilisateur et vide toutes les textBox pour un nouvel ajout.

## CONTROLLER

Le contrôleur se trouve dans le fichier [FrmMediatekController.cs](#) Pour son fonctionnement, je m'inspire de l'existant.

L'objet Livre devient un objet <T> ce qui signifie en C# que cela devient un type générique. L'avantage de ce procédé et que je pourrai utiliser le même contrôleur pour l'envoi d'un livre, un dvd ou une revue. Et retournera un booléen (true) si cela a fonctionné.

```
//Cette méthode envoi à l'accès livre, dvd et Revue.  
//En C#, T est un paramètre de type générique.  
//Ce qui implique quelle va traiter indifféremment un objet Livre/Dvd/revue  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool EnvoiDocuments<T>(T unDocument)  
{  
    return access.CreerDocument(unDocument);  
}
```

## DAL

Dans le fichier [Access.cs](#) la méthode [CreerDocument<T>\(\)](#) est appelée. Cette méthode va convertir au format JSON le livre envoyé en paramètre. Et va appeler à son tour la méthode [TraitementRecup\(\)](#) avec comme paramètre POST, le nom du type d'objet en minuscule ici livre et le JSON.

POST détermine pour l'API quelle opération réaliser dans ce cas insérer une information.

Le type d'objet envoyé dans ce cas est un Livre. Il se trouve que livre est le nom de la table dans la base de donnée alors le fait d'envoyer le nom de l'objet en minuscule me permet de sélectionner la table sans ajouter de paramètre pour spécifier que c'est livre.

Dans le JSON si je remplace les espaces par des moins c'est parce que l'API ne sait pas les traiter malgré le fait que le fichier [.htaccess](#) de l'API qui gère le type de caractères à traiter les autorisent. Le tout dans un contrôle d'erreur (try/catch) et la méthode me retournera un booléen.

```
//Envoi l'objet en paramètre à l'API  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool CreerDocument<T>(T unDocument)  
{  
    //Convertit en json l'objet en paramètre  
    String jsonCreerDocument = JsonConvert.SerializeObject(unDocument);  
    try  
    {  
        //Appel TraitementRecup avec en paramètre POST, le nom du type d'objet et le json  
        List<Object> liste = TraitementRecup<Object>(POST, typeof(T).Name.ToLower() + "/" + jsonCreerDocument.Replace(" ", "-"));  
        return (liste != null);  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
    }  
    return false;  
}
```

Je ne détaille pas la méthode [TraitementRecupe\(\)](#) car elle est déjà écrite, son rôle est de traiter la récupération du retour de l'API.

## API

Au sein de l'API, [Mediatekdocuments.php](#) est le point d'entrée, il récupère les paramètres et en fonction du verbe HTTP utilisé ici POST appelle la méthode concerné de la classe [Controle](#).

```

    // traitement suivant le verbe HTTP utilisé
    if ($_SERVER['REQUEST_METHOD'] === 'GET') {
        $controle->get($table, $champs);
    } else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $controle->post($table, $champs);
    } else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
        $controle->put($table, $id, $champs);
    } else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
        $controle->delete($table, $champs);
    }

```



[Controle.php](#) reçoit livre pour table et va donc appeler la méthode `ajoutLivre()` avec en paramètre le contenu du JSON envoyé depuis l'application.

```

public function post($table, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->ajoutLivre($champs);
    } elseif ($table == "dvd") {

```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. Dans le cas d'un ajout de livre il est important de respecter un certain ordre pour l'envoi des requêtes. Comme un livre hérite de livre\_dvd qui hérite lui même de document il est important de créer en premier un document puis un livre\_dvd et enfin un livre sinon cela ne fonctionne pas.

```

public function ajoutLivre($champs) {
    $champsDocument = [
        "id" => $champs["Id"],
        "titre" => $champs["Titre"],
        "image" => $champs["Image"],
        "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"],
        "idGenre" => $champs["IdGenre"]];
    $champsDvdLivre = ["id" => $champs["Id"]];
    $champsLivre = [
        "id" => $champs["Id"],
        "ISBN" => $champs["Isbn"],
        "auteur" => $champs["Auteur"],
        "collection" => $champs["Collection"]];
    $result = $this->insertOne("document", $champsDocument);
    if ($result == null || $result == false) {
        return null;
    }
    $result = $this->insertOne("livres_dvd", $champsDvdLivre);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->insertOne("livre", $champsLivre);
}

```

La méthode `inserOne()` va créer la requête SQL à partir de la table et des champs envoyés mais comme elle est fournie avec l'API je ne vais donc pas la détailler.

Si tout à fonctionné on a le retour d'un booléen à true. Ce booléen qui va déclencher dans la vue le message pour confirmer la création d'un livre.

## Modifier un Livre

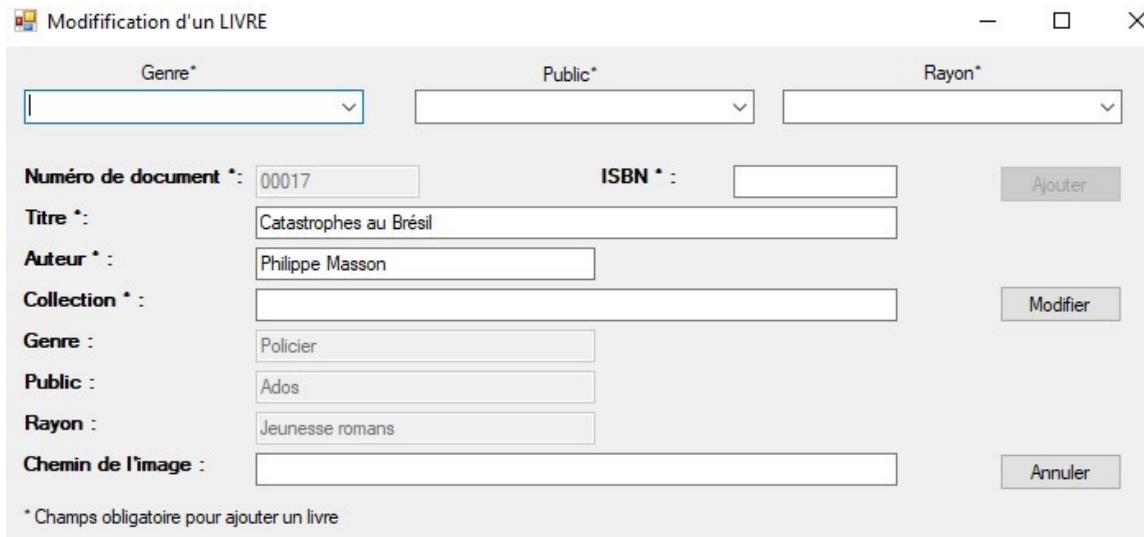
### VIEW

Le cheminement pour la modification d'un livre est similaire à celui de l'ajout d'un livre.

Depuis [FrmMediatek.cs](#) il suffit de sélectionner un livre et de cliquer sur le bouton modifier. Son rôle est de lancer [FrmAjout.cs](#) avec divers paramètres qui vont affecter la seconde vue.

```
//Action du bouton modifier (livre)
//Le boolean est pour l'affichage des TextBoxs genre public rayon et Bouton modifier.
1 référence | hedi-k, il y a 7 jours | 1 auteur, 7 modifications
private void btnModifier_Click(object sender, EventArgs e)
{
    //Contrôle qu'un livre est sélectionné.
    if (dgvLivresListe.CurrentCell != null)
    {
        //Chaîne de caractère qui va influencer les structures conditionnelles de FrmAjout
        string ongletLivre = "livre";
        List<Object> listeObjLivre = lesLivres.ConvertAll(Livre => (Object)Livre); //Convertit la liste de livre en objet
        Object aModifier = (Object)bdgLivresListe.List[bdgLivresListe.Position]; //convertit le livre sélectionné en objet
        //Lance frmAjout avec les paramètres nécessaires à la modification d'un livre
        FrmAjout frmAjout = new FrmAjout(bdgGenres, bdgPublics, bdgRayons, true, ongletLivre, listeObjLivre, aModifier);
        frmAjout.Text = "Modification d'un LIVRE";
        frmAjout.Show();
        this.Hide();
    }
    else{ MessageBox.Show("selectionnez un livre !");}
}
```

[FrmAjout.cs](#) va s'exécuter mais l'affichage sera modifier, il y aura cette fois les informations concernant le livre envoyé en paramètre qui seront affichées.



Le remplissage des textBox est le fait de la fonction [ChargeObjet\(\)](#) qui valorise chaque textBox par l'élément correspondant du livre envoyé en paramètre.

```
//Méthode pour la modification d'un livre ou un dvd
1 référence | hedi-k, il y a 7 jours | 1 auteur, 4 modifications
private void ChargeObjet(string onglet, Object aModifier)
{
    switch (onglet)
    {
        case "livre":
            livreModif = (Livre)aModifier;
            if (livreModif != null)
            {
                txbNumero.Text = livreModif.Id;
                txbTitre.Text = livreModif.Titre;
                txbImage.Text = livreModif.Image;
                txbIsbnDuree.Text = livreModif.Isbn;
                txbAuteurRealisateurPer.Text = livreModif.Auteur;
                txbCollectionSynopsisDel.Text = livreModif.Collection;
                txbGenre.Text = livreModif.Genre;
                txbPublic.Text = livreModif.Public;
                txbRayon.Text = livreModif.Rayon;
            }
            break;
        case "dvd":
```

L'action du bouton modifier fonctionne sur le même principe que celui de l'ajout d'un livre.

```
//Action du bouton Modifier
1 référence | hedi-k, il y a 7 jours | 1 auteur, 7 modifications
private void btnModifier_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Modifier ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        switch (quelEnvoi)
        {
            case "livre":
                Livre livre = SuperLivre();
                if (livre != null)
                {
                    if (controller.ModifierDocuments(livre))// l'api return true si l'ajout a fonctionné
                    {
                        MessageBox.Show("Livre modifié");
                    }
                }
                break;
            case "dvd":
```

La fonction [SuperLivre\(\)](#) va retourner un livre et il sera envoyé au contrôleur. Si cela fonctionne un booléen a true sera retourné et affichera un message à l'utilisateur.

[SuperLivre\(\)](#) va modifier un test, celui des comboBox en effet on peut ne pas vouloir modifier le genre, rayon ou public alors on doit autoriser dans le cas d'une modification de les laisser vide.

```
1 référence | hedi-k, il y a 7 jours | 1 auteur, 4 modifications
private Livre SuperLivre()
{
    try
    {
        Livre livre = ValoriseLivre();
        //contrôle les ComboBox sauf si modification
        if (cbxGenres.Text != "" && cbxPublics.Text != "" && cbxRayons.Text != "" || livreModif != null)
        {
            //contrôle les TextBox
```



Pareil pour la valorisation d'un livre, si on ne change pas les comboBox il faut ajouter une condition pour préciser que c'est une modification et dans ce cas il faut récupérer les id pour les comboBox.

```
// les id pas valorisés ne sont pas des chaînes vides mais null (rappel)
if (livreModif != null)
{
    //si le genre n'est pas modifier
    if (idGenre == null)
    {
        idGenre = GetIdGenre(txbGenre.Text);
    }
    //si le public n'est pas modifier
    if (idPublic == null)
    {
        idPublic = GetIdPublic(txbPublic.Text);
    }
    //si le rayon n'est pas modifier
    if (idRayon == null)
    {
        idRayon = GetIdRayon(txbRayon.Text);
    }
}
Livre livreValorise = new Livre(id, titre, image, isbn, auteur, collection, idGenre, genre = null, idPublic, Public = null, idRayon, rayon = null);
return livreValorise;
```

Les fonction [GetIdXxx\(\)](#) fonctionnent toute de la même manier. Exemple pour un genre, elle récupère le contenu de la textBoxGenre. Charge la liste qui contient tout les genres (la classe genre hérite de catégorie c'est pour cela la liste de catégorie). Et parcourt la liste à la recherche d'un labelle correspondant. Elle retourne ensuite son Id.

```
private void vidé()...
//Retourne l'ID du genre sélectionné dans la cbxGenre
6 références | hedi-k, il y a 7 jours | 1 auteur, 2 modifications
private string GetIdGenre(string unGenre)
{
    List<Categorie> uneListe = controller.GetAllGenres();
    foreach (Categorie uneCategorie in uneListe)
    {
        if (uneCategorie.Libelle == unGenre)
        {
            return uneCategorie.Id;
        }
    }
    return null;
}
```

Le livre valorisé est donc envoyé au contrôleur.

## CONTROLLER

Le contrôleur se trouve dans le fichier [FrmMediatekController.cs](#) il fonctionne comme pour l'ajout d'un livre, il attend un type générique à envoyer à l'access.

```
//Modifie un livre, un dvd ou une revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocuments<T>(T unDocument)
{
    return access.ModifierDocument(unDocument);
}
```

## DAL

Pour la modification d'un livre, il faut changer le verbe envoyé, cela devient PUT mais l'API attend aussi l'id du livre à envoyer. Alors il faut re convertir unDocument d'un type générique en type Livre pour récupérer l'Id. Il est intéressant de noter que l'on peut convertir un objet Livre en type générique et l'utiliser mais que l'on ne peut pas récupérer des membres de la classe Livre depuis un type générique. Concrètement unDocument.Id cela ne fonctionne pas.

```
//Envoi l'objet en paramètre à l'API pour modification
//référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonModifierDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de modification
        string id = "";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
        }
        //unDocument.Id ça ne marche pas !!
        List<Object> liste = TraitementRecup<Object>(PUT, typeof(T).Name.ToLower() + "/" + id + "/" + jsonModifierDocument.Replace(" ", "-"));
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

## API

```
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
    $controle->put($table, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controle->delete($table, $champs);
}
```

Dans l'API on suit le même cheminement que précédemment. [Mediatekdocuments.php](#) est le point d'entrée, il récupère les paramètres et en fonction du verbe HTTP utilisé ici PUT appelle la méthode concerné de la classe Controle. [Controle.php](#) reçoit livre pour table, l'id et les champs et va donc appeler la méthode [modifiLivre\(\)](#) avec en paramètre.

```
public function put($table, $id, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->modifiLivre($id, $champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->modiDvd($id, $champs);
```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. A noter qu'il est demandé à ce que l'id d'un document ne puisse pas être modifier. Donc bien que cela ne soit pas possible dans l'application on constate que la fonction [modifiLivre\(\)](#) ne contient pas de champs pour faire cette manipulation.

```

public function modifiLivre($id, $champs) {

    $champsDocument = [
        "titre" => $champs["Titre"],
        "image" => $champs["Image"],
        "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"],
        "idGenre" => $champs["IdGenre"]];

    $champsLivre = [ // "id" => $champs["Id"],
        "ISBN" => $champs["Isbn"],
        "auteur" => $champs["Auteur"],
        "collection" => $champs["Collection"]];

    $result = $this->updateOne("document", $id, $champsDocument);
    if ($result == null || $result == false) {
        return null;
    }

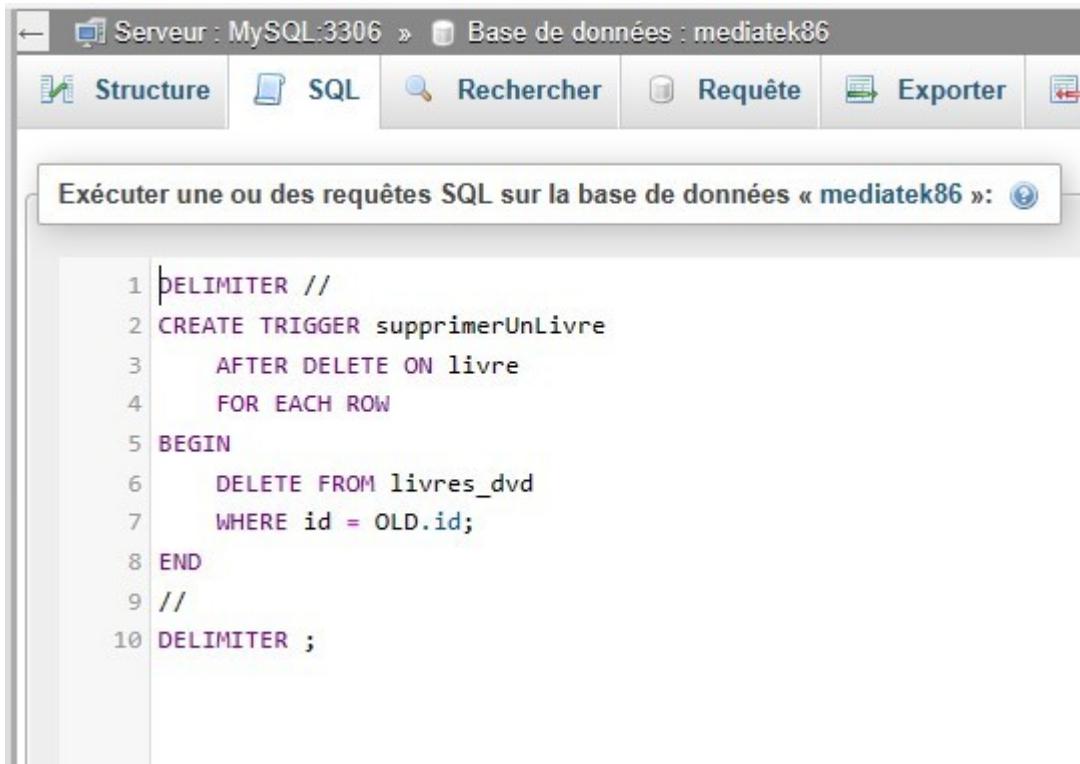
    return $this->updateOne("livre", $id, $champsLivre);
}

```

Ensuite `updateOne()` est appelé et retourne true si cela fonctionne. N'ayant pas écrit le reste de l'API je ne vais pas le détailler.

## Supprimer un livre

Pour la suppression d'un livre il va falloir créer un trigger qui prend en compte les contraintes de partitions des héritages sur les tables document et livres\_dvd. L'idée est de supprimer uniquement un tuple de la table livre et qu'automatiquement la base de données efface les tuples des tables livre\_dvd et document associés.



The screenshot shows the MySQL Workbench interface with the following details:

- Server:** MySQL:3306
- Database:** mediatek86
- Tab:** SQL
- Content:** SQL query to create a trigger named `supprimerUnLivre` that triggers after a delete operation on the `livre` table. It uses a delimiter to prevent MySQL from interpreting the semicolon as a command separator. The trigger logic is to delete the corresponding row from the `livres_dvd` table where the `id` matches the deleted row's `id`.

```

1 DELIMITER //
2 CREATE TRIGGER supprimerUnLivre
3     AFTER DELETE ON livre
4     FOR EACH ROW
5 BEGIN
6     DELETE FROM livres_dvd
7     WHERE id = OLD.id;
8 END
9 //
10 DELIMITER ;

```

Après un effacement sur la table livre, le trigger va effacer sur la table livres\_dvd le tuple qui contient la même id.

Et le second trigger procède de manière identique mais sur la table document après effacement sur la table livres\_dvd



The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL 3306
- Base de données: mediatek86
- Tab: Structure (selected)
- Tab: SQL
- Tab: Rechercher
- Tab: Requête (selected)
- Tab: Exporter
- SQL Editor Content:

```
1 DELIMITER //
2 CREATE TRIGGER supprimerUnDocument
3     AFTER DELETE ON livres_dvd
4     FOR EACH ROW
5 BEGIN
6     DELETE FROM document
7     WHERE id = OLD.id;
8 END
9 //
10 DELIMITER ;
```

Il est imposé qu'un document ne peut être supprimé uniquement si aucun exemplaire ou commande n'est rattaché. J'ai donc réalisé cette tâche à l'issue de la mission 3 car les commandes et exemplaires ne sont pas encore traités.

## VIEW

Je code l'action du bouton Supprimer de [FrmMediatek.cs](#). Pour supprimer, je contrôle qu'un livre est sélectionné dans liste des livres. Suite à cela je m'assure qu'aucun exemplaires n'est rattachés à ce livre en contrôlant que la liste des exemplaires sur ce livre sélectionné soit null. (Cette liste apparaîtra ultérieurement mission3). Et pour m'assurer qu'aucune commandes porte sur ce livre, je compare l'id du livre et avec celui des livre commandés. (Traitements des commande mission2).

```
//Action du bouton supprimer (livre)
//référence | hedi-k, il y a 8 jours | 1 auteur, 4 modifications
private void btnSupprimer_Click(object sender, EventArgs e)
{
    if (dgvLivresListe.CurrentCell != null)      //vérification de selection d'un livre
    {
        if (dgvLivreExemplaire.CurrentCell == null)//Vérification qu'aucun exemplaire est rattaché
        {
            Livre livre = (Livre)bdgLivresListe.List[bdgLivresListe.Position];
            CommandeDocument cmdLivre = listCmdLivres.Find(x => x.IdLivreDvd.Equals(livre.Id));
            if (cmdLivre == null) //Vérification qu'aucune commande n'est en cours.
            {
                if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    controller.SupprimerDocument(livre);
                    TabLivres_Enter(null, null);
                }
            }
            else { MessageBox.Show("Une commande est en cours sur ce livre, il ne peut pas être effacé !"); }
        }
        else { MessageBox.Show("Les livres qui possèdent un exemplaire ne peuvent pas être supprimer !"); }
    }
    else { MessageBox.Show("selectionner un livre !"); }
}
```

Si les conditions sont respectés, le livre sélectionné est envoyé au contrôleur.

## CONTROLLER

Le contrôleur traite le livre comme un objet de type générique et appelle l'accès.

```
//Supprime un livre, dvd ou revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool SupprimerDocument<T>(T unDocument)
{
    return access.SupprimerDocument(unDocument);
}
```

## DAL

La méthode pour supprimer un livre a besoin de la table et de l'id du livre à supprimer. C'est pour cela que je le convertis de type générique à Livre pour récupérer l'id du livre. Ensuite j'appelle le verbe HTTP DELETE pour la suppression. Pour trouver le bon formatage à envoyer j'ai fait des essais avec PostMan pour voir quel format était attendu par l'API.

```
//Supprime le document en paramètre
1 référence | 0 modification | 0 auteur, 0 modification
public bool SupprimerDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonSupprimerDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de modification
        string id = "";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
            case Dvd dvd:
                id = dvd.Id;
                break;
        }
        List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + "Id\":" + id + "}");
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

## API

Dans l'API on suit le même cheminement que précédemment. [Mediatekdocuments.php](#) est le point d'entrée, il récupère les paramètres et en fonction du verbe HTTP utilisé ici DELETE appelle la méthode concernée de la classe `Control`

```
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controler->delete($table, $champs);
}
```

[Controle.php](#) va donc appeler la méthode `delete()` avec en paramètre la table ici livre et l'id du livre sélectionné. Cette fonction va appeler `delete()` de [AccesBDD.php](#) pour construire la requête SQL.

```
public function delete($table, $champs) {
    $result = $this->accessBDD->delete($table, $champs);
    if ($result == null || $result == false) {
        $this->reponse(400, "requete invalide");
    } else {
        $this->reponse(200, "OK");
    }
}
```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. Ces fonctions sont fournies avec l'API alors je ne les détaille pas.

```
public function delete($table, $champs) {
    if ($this->conn != null) {
        // construction de la requête
        $requete = "delete from $table where ";
        foreach ($champs as $key => $value) {
            $requete .= "$key=:{$key} and ";
        }
        // (enlève le dernier and)
        $requete = substr($requete, 0, strlen($requete) - 5);
        return $this->conn->execute($requete, $champs);
    } else {
        return null;
    }
}
```

Cette requête construite va donc supprimer le tuple de la table livre qui contient l'id. Les triggers de la BDD se chargent de supprimer les tuples des tables livres\_dvd et document.

## Ajouter un DVD

Pour travailler sur un objet de type Dvd il faut étudier ce que représente un dvd. Le fonctionnement de la base de donnée ne diffère pas à celle d'un Livre et le modèle est très proche de celui d'un livre. Il y a une petite différence dans les types de données à traiter alors il faudra penser à convertir au bon format la durée d'un dvd qui est un entier et non une chaîne de caractère.

```
public class Dvd : LivreDvd
{
    5 références | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public int Duree { get; }
    6 références | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public string Realisateur { get; }
    3 références | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public string Synopsis { get; }

    1 référence | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public Dvd(string id, string titre, string image, int duree, string realisateur, string synopsis,
               string idGenre, string genre, string idPublic, string lePublic, string idRayon, string rayon)
        : base(id, titre, image, idGenre, genre, idPublic, lePublic, idRayon, rayon)
    {
        this.Duree = duree;
        this.Realisateur = realisateur;
        this.Synopsis = synopsis;
    }
}
```

## VIEW

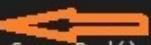
J'ajoute un bouton Ajouter sur l'onglet dvd de [FrmMediatek.cs](#) Ce bouton aura pour fonction d'ouvrir [FrmAjout.cs](#) avec divers paramètres qui vont affecter la seconde vue dans son affichage et le fonctionnement de ses méthodes.

```
//Action du btn AjoutDvd lance la forme Ajout pour un dvd
1 référence | hedi-k, il y a 13 jours | 1 auteur, 3 modifications
private void btnAjoutDvd_Click(object sender, EventArgs e)
{
    //Chaîne de caractère qui va influencer les structures conditionnelles de FrmAjout
    string ongletDvd = "dvd";
    //Converti la liste de livres en liste d'objets
    List<Object> listeObjDvd = lesDvd.ConvertAll(Dvd => (Object)Dvd);
    //Lance frmAjout avec les paramètres nécessaires à l'ajout d'un livre
    FrmAjout frmAjout = new FrmAjout(bdgGenres, bdgPublics, bdgRayons, false, ongletDvd, listeObjDvd);
    frmAjout.Text = "Ajout d'un DVD";
    frmAjout.Show();
    this.Hide();
}
```

Le fonctionnement est identique à celui du livre alors je ne vais pas détailler la gestion de l'affichage.

Une fois tout les champs remplis on clique sur le bouton Ajouter et cela active la méthode du bouton. Cette fois la chaîne de caractère contenue dans la variable `quelEnvoi` contient dvd donc c'est la valorisation d'un dvd qui est lancé et un dvd qui est envoyé au contrôleur.

```
//Action du bouton Ajouter. Envoi un livre, un dvd ou une revue dans la bdd
1 référence | hedi-k, il y a 8 jours | 1 auteur, 7 modifications
private void btnAjouter_Click(object sender, EventArgs e)
{
    switch (quelEnvoi)
    {
        case "livre":
            Livre livre = SuperLivre();
            if (livre != null)...
            break;

        case "dvd": 
            Dvd dvd = SuperDvd();
            if (dvd != null)
            {
                if (controller.EnvoiDocuments(dvd))// l'api return true si l'ajout a fonctionné
                {
                    MessageBox.Show("DVD ajouté");
                    Vide();
                }
            }
            break;
    }
}
```

La fonction `SuperDvd()` va appeler `ValoriseDvd()` et vérifier que toutes les valeur entrés soient cohérentes avant de retourner un dvd. Je limite les choix d'id pour les dvd entre 2000 et 3000 pour que les valeurs soit cohérentes avec les données déjà présentes.

```

//Contrôle des valeurs entrées
2 références | hedi-k, il y a 8 jours | 1 auteur, 4 modifications
private Dvd SuperDvd()
{
    try
    {
        Dvd dvd = ValoriseDvd();
        //contrôle les ComboBox sauf si modification
        if (cbxGenres.Text != "" & cbxPublics.Text != "" & cbxRayons.Text != "" || dvdModif != null)
        {
            //contrôle les TextBox avant appel du constructeur
            if (txbTitre.Text != "" & txbIsbnDuree.Text != "" & txbAuteurRealisateurPer.Text != "" & txbCollectionSynopsisDel.Text != "" & txbNumero.Text != "")
            {
                //contrôle la valeur de l'ID
                if ((int.Parse(txbNumero.Text)) > 19999 && (int.Parse(txbNumero.Text)) < 30000)
                {
                    Dvd testId = listeDvd.Find(x => x.Id.Equals(txbNumero.Text));
                    if (testId == null || dvdModif != null)
                    {
                        //contrôle la valeur de Durée
                        if (int.Parse(txbIsbnDuree.Text) > 1)
                        {
                            return dvd;
                        }
                        else { MessageBox.Show("Valeur de durée incorrecte."); return null; }
                    }
                }
            }
        }
    }
}

```

Pour la valorisation d'un dvd on fait attention a convertir en int la durée.

```

private Dvd ValoriseDvd()
{
    try // nécessaire si txbIsbnDuree est null, le Parse fait planter.
    {
        //valorise tout les paramètres
        string id = txbNumero.Text;
        string titre = txbTitre.Text;
        string image = txbImage.Text;
        int duree = int.Parse(txbIsbnDuree.Text);
        string realisateur = txbAuteurRealisateurPer.Text;
        string synopsis = txbCollectionSynopsisDel.Text;
        string idGenre = GetIdGenre(cbxGenres.Text);
        string genre = txbGenre.Text;
        string idPublic = GetIdPublic(cbxPublics.Text);
        string Public = txbPublic.Text;
        string idRayon = GetIdRayon(cbxRayons.Text);
        string rayon = txbRayon.Text;

        if (dvdModif != null)[...]
        Dvd dvdValorise = new Dvd(id, titre, image, duree, realisateur, synopsis, idGenre, genre = null, idPublic, Public = null, idRayon, rayon = null);
        return dvdValorise;
    }
    catch (Exception ex) { return null; }
}

```

Le dvd est donc créer avec les entrées des divers textBox et comboBox, le tout est envoyé au contrôleur.

## CONTROLLE

Dans le contrôleur c'est toujours la même fonction est appellée et qui transmet à l'access. Comme elle attend un type générique elle peut traiter un DVD.

```

//Cette méthode envoi à l accès livre, dvd et Revue.
//En C#, T est un paramètre de type générique.
//Ce qui implique quelle va traiter indifféremment un objet Livre/Dvd/revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool EnvoiDocuments<T>(T unDocument)
{
    return access.CreerDocument(unDocument);
}

```

## DAL

Dans le fichier [Access.cs](#) la méthode `CreerDocument<T>()` est appelée. Cette méthode va convertir au format JSON le dvd envoyé en paramètre. Le nom du type d'objet en minuscule ici dvd correspond au nom de la table à sélectionner pour l'API, même fonctionnement que pour un livre.

```
//Envoi l'objet en paramètre à l'API
//référence | modification | auteur, modification
public bool CreerDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonCreerDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //Appel TraitementRecup avec en paramètre POST, le nom du type d'objet et le json
        List<Object> liste = TraitementRecup<Object>(POST, typeof(T).Name.ToLower() + "/" + jsonCreerDocument.Replace(" ", "-"));
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

## API

Au sein de l'API, [Mediatekdocuments.php](#) est le point d'entrée. Récupère les paramètres et en fonction du verbe HTTP utilisé ici POST appelle la méthode concernée de la classe [Controle](#).

```
// traitement suivant le verbe HTTP utilisé
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $controle->get($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controle->post($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
    $controle->put($table, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controle->delete($table, $champs);
}
```

[Controle.php](#) reçoit dvd pour table et va donc appeler la méthode `ajoutDvd()` avec en paramètre le contenu du JSON envoyé depuis l'application.

```
public function post($table, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->ajoutLivre($champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->ajoutDvd($champs); ←
    } elseif ($table == "revue") {
        $result = $this->accessBDD->ajoutRevue($champs);
    } elseif ($table == "commandeLivreOuDvd") {
        $result = $this->accessBDD->ajoutCommandeLivreOuDvd($champs);
    } elseif ($table == "commandeAbonnementRevue") {
        $result = $this->accessBDD->ajoutCommandeAbonnementRevue($champs);
    } else {
        $result = $this->accessBDD->insertOne($table, $champs);
    }
}
```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés

```
public function ajoutDvd($champs) {
    $champsDocument = [
        "id" => $champs["Id"],
        "titre" => $champs["Titre"],
        "image" => $champs["Image"],
        "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"],
        "idGenre" => $champs["IdGenre"]];
    $champsDvdLivre = ["id" => $champs["Id"]];

    $champsDvd = [
        "id" => $champs["Id"],
        "synopsis" => $champs["Synopsis"],
        "realisateur" => $champs["Realisateur"],
        "duree" => $champs["Duree"]];

    $result = $this->insertOne("document", $champsDocument);
    if ($result == null || $result == false) {
        return null;
    }

    $result = $this->insertOne("livres_dvd", $champsDvdLivre);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->insertOne("dvd", $champsDvd);
}
```

Il est important de bien respecter l'ordre des requêtes pour une création, en premier on ajoute un document, ensuite un livres\_dvd et pour finir un dvd. Sinon cela ne respectera pas l'héritage et ne fonctionnera pas.

## Modifier un DVD

### VIEW

Le cheminement pour la modification d'un dvd est similaire à celle de l'ajout d'un livre. Depuis [FrmMediatek.cs](#) il suffit de sélectionner un dvd et de cliquer sur le bouton modifier. Son rôle est de lancer [FrmAjout.cs](#) avec divers paramètres qui vont affecter la seconde vue.

```
private void btnModifDvd_Click(object sender, EventArgs e)
{
    if (dgvDvdListe.CurrentCell != null)
    {
        string ongletDvd = "dvd";
        List<Object> listeObjDvd = lesDvd.ConvertAll(Dvd => (Object)Dvd);
        Object aModifier = (Object)bdgDvdListe.List[bdgDvdListe.Position];
        FrmAjout frmAjout = new FrmAjout(bdgGenres, bdgPublics, bdgRayons, true, ongletDvd, listeObjDvd, aModifier);
        frmAjout.Text = "Modification d'un DVD";
        frmAjout.Show();
        this.Hide();
    }
    else{ MessageBox.Show("Selectionnez un Dvd !");}
}
```

Depuis [FrmAjout.cs](#) l'affichage et les divers fonctions vont s'adapter comme pour un livre mais pour les données qui concernent un dvd.

Une fois modifier il suffit de cliquer sur le bouton modifier pour lancer la modification.

```
//Action du bouton Modifier
1 référence | hedi-k, il y a 8 jours | 1 auteur, 7 modifications
private void btnModifier_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Modifier ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        switch (quelEnvoi)
        {
            case "livre":
                Livre livre = SuperLivre();
                if (livre != null)...
                break;
            case "dvd": ←
                Dvd dvd = SuperDvd();
                if (dvd != null)
                {
                    if (controller.ModifierDocuments(dvd))
                    {
                        MessageBox.Show("DVD modifié");
                    }
                }
                break;
        }
    }
}
```

SuperDvd() va se comporter différemment comme c'est une modification, il faut autoriser l'absence de changement des comboBox et que l'id du dvd existe.

```
private Dvd SuperDvd()
{
    try
    {
        Dvd dvd = ValoriseDvd();
        //contrôle les ComboBox sauf si modification
        if (cbxGenres.Text != "" && cbxPublics.Text != "" && cbxRayons.Text != "" || dvdModif != null)
        {
            //contrôle les TextBox avant appel du constructeur
            if (txbTitre.Text != "" && txbIsbnDuree.Text != "" && txbAuteurRealisateurPer.Text != "" && tx...)
            {
                //contrôle la valeur de l'ID
                if ((int.Parse(txbNumero.Text)) > 19999 && (int.Parse(txbNumero.Text)) < 30000)
                {
                    Dvd testId = listeDvd.Find(x => x.Id.Equals(txbNumero.Text));
                    if (testId == null || dvdModif != null)
                    {
                        //contrôle la valeur de Durée
                        if (int.Parse(txbIsbnDuree.Text) > 1)
                        {
                            return dvd;
                        }
                    }
                }
            }
        }
    }
}
```

La valorisation d'un dvd doit aussi prendre en compte ces cas, si on ne sélectionne rien dans les comboBox, elle doit récupérer leur id sur les labelles déjà existant.

```
if (dvdModif != null) //cas d'une modification d'un dvd
{
    //si le Genre n'est pas modifier
    if (idGenre == null)
    {
        idGenre = GetIdGenre(txbGenre.Text);
    }
    //si le public n'est pas modifier
    if (idPublic == null)
    {
        idPublic = GetIdPublic(txbPublic.Text);
    }
    //si le rayon n'est pas modifier
    if (idRayon == null)
    {
        idRayon = GetIdRayon(txbRayon.Text);
    }
}
```

## CONTROLLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access. Comme elle attend un type générique elle peut traiter un DVD.

```
//Modifie un livre, un dvd ou une revue  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool ModifierDocuments<T>(T unDocument)  
{  
    return access.ModifierDocument(unDocument);  
}
```

## DAL

Pour la modification d'un dvd, il faut changer le verbe envoyé, cela devient PUT mais l'API attend aussi l'id du dvd à envoyer.

```
public bool ModifierDocument<T>(T unDocument)  
{  
    //Convertit en json l'objet en paramètre  
    String jsonModifierDocument = JsonConvert.SerializeObject(unDocument);  
    try  
    {  
        //L'id est nécessaire pour l'API en cas de modification  
        string id = "";  
        switch (unDocument)  
        {  
            case Livre livre:  
                id = livre.Id;  
                break;  
            case Dvd dvd:  
                id = dvd.Id;   
                break;  
        }  
        List<Object> liste = TraitementRecup<Object>(PUT, typeof(T).Name.ToLower() + "/" + id + "/" + jsonModifierDocument);  
        return (liste != null);  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
    }  
    return false;  
}
```

## API

Au sein de l'API, [Mediatekdocuments.php](#) Récupère les paramètres et en fonction du verbe HTTP utilisé ici PUT appelle la méthode concerné de la classe Controle.

```
// traitement suivant le verbe HTTP utilisé  
if ($_SERVER['REQUEST_METHOD'] === 'GET') {  
    $controle->get($table, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    $controle->post($table, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {   
    $controle->put($table, $id, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {  
    $controle->delete($table, $champs);  
}
```

Le contrôleur va sélectionner la fonction put et comme c'est un dvd il va appeler la fonction modifDvd()

```
public function put($table, $id, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->modifiLivre($id, $champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->modifDvd($id, $champs);
    } elseif ($table == "revue") {
        $result = $this->accessBDD->modifiRevue($id, $champs);
    } elseif ($table == "commandeLivreOuDvd") {
```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. Et on ne met pas de paramètre concernant l'id du document car il est demandé de ne pas pouvoir le changer.

```
public function modifDvd($id, $champs) {

    $champsDocument = [
        "titre" => $champs["Titre"],
        "image" => $champs["Image"],
        "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"],
        "idGenre" => $champs["IdGenre"]
    ];

    $champsDvd = [
        "synopsis" => $champs["Synopsis"],
        "realisateur" => $champs["Realisateur"],
        "duree" => $champs["Duree"]
    ];

    $result = $this->updateOne("document", $id, $champsDocument);
    if ($result == null || $result == false) {
        return null;
    }

    return $this->updateOne("dvd", $id, $champsDvd);
}
```

## Supprimer un dvd

Pour la suppression d'un dvd il va falloir créer un trigger qui prend en compte les contraintes de partitions des héritages sur la table livres\_dvd. Comme pour la suppression d'un livre, on supprime un tuple dans la table dvd et les trigger suppriment les tuples associés des autres tables.

Il est imposé qu'un document ne peut être supprimé uniquement si aucun exemplaire ou commande n'est rattaché. J'ai donc réalisé cette tâche à l'issue de la mission 3 car les commandes et exemplaires ne sont pas encore traités.

## VIEW

Je code l'action du bouton Supprimer de [FrmMediatek.cs](#). Pour supprimer, je contrôle qu'un dvd est sélectionné dans liste des dvd. Suite à cela je m'assure qu'aucun exemplaire n'est rattachés à ce dvd en contrôlant que la liste des exemplaires sur ce dvd sélectionné soit null. (Cette liste apparaîtra ultérieurement mission3). Et pour m'assurer qu'aucune commandes portent sur ce dvd, je compare l'id du dvd et avec celui des dvd commandés. (Traitements des commande mission2).

```
private void btnSupprimerDvd_Click(object sender, EventArgs e)
{
    //contrôle la selection
    if (dgvDvdListe.CurrentCell != null)
    {
        //contrôle un exemplaire
        if (dgvDvdExemplaire.CurrentCell == null)
        {
            Dvd dvd = (Dvd)bdgDvdListe.List[bdgDvdListe.Position];
            CommandeDocument cmdDvd = listCmdDvds.Find(x => x.IdLivreDvd.Equals(dvd.Id));
            //contrôle une commande
            if (cmdDvd == null)
            {
                if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    controller.SupprimerDocument(dvd);
                    tabDvd_Enter(null, null);
                }
            }
        }
    }
}
```

Si tous les tests sont bon, on passe au contrôleur.

## CONTROLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access. Comme elle attend un type générique elle peut traiter un DVD.

```
//Supprime un livre, dvd ou revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool SupprimerDocument<T>(T unDocument)
{
    return access.SupprimerDocument(unDocument);
}
```

## DAL

La méthode pour supprimer un dvd à besoin de la table et de l'id du DVD à supprimer. Ensuite j'appelle le verbe HTTP DELETE pour la suppression.

```
public bool SupprimerDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonSupprimerDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de modification
        string id = "";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
            case Dvd dvd:
                id = dvd.Id; <-- Orange arrow
                break;
        }
        List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + "Id" + ":" + id + "}");
        return (liste != null);
    }
}
```

## API

Au sein de l'API, [Mediatekdocuments.php](#) Récupère les paramètres et en fonction du verbe HTTP utilisé ici DELETE appelle la méthode concernée de la classe Controle.

```
// traitement suivant le verbe HTTP utilisé
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $controle->get($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controle->post($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
    $controle->put($table, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controle->delete($table, $champs); ←
}
```

[Controle.php](#) va donc appeler la méthode `delete()` avec en paramètre la table ici dvd et l'id du dvd sélectionné. Cette fonction va appeler `delete()` de [AccesBDD.php](#) pour construire la requête SQL.

```
public function delete($table, $champs) {
    $result = $this->accessBDD->delete($table, $champs);
    if ($result == null || $result == false) {
        $this->reponse(400, "requete invalide");
    } else {
        $this->reponse(200, "OK");
    }
}
```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés.

```
public function delete($table, $champs) {
    if ($this->conn != null) {
        // construction de la requête
        $requete = "delete from $table where ";
        foreach ($champs as $key => $value) {
            $requete .= "$key=:key and ";
        }
        // (enlève le dernier and)
        $requete = substr($requete, 0, strlen($requete) - 5);
        return $this->conn->execute($requete, $champs);
    } else {
        return null;
}
```

Cette requête construite va donc supprimer le tuple de la table dvd qui contient l'id.  
Les triggers de la BDD se chargent de supprimer les tuples des tables livres\_dvd et document.

## Ajouter une Revue

Pour la gestion des revues, il faut comprendre leur fonctionnement. Elles héritent directement de document, il y a une partition sur héritage sur la table document avec les tables revue et live\_dvd. Les éléments que composent une revue différents également et il faut en tenir compte.

```
public class Revue : Document
{
    6 références | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public string Periodicite { get; set; }
    6 références | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public int DelaiMiseADispo { get; set; }

    1 référence | hedi-k, il y a 19 jours | 1 auteur, 1 modification
    public Revue(string id, string titre, string image, string idGenre, string genre,
        string idPublic, string lePublic, string idRayon, string rayon,
        string periodicite, int delaiMiseADispo)
        : base(id, titre, image, idGenre, genre, idPublic, lePublic, idRayon, rayon)
    {
        Periodicite = periodicite;
        DelaiMiseADispo = delaiMiseADispo;
    }

}
```

Donc je procède comme pour les précédents, j'ajoute trois boutons Ajouter, Modifier et Supprimer dans l'onglet revue.

## VIEW

J'ajoute un bouton Ajouter sur l'onglet revue de [FrmMediatek.cs](#) Ce bouton aura pour fonction d'ouvrir [FrmAjout.cs](#) avec divers paramètres qui vont affecter la seconde vue dans son affichage et le fonctionnement de ses méthodes.

```
//Action, du bouton Ajouter (Revue)
1 référence | hedi-k, il y a 7 jours | 1 auteur, 3 modifications
private void btnAjoutRevue_Click(object sender, EventArgs e) // List<Revue> lesRevues
{
    //Chaine de caractère qui va influencer FrmAjout
    string ongletRevue = "revue";
    //Converti la liste de revues en liste d'objets
    List<Object> listeObjRevue = lesRevues.ConvertAll(Revue => (Object)Revue);
    //Lance frmAjout avec les paramètres nécessaires à l'ajout d'un livre
    FrmAjout frmAjout = new FrmAjout(bdgGenres, bdgPublics, bdgRayons, false, ongletRevue, listeObjRevue);
    frmAjout.Text = "Ajout d'une REVUE";
    frmAjout.Show();
    this.Hide();
}
```

Une fois tout les champs remplis on clique sur le bouton Ajouter. Qui envoi la revue créée au contrôleur.

```

private void btnAjouter_Click(object sender, EventArgs e)
{
    switch (quelEnvoi)
    {
        case "livre":
            Livre livre = SuperLivre();
            if (livre != null)...
            break;

        case "dvd":
            Dvd dvd = SuperDvd();
            if (dvd != null)...
            break;

        case "revue": <-->
            Revue revue = SuperRevue();
            if (revue != null)
            {
                if (controller.EnvoiDocuments(revue))
                {
                    MessageBox.Show("Revue ajouté");
                    Vide();
                }
            }
            break;
    }
}

```

La fonction `SuperRevue()` contrôle que tous les champs entrés par l'utilisateur sont correctes.

```

//Contrôle des valeurs entrées
2 références | hedik, il y a 8 jours | 1 auteur, 3 modifications
private Revue SuperRevue()
{
    try
    {
        Revue revue = ValoriseRevue();
        //Contrôle les comboBox sauf si modification
        if (cbxGenres.Text != "" && cbxPublics.Text != "" && cbxRayons.Text != "" || revueModif != null)
        {
            //Contrôle les textBox avant appel du constructeur
            if (txbTitre.Text != "" && txbAuteurRealisateurPer.Text != "" && txbCollectionSynopsisDel.Text != "" && txbNumero.Text != "")
            {
                if ((int.Parse(txbCollectionSynopsisDel.Text) > 1) && (int.Parse(txbCollectionSynopsisDel.Text) < 1000))
                {
                    if ((int.Parse(txNumero.Text)) > 10000 && (int.Parse(txNumero.Text)) < 20000)
                    {
                        //Contrôle que l'id n'est pas déjà utilisé avant appel du constructeur
                        Revue testId = listeRevue.Find(x => x.Id.Equals(txNumero.Text));
                        if (testId == null || revueModif != null)
                        {
                            return revue;
                        }
                    }
                }
            }
        }
    }
}

```

Et la fonction `ValoriseRevue()` crée l'objet revue. Il faut faire attention aux types qui diffèrent des précédents livre et dvd.

```

private Revue ValoriseRevue()
{
    try // nécessaire, si txbCollectionSynopsisDel est null, le Parse fait planter.
    {
        //valorise tout les paramètres
        string id = txNumero.Text;
        string titre = txbTitre.Text;
        string image = txbImage.Text;
        string periodicite = txbAuteurRealisateurPer.Text;
        int delaiMiseADispo = int.Parse(txbCollectionSynopsisDel.Text);
        string idGenre = GetIdGenre(cbxGenres.Text);
        string genre = txbGenre.Text;
        string idPublic = GetIdPublic(cbxPublics.Text);
        string lePublic = txbPublic.Text;
        string idRayon = GetIdRayon(cbxRayons.Text);
        string rayon = txbRayon.Text;
        if (revueModif != null)...
        Revue revueValorise = new Revue(id, titre, image, idGenre, genre = null, idPublic, lePublic = null, idRayon, rayon = null, periodicite, delaiMiseADispo);

        return revueValorise;
    }
    catch (Exception ex) { }
    return null;
}

```

## CONTROLLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access. Comme elle attend un type générique elle peut traiter une revue.

```
//Cette méthode envoi à l acces livre, dvd et Revue.  
//En C#, T est un paramètre de type générique.  
//Ce qui implique quelle va traiter indiféremment un objet Livre/Dvd/revue  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool EnvoiDocuments<T>(T unDocument)  
{  
    return access.CreerDocument(unDocument);  
}
```

## DAL

Dans le fichier [Access.cs](#) la méthode `CreerDocument<T>()` est appelée. Cette méthode va convertir au format JSON la revue envoyée en paramètre. Le nom du type d'objet en minuscule ici revue correspond au nom de la table à sélectionner pour l'API, même fonctionnement comme pour un livre et un dvd.

```
//Envoi l'objet en paramètre à l'API  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool CreerDocument<T>(T unDocument)  
{  
    //Convertit en json l'objet en paramètre  
    String jsonCreerDocument = JsonConvert.SerializeObject(unDocument);  
    try  
    {  
        //Appel TraitementRecup avec en paramètre POST, le nom du type d'objet et le json  
        List<Object> liste = TraitementRecup<Object>(POST, typeof(T).Name.ToLower() + "/" + jsonCreerDocument.Replace(" ", "-"));  
        return (liste != null);  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
    }  
    return false;  
}
```

## API

Au sein de l'API, [Mediatekdocuments.php](#) est le point d'entrée. Il récupère les paramètres et en fonction du verbe HTTP utilisé ici POST appelle la méthode concernée de la classe Controle.

```
// traitement suivant le verbe HTTP utilisé  
if ($_SERVER['REQUEST_METHOD'] === 'GET') {  
    $controle->get($table, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    $controle->post($table, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {  
    $controle->put($table, $id, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {  
    $controle->delete($table, $champs);  
}
```



[Controle.php](#) reçoit revue pour table et va donc appeler la méthode [ajoutRevue\(\)](#) avec en paramètre le contenu du JSON envoyé depuis l'application.

```
public function post($table, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->ajoutLivre($champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->ajoutDvd($champs);
    } elseif ($table == "revue") {
        $result = $this->accessBDD->ajoutRevue($champs); ←
    } elseif ($table == "commandeLivreOuDvd") {
        $result = $this->accessBDD->ajoutCommandeLivreOuDvd($champs);
    } elseif ($table == "commandeAbonnementRevue") {
        $result = $this->accessBDD->ajoutCommandeAbonnementRevue($champs);
    } else {
        $result = $this->accessBDD->insertOne($table, $champs);
    }
}
```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. Il est important de respecter l'ordre des tables pour que l'ajout fonctionne. En premier la table document et ensuite la table revue.

```
public function ajoutRevue($champs) {
    $champsDocument = [
        "id" => $champs["Id"],
        "titre" => $champs["Titre"],
        "image" => $champs["Image"],
        "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"],
        "idGenre" => $champs["IdGenre"]
    ];

    $champsRevue = [
        "id" => $champs["Id"],
        "periodicite" => $champs["Periodicite"],
        "delaiMiseADispo" => $champs["DelaiMiseADispo"]
    ];

    $result = $this->insertOne("document", $champsDocument);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->insertOne("revue", $champsRevue);
}
```

## Modifier une revue

### VIEW

Le cheminement pour la modification d'une revue est similaire. Depuis [FrmMediatek.cs](#) il suffit de sélectionner une revue et de cliquer sur le bouton modifier. Son rôle est de lancer [FrmAjout.cs](#) avec divers paramètres qui vont affecter la seconde vue.

```

private void btnModifierRevue_Click(object sender, EventArgs e)
{
    if (dgvRevuesListe.CurrentCell != null)
    {
        //Chaîne de caractère qui va influencer FrmAjout
        string ongletRevue = "revue";
        //Converti la liste de revues en liste d'objets
        List<Object> listeObjRevue = lesRevues.ConvertAll(Revue => (Object)Revue);
        //Lance frmAjout avec les paramètres nécessaires à l'ajout d'une revue
        Object aModifier = (Object)bdgRevuesListe.List[bdgRevuesListe.Position];
        FrmAjout frmAjout = new FrmAjout(bdgGenres, bdgPublics, bdgRayons, true, ongletRevue, listeObjRevue, aModifier);
        frmAjout.Text = "Modification d'une REVUE";
        frmAjout.Show();
        this.Hide();
    }
}

```

Depuis [FrmAjout.cs](#) l'affichage et les diverses fonctions vont s'adapter.

Une fois modifié il suffit de cliquer sur le bouton modifier pour lancer la modification.

```

private void btnModifier_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Modifier ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        switch (quelEnvoi)
        {
            case "livre":
                Livre livre = SuperLivre();
                if (livre != null)[...]
                break;
            case "dvd":
                Dvd dvd = SuperDvd();
                if (dvd != null)
                {
                    if (controller.ModifierDocuments(dvd))[...]
                }
                break;
            case "revue": 
                Revue revue = SuperRevue();
                if (revue != null)
                {
                    if (controller.ModifierDocuments(revue))
                    {
                        MessageBox.Show("Revue modifiée");
                    }
                }
                break;
        }
    }
}

```

[SuperRevue\(\)](#) va se comporter différemment comme c'est une modification, il faut autoriser l'absence de changement des comboBox et que l'id de la revue qui existe. Ainsi que la valorisation d'une revue doit aussi prendre en compte ces cas.

Une fois que la revue est correctement valorisé elle est envoyée au contrôleur.

```

private Revue SuperRevue()
{
    try
    {
        Revue revue = ValoriseRevue();
        //Contrôle les comboBox sauf si modification
        if (cbxGenres.Text != "" && cbxPublics.Text != "" && cbxRayons.Text != "" || revueModif != null)
        {
            //Contrôle les textBox avant appel du constructeur
            if (txbTitre.Text != "" && txbAuteurRealisateurPer.Text != "" && txbCollectionSynopsisDel.Text != "")
            {
                if ((int.Parse(txbCollectionSynopsisDel.Text) > 1) && (int.Parse(txbCollectionSynopsisDel.Text) < 1000))
                {
                    if ((int.Parse(txbNumero.Text)) > 10000 && (int.Parse(txbNumero.Text)) < 20000)
                    {
                        //Contrôle que l'id n'est pas déjà utilisé avant appel du constructeur
                        Revue testId = listeRevue.Find(x => x.Id.Equals(txbNumero.Text));
                        if (testId == null || revueModif != null)
                        {
                            return revue;
                        }
                    }
                }
            }
        }
    }
}

```

## CONTROLLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access. Comme elle attend un type générique elle peut traiter une revue

```

//Modifie un livre, un dvd ou une revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocuments<T>(T unDocument)
{
    return access.ModifierDocument(unDocument);
}

```

## DAL

Pour la modification d'une revue, il faut changer le verbe envoyé, cela devient PUT et l'API attend aussi l'id de la revue à envoyer.

```

public bool ModifierDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonModifierDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de modification
        string id = "";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
            case Dvd dvd:
                id = dvd.Id;
                break;
            case Revue revue:
                id = revue.Id;
                break;
        }
        List<Object> liste = TraitementRecup<Object>(PUT, typeof(T).Name.ToLower() + "/" + id + "/" + jsonModifierDocument.Replace(" ", "-"));
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
//Supprime le document en paramètre

```

## API

Au sein de l'API, [Mediatekdocuments.php](#), récupère les paramètres et en fonction du verbe HTTP utilisé ici PUT appelle la méthode concernée de la classe Contrôle.

```
// traitement suivant le verbe HTTP utilisé
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $controle->get($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controle->post($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') { ←
    $controle->put($table, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controle->delete($table, $champs);
}
```

Le contrôleur va sélectionner la fonction `put()` et comme c'est un dvd il va appeler la fonction `modifRevue()`

```
public function put($table, $id, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->modifiLivre($id, $champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->modifiDvd($id, $champs);
    } elseif ($table == "revue") { ←
        $result = $this->accessBDD->modifiRevue($id, $champs);
    } elseif ($table == "commandeLivreOuDvd") {
```

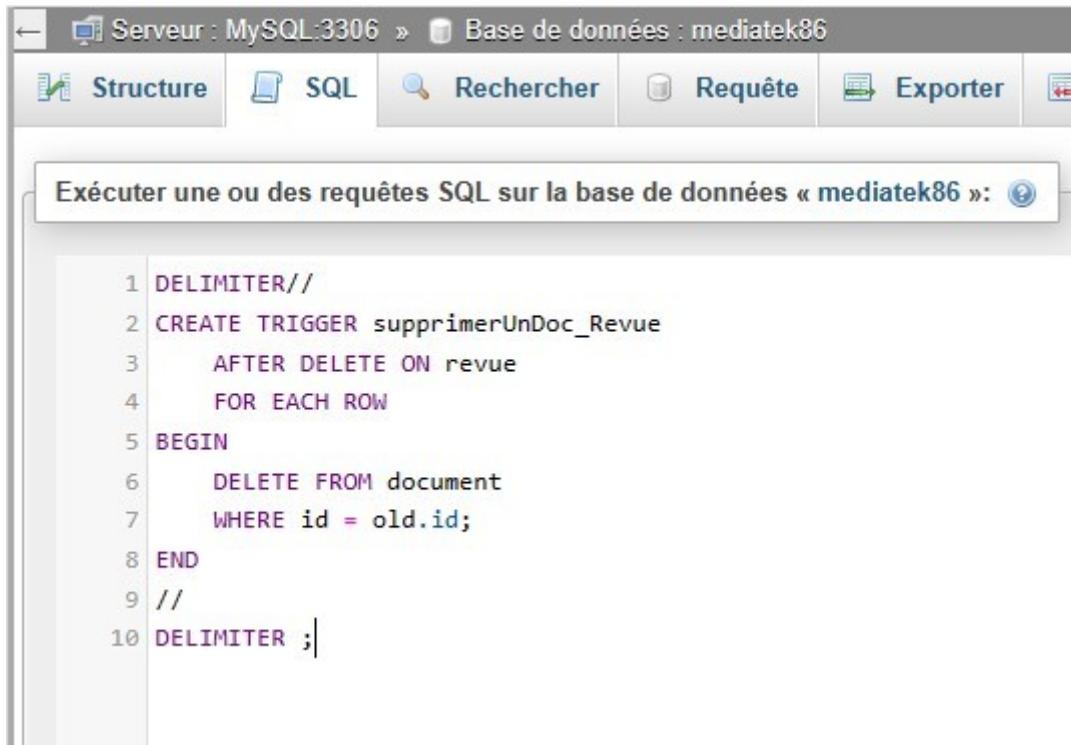
Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. Et on ne met pas de paramètre concernant l'id du document car il est demandé de ne pas pouvoir le changer.

```
public function modifiRevue($id, $champs) {
    $champsDocument = [
        "titre" => $champs["Titre"],
        "image" => $champs["Image"],
        "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"],
        "idGenre" => $champs["IdGenre"]
    ];

    $champsRevue = [
        "periodicite" => $champs["Periodicite"],
        "delaiMiseADispo" => $champs["DelaiMiseADispo"]
    ];
    $result = $this->updateOne("document", $id, $champsDocument);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->updateOne("revue", $id, $champsRevue);
}
```

## Supprimer une revue

Pour la suppression d'une revue il faut créer un trigger comme pour les précédents. Le fait de supprimer un tuple dans la table revue, cela doit supprimer le tuple associé de la table document.



The screenshot shows the MySQL Workbench interface. The title bar reads "Serveur : MySQL:3306 » Base de données : mediatek86". The tabs at the top are "Structure", "SQL" (which is selected), "Rechercher", "Requête", "Exporter", and "Historique". Below the tabs, a panel titled "Exécuter une ou des requêtes SQL sur la base de données « mediatek86 »:" contains the following SQL code:

```
1 DELIMITER//  
2 CREATE TRIGGER supprimerUnDoc_Revue  
3     AFTER DELETE ON revue  
4     FOR EACH ROW  
5 BEGIN  
6     DELETE FROM document  
7     WHERE id = old.id;  
8 END  
9 //  
10 DELIMITER ;|
```

De plus il faut prendre en compte si une commande et un exemplaire est rattaché. j'ai donc réalisé cette tâche à l'issue de la mission 3 car les commandes et exemplaires ne sont pas encore traités. Contrairement aux suppression de livre et dvd, il n'est pas permis de faire un simple contrôle sur une DataGridView car la gestion des exemplaire est sur un autre onglet. Donc il faut chercher dans des listes de commande et d exemplaire si l'id de la revue est présente

## VIEW

Je code l'action du bouton Supprimer de FrmMediatek.cs.

```
private void btnSupprimerRevue_Click(object sender, EventArgs e)  
{  
    //vérification de la sélection  
    if (dgvRevuesListe.CurrentCell != null)  
    {  
        //contrôle si un abonnement (donc une commande) est en cours  
        Revue revue = (Revue)bdgRevuesListe.List[bdgRevuesListe.Position];  
        Abonnement testId = listCmdRevues.Find(x => x.IdRevue.Equals(revue.Id));  
        if (testId == null)  
        {  
            //contrôle si des exemplaires sont rattaché à la revue.  
            lesExemplaires = controller.GetExemplairesRevue(revue.Id);  
            Exemplaire exemplaire = lesExemplaires.Find(x => x.Id.Equals(revue.Id));  
            if (exemplaire == null)  
            {  
                if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)  
                {  
                    controller.SupprimerDocument(revue);  
                    tabRevues_Enter(null, null);  
                }  
            }  
            else { MessageBox.Show("Impossible de supprimer une revue avec des exemplaires !"); }  
        }  
        else { MessageBox.Show("Impossible de supprimer une revue avec un abonnement en cours !"); }  
    }  
    else{MessageBox.Show("selectionner une Revue !");}  
}
```

## CONTROLLER

Le contrôleur traite le livre comme un objet de type générique et appelle l'accès.

```
//Supprime un livre, dvd ou revue  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool SupprimerDocument<T>(T unDocument)  
{  
    return access.SupprimerDocument(unDocument);  
}
```

## DAL

C'est identique aux précédentes suppressions sauf que l'on traite une revue.

```
public bool SupprimerDocument<T>(T unDocument)  
{  
    //Convertit en json l'objet en paramètre  
    String jsonSupprimerDocument = JsonConvert.SerializeObject(unDocument);  
    try  
    {  
        //L'id est nécessaire pour l'API en cas de modification  
        string id = "";  
        switch (unDocument)  
        {  
            case Livre livre:  
                id = livre.Id;  
                break;  
            case Dvd dvd:  
                id = dvd.Id;  
                break;  
            case Revue revue:  
                id = revue.Id; <---- Orange arrow  
                break;  
        }  
        List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + "Id\":" + id + "});  
        return (liste != null);  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
    }  
    return false;  
}
```

## API

Dans l'API on suit le même cheminement que précédemment. [Mediatekdocuments.php](#) Récupère les paramètres et en fonction du verbe HTTP utilisé ici DELETE appelle la méthode concernée de la classe Control

```
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {  
    $controler->delete($table, $champs);  
}
```

[Contrôle.php](#) va donc appeler la méthode `delete()` avec en paramètre la table ici revue et l'id de la revue sélectionné. Cette fonction va appeler `delete()` de [AccesBDD.php](#) pour construire la requête SQL.

```

public function delete($table, $champs) {
    $result = $this->accessBDD->delete($table, $champs);
    if ($result == null || $result == false) {
        $this->reponse(400, "requete invalide");
    } else {
        $this->reponse(200, "OK");
    }
}

```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. Ces fonctions sont fournies avec l'API alors je ne les détaille pas.

Une fois la suppression des revues fait, cela termine la mission 1. Je mets à jour mon kaban et passe à la mission 2.

projet kanban atelier 3		
<a href="#">View</a>	<a href="#">New view</a>	
<a href="#">Filter by keyword or by field</a>		
<p><span style="color: red;">●</span> To do 12</p> <p>This is ready to be picked up</p> <ul style="list-style-type: none"> <li><span style="color: gray;">●</span> Draft Mission 2 : gérer les commandes Tâche 2 : gérer les commandes de revues</li> <li><span style="color: gray;">●</span> Draft Mission 3 : gérer le suivi de l'état des exemplaires tâche</li> <li><span style="color: gray;">●</span> Draft Mission 4 : mettre en place des authentifications</li> <li><span style="color: gray;">●</span> Draft Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 1 : corriger des problèmes de sécurité</li> </ul>	<p><span style="color: orange;">●</span> In progress 1</p> <p>This is actively being worked on</p> <ul style="list-style-type: none"> <li><span style="color: gray;">●</span> Draft Mission 2 : gérer les commandes Tâche 1 : gérer les commandes de livres ou de DVD</li> </ul>	<p><span style="color: green;">●</span> Done 2</p> <p>This has been completed</p> <ul style="list-style-type: none"> <li><span style="color: gray;">●</span> Draft Mission 1 : gérer les documents tâche1 Dans les onglets actuels (Livres, Dvd, Revues), ajouter les fonctionnalités (boutons) qui permettent d'ajouter, de modifier ou de supprimer un document.</li> <li><span style="color: gray;">●</span> Draft Mission 1 : gérer les documents tâche2 Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Document, idem pour LivresDvd.</li> </ul>

## Mission 2 : gérer les commandes

### Tâche 1 : gérer les commandes de livres ou de DVD

Tâches à réaliser

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.

- Créer un onglet (ou une nouvelle fenêtre) pour gérer les commandes de livres.
- La charte graphique doit correspondre à l'existant.
- Dans toutes les listes, permettre le tri sur les colonnes.
- Dans l'onglet (ou la fenêtre), 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.

— 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".

— 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).

— Permettre de supprimer une commande uniquement si elle n'est pas encore livrée. Faire un trigger qui réalise aussi la suppression dans la classe Ile.

— 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.

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

— Toutes les sécurités seront mises en place pour éviter des erreurs de manipulation.

— Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Commande

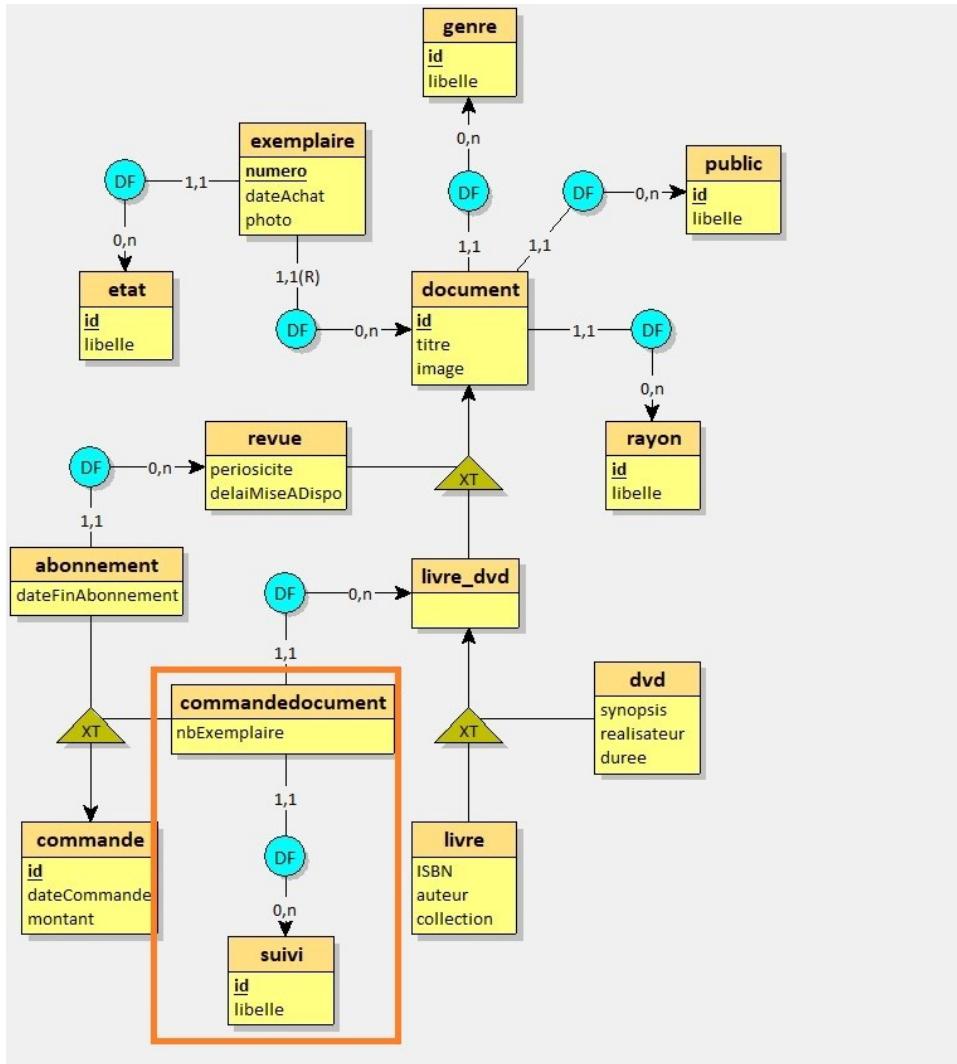
Diagramme de cas d'utilisation



## Création de la table suivi

Selon le cahier des charges, les différentes étapes de suivi d'une commande pour un livre ou DVD sont en cours, relancée, livrée et réglée. Une commande de revue correspond à un abonnement. Je construis un MCD depuis les informations données pour la réalisation.

MCD



Je crée la table suivie dans la base de données avec les champs libelle et id.

Je fait attention à avoir le même interclassement que les précédentes tables sinon problèmes de compatibilité.

```

CREATE TABLE suivie(
    id VARCHAR(50),
    libelle VARCHAR(50),
    PRIMARY KEY(id)
);

```

J'ajoute la colonne idSivi dans la table commandeedocument. Et je fais le lien entre les deux tables via leur id.

```

1 ALTER TABLE commandeedocument
2 ADD idSuivi VARCHAR(50);
3 ALTER TABLE commandeedocument
4 ADD FOREIGN KEY (idSuivi) REFERENCES suivi(id);

```

Et je remplis la table suivit avec les différentes étapes de suivit. L'id 0005 pour réglée n'est pas anodin, cela va me servir pour faire des tests qui ne peuvent se faire si c'est 0004.

Exécuter une ou des requêtes SQL sur la table « mediatek86.suivi »:

```
1 INSERT INTO suivi (id,libelle) VALUES('0001','enCours'),('0002','relance'),('0003','livré'),('0005','réglée');
```

Le trigger qui contrôle la contrainte de partition de l'héritage sur Commande. Avant d'insérer un tuple dans la table commandeedocument, il va simplement parcourir la table abonnement et chercher si ils ont le même id. Si c'est le cas il va empêcher l'insertion.

```

DELIMITER //
CREATE TRIGGER commandeedocumentOuAbonnement
    BEFORE INSERT ON commandeedocument
    FOR EACH ROW
BEGIN
    DECLARE nb INTEGER ;
    SELECT COUNT(*) INTO nb
        FROM abonnement
        WHERE id = NEW.id ;
    IF (nb = 1) THEN
        SIGNAL SQLSTATE "45000"
        SET MESSAGE_TEXT = "opération impossible";
    END IF ;
END
//
DELIMITER ;

```

## Classes nécessaire aux commandes

Après avoir ajouté la table suivit, je crée la classe **Suivi** dans le modèle. Elle hériter de **Categorie** fourni avec l'application cela me permet de l'utiliser comme les classes **Genre**, **Public** et **Rayon**.

```

namespace MediaTekDocuments.model
{
    3 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    class Suivi : Categorie
    {
        0 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
        public Suivi(string id, string libelle) : base(id, libelle)
        {
        }
    }
}

```

Je crée aussi la classe **Commande** en respectant les noms et type de données du MCD et de la BDD.

```

5 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
public class Commande
{
    15 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public string Id { get; }
    7 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public DateTime DateCommande { get; }
    6 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public double Montant { get; } //type double imposé par la BDD

    2 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public Commande(string id, DateTime dateCommande, double montant)
    {
        Id = id;
        DateCommande = dateCommande;
        Montant = montant;
    }
}

```

Et la classe CommandeDocument qui hérite de Commande comme dans le MCD.

```

5 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
public class CommandeDocument : Commande
{
    5 références | hedi-k, il y a 8 jours | 1 auteur, 1 modification
    public int NbExemplaire { get; }
    9 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public string IdSuivi { get; }
    3 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public string Suivi { get; }
    7 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public string IdLivreDvd { get; }

    2 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
    public CommandeDocument(string id, DateTime dateCommande, double montant, int nbExemplaire, string idSuivi, string suivi, string idLivreDvd )
        : base(id, dateCommande, montant)
    {
        NbExemplaire = nbExemplaire;
        IdSuivi = idSuivi;
        Suivi = suivi;
        IdLivreDvd = idLivreDvd;
    }
}

```

Donc créer une commande c'est créer un objet de type CommandeDocument.

## Charte graphique

Il est demandé que la charte graphique correspond à l'existant, je fais le choix d'ajouter des onglets pour la gestion des commandes. Cela donne.

DateCommande	NbExemplaire	Suivi	n° de document	n° de commande	Montant
29/02/2024	2	livrée	20050	1006	2
29/02/2024	2	livrée	20050	1007	1
29/02/2024	5	en cours	20050	1010	5
29/02/2024	10	en cours	20051	1011	10
27/02/2024	4	livrée	20002	1005	4
22/02/2024	2	en cours	20001	0060	2
05/02/2024	3	livrée	20003	1002	3

## Commande de Livres

### Recherche d'un livre

Avant d'ajouter modifier ou supprimer une commande, il faut pouvoir sélectionner un livre pour pouvoir gérer ses commandes .

Pour se faire je recherche l'id d'un document en parcourant la liste concerné. (livre, dvd ou revue)

```
//Action du bouton recherche d'un livre sur un ID
1 référence | hedi-k, il y a 6 jours | 1 auteur, 2 modifications
private void btnRechCmdLivre_Click(object sender, EventArgs e)
{
    //Crée un livre si il a l'id qui est entré dans la recherche
    Livre livre = lesLivres.Find(x => x.Id.Equals(txtCmdLivreNum.Text));
    if (livre != null)
    {
        AfficheCommandeLivreInfos(livre);
    }
    else
    {
        MessageBox.Show("numéro introuvable");
    }
}
```

Si dans le cas d'une recherche de livre, on trouve un id qui correspond, on construit l'objet livre qui correspond à l'id de la listeLivres. (cette liste est chargé au lancement de l'application)  
Et on appelle la fonction `AfficheCommandeLivreInfo()` avec ce livre en paramètre.

```

//Affiche les informations du livre dans commande livre.
2 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
private void AfficheCommandeLivreInfos(Livre livre)
{
    txbCmdLivreTitre.Text = livre.Titre;
    txbCmdLivreAuteur.Text = livre.Auteur;
    txbCmdLivreCollection.Text = livre.Collection;
    txbCmdLivreGenre.Text = livre.Genre;
    txbCmdLivrePublic.Text = livre.Public;
    txbCmdLivreRayon.Text = livre.Rayon;
    txbCmdLivreISBN.Text = livre.Isbn;
    txbCmdLivreId.Text = livre.Id;
    string image = livre.Image;
    try
    {
        pcbCmdLivreImage.Image = Image.FromFile(image);
    }
    catch
    {
        pcbCmdLivreImage.Image = null;
    }
}

```

Cela va remplir toutes les textBox et image du groupBox informations détaillées avec les données du livre. Exemple :

Informations détaillées	
Numéro de document :	00017
	Code ISBN :
Titre :	Catastrophes au Brésil
Auteur(e) :	Philippe Masson
Collection :	
Genre :	Policier
Public :	Ados
Rayon :	Jeunesse romans

## Ajouter une commande de livre

### VIEW

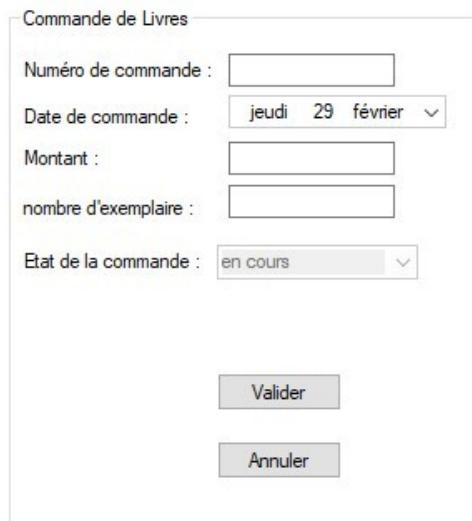
Une fois un livre recherché, j'ai la textBox titre qui est valorisée. J'utilise cette textBox pour savoir si un livre est sélectionné. Donc le clique sur le bouton Ajouter, va autoriser l'accès au groupBox Commande de livre qui permet de commander un livre. Je bloque l'accès à la comboBox qui modifie l'état de la commande car il est imposé qu'une commande enregistré est un état en cours.

```

//Action du btn ajouter une commande de livre
1 référence | hedi-k, il y a 7 jours | 1 auteur, 2 modifications
private void btnAjouterCmdLivre_Click(object sender, EventArgs e)
{
    //Teste sur le titre, un livre à toujours un titre.
    if (txbCmdLivreTitre.Text != "")
    {
        grbCmdLivre2.Enabled = true;
        txbCmdLivreNumCmd.Enabled = true;
        cbxSuivi.SelectedIndex = 0;
        cbxSuivi.Enabled = false;
    }
}

```

La grpBox étant autorisé, je peux la modifier pour commander un livre.



J'entre les informations pour une commande et je sélectionne valider.

```
1 référence | hedi-k, il y a 4 jours | 1 auteur, 5 modifications
private void btnCmdLivreValider_Click(object sender, EventArgs e)
{
    CommandeDocument cmdLivre = SuperCmdLivre();
    //Si la commande est valorisé ou que l'on ne modifie pas une commande
    if (cmdLivre != null && cmdLivreOuDvdModif == false)
    {
        //Envoyer la commande au contrôleur.
        if (controller.EnvoiDocuments(cmdLivre))
        {
            MessageBox.Show("Livre commandé.");
            VideCmdLivre();
            tabOngletCommandeLivre_Enter(null, null);
        }
    }
}
```

`SuperCmdLivre()` va contrôler le contenu des entrés. J'impose volontairement des numéros de commande compris entre 0 et 1000 pour les livres, cela garde une cohérence dans les numéros affichés dans la base de données. Pour le montant et les exemplaires le fait de m'assurer que l'entrée soit supérieur à 0 est logique mais surtout cela élimine les lettres.

```
private CommandeDocument SuperCmdLivre()
{
    try
    {
        //Valorise la commande.
        CommandeDocument cmdValorise = ValoriseCommandeLivre();
        //Contrôle le numéro de la commande.
        if (int.Parse(txbCmdLivreNumCmd.Text) > 0 && int.Parse(txbCmdLivreNumCmd.Text) < 1000)
        {
            //Contrôle le montant de la commande.
            if (double.Parse(txbCmdLivreMontantCmd.Text) > 0)
            {
                //Contrôle le nombre d'exemplaire de la commande.
                if (int.Parse(txbCmdLivreNbExCmd.Text) > 0)
                {
                    if (cmdValorise.IdSuivi != null)
                    {
                        //Contrôle que l'id de la commande soit disponible.
                        CommandeDocument testId = listCmdLivres.Find(x => x.Id.Equals(cmdValorise.Id));
                        if (testId == null || cmdLivreOuDvdModif == true)
                        {
                            return cmdValorise;
                        }
                        else { MessageBox.Show("Numéro de commande déjà utilisé"); return null; }
                    }
                }
            }
        }
    }
}
```

`ValoriseCommandeLivre()` crée la commande. La seul contrainte est de bien prendre en compte le type de valeur attendu.

```
private CommandeDocument ValoriseCommandeLivre()
{
    try
    {
        string id = FormaterId(txBCmdLivreNumCmd.Text);
        DateTime dateCommande = dtpCmdLivreDateCmd.Value;
        double montant = double.Parse(txBCmdLivreMontantCmd.Text);
        int nbExemplaire = int.Parse(txBCmdLivreNbExCmd.Text);
        string idSuivi = GetIdSuivi(cbxSuivi.Text);
        string suivi = null;
        String idLivreDvd = txBCmdLivreId.Text;

        CommandeDocument commandeValorise = new CommandeDocument(id, dateCommande, montant, nbExemplaire, idSuivi, suivi, idLivreDvd);
        return commandeValorise;
    }
    catch (Exception ex) { return null; }
}
```

`GetIdSuivi()` me permet de retourner l'id du libelle en fonction du libelle comme pour les genres publics et rayons c'est pour cela que je l'ai fais hériter de `Categorie`.

```
//retourne l'id de suivi selectionné
2 références | hedi-k, il y a 7 jours | 1 auteur, 2 modifications
private string GetIdSuivi(string unSuivi)
{
    List<Categorie> uneListe = controller.GetSuivi();
    foreach (Categorie uneCategorie in uneListe)
    {
        if (uneCategorie.Libelle == unSuivi)
        {
            return uneCategorie.Id;
        }
    }
    return null;
}
```

`FormaterId()` me rend juste l'id écrit avec 4 chiffres pour que cela soit plus jolie dans la base de donnée, 10 devient 0010.

```
//retourne l'id au format 4 digits 1 =>0001
3 références | hedi-k, il y a 10 jours | 1 auteur, 1 modification
private string FormaterId(string id)
{
    int idFormaté = int.Parse(id);
    return idFormaté.ToString("D4");
}
```

Une fois la commande valorisé elle est envoyé au contrôleur.

## CONTROLLER

C'est toujours le même qui est appelé, il est capable de gérer un objet `CommandeDocument` comme un type générique.

```

//Cette méthode envoi à l'accès livre, dvd et Revue.
//En C#, T est un paramètre de type générique.
//Ce qui implique quelle va traiter indifféremment un objet Livre/Dvd/revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool EnvoiDocuments<T>(T unDocument)
{
    return access.CreerDocument(unDocument);
}

```

## DAL

Comme pour les précédant, cependant cette fois CommandeDocument ne correspond pas au nom de table visé alors il faudra adapter l'API.

```

//Envoi l'objet en paramètre à l'API
1 référence | 0 modification | 0 auteur, 0 modification
public bool CreerDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonCreerDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //Appel TraitementRecup avec en paramètre POST, le nom du type d'objet et le json
        List<Object> liste = TraitementRecup<Object>(POST, typeof(T).Name.ToLower() + "/" + jsonCreerDocument.Replace(" ", "-"));
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

```

## API

Au sein de l'API, [Mediatekdocuments.php](#), il récupère les paramètres et en fonction du verbe HTTP utilisé ici POST appelle la méthode concerné de la classe Contrôle.

```

// traitement suivant le verbe HTTP utilisé
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $contrôle->get($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $contrôle->post($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
    $contrôle->put($table, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $contrôle->delete($table, $champs);
}
----- ;
} elseif ($table == "commandedocument") {
    $result = $this->accessBDD->ajoutCommandeLivreOuDvd($champs);
}

```

[Contrôle.php](#) reçoit commandedocument pour table et va donc appeler la méthode ajoutCommandeLivreOuDvd() avec en paramètre le contenu du JSON envoyé depuis l'application.

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL en fonction des paramètres envoyés. C'est ici que l'on sélectionne les tables à modifier. Attention ici l'id correspond à l'id de la commande et pas à celui du document. L'id du livre devient idLivreDvd. On voit aussi ici que l'on valorise d'un idSuivi de la table créée en début de mission.

```

public function ajoutCommandeLivreOuDvd($champs) {
    $champsCommande = [
        "id" => $champs["Id"],
        "dateCommande" => $champs["DateCommande"],
        "montant" => $champs["Montant"]
    ];

    $champsCommandeDocument = [
        "id" => $champs["Id"],
        "nbExemplaire" => $champs["NbExemplaire"],
        "idLivreDvd" => $champs["IdLivreDvd"],
        "idSuivi" => $champs["IdSuivi"]
    ];

    $result = $this->insertOne("commande", $champsCommande);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->insertOne("commandedocument", $champsCommandeDocument);
}

```

La fonction `insertOne()` elle est fournie avec l'API je ne la détail pas, elle doit juste me retourner un true si tout a fonctionné.

Ce booléen qui revient jusqu'à la vue et m'actualise l'affichage.

```

if (controller.EnvoiDocuments(cmdLivre))
{
    MessageBox.Show("Livre commandé.");
    VideCmdLivre();
    tabOngletCommandeLivre_Enter(null, null);
}

```

`VideCmdLivre()` nettoie les textBox.

```

//Vide les txb de commandelivre
2 références | hedi-k, il y a 7 jours | 1 auteur, 1 modification
private void VideCmdLivre()
{
    grbCmdLivre2.Enabled = false;
    txbCmdLivreTitre.Text = "";
    txbCmdLivreAuteur.Text = "";
    txbCmdLivreCollection.Text = "";
    txbCmdLivrePublic.Text = "";
    txbCmdLivreRayon.Text = "";
    txbCmdLivreISBN.Text = "";
    txbCmdLivreId.Text = "";
    txbCmdLivreNumCmd.Text = "";
    txbCmdLivreMontantCmd.Text = "";
    txbCmdLivreNbExCmd.Text = "";
    cbxSuivi.SelectedIndex = -1;
    dtpCmdLivreDateCmd.Value = DateTime.Today;
}

```

## DataGridView des commande de livre

Avant de détailler la modification d'un livre il faut expliquer le fonctionnement de la DataGridView qui affiche les commandes car la modification l'utilise.

	DateCommande	NbExemplaire	Suivi	n° de document	n° de commande	Montant
▶	29/02/2024	2	en cours	00102	0107	2
	28/02/2024	5	livrée	00100	0105	2
	28/02/2024	2	en cours	00102	0106	2

Des que l'on sélectionne l'onglet commande livre, la liste des commandes est chargée. Puis la fonction `ChargeDgvCmdLivre()` avec en paramètre cette liste de commande.

```
private void tabOngletCommandeLivre_Enter(object sender, EventArgs e)
{
    //Charge la liste de livre commandé
    listCmdLivres = controller.GetAllCommandeLivres();
    //Remplit la DGV
    ChargeDgvCmdLivres(listCmdLivres);
    //Charge la cbx suivi
    RemplirComboCategorie(controller.GetSuivi(), bdgSuivi, cbxSuivi);
    grbCmdLivre2.Enabled = false;
    grbCmdLivre2.Enabled = false;
    indiceSuivi = -2;
    cmdLivreOuDvdModif = false;
}
```

La fonction va actualiser la DataGridView avec le contenu de la liste.

```
//Methode pour remplir la liste des commandes de livre
2 références | hedi-k, il y a 7 jours | 1 auteur, 1 modification
private void ChargeDgvCmdLivres(List<CommandeDocument> cmdLivre)
{
    //transfère de la liste à la bidingsource
    bdgCommandeLivresListe.DataSource = cmdLivre;
    //transfère de la bindingsource à la data grid view
    dgvCmdLivre.DataSource = bdgCommandeLivresListe;
    //gestions des tailles de colonnes
    dgvCmdLivre.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
    //Choix des colonnes affichés et de leur noms.
    dgvCmdLivre.Columns["IdSuivi"].Visible = false;
    dgvCmdLivre.Columns["DateCommande"].DisplayIndex = 0;
    dgvCmdLivre.Columns[3].HeaderText = "n° de document";
    dgvCmdLivre.Columns[4].HeaderText = "n° de commande";
}
```

Il est imposé un tri sur les colonnes dans les consignes.

```

//Trie de la grid view
1 référence | hedi-k, il y a 6 jours | 1 auteur, 3 modifications
private void dtgCmdLivre_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)
{
    List<CommandeDocument> sortedList = new List<CommandeDocument>();
    string titreColonne = dgvCmdLivre.Columns[e.ColumnIndex].HeaderText;
    switch (titreColonne)
    {
        case "DateCommande":
            sortedList = listCmdLivres.OrderBy(o => o.DateCommande).ToList();
            break;
        case "NbExemplaire":
            sortedList = listCmdLivres.OrderBy(o => o.NbExemplaire).ToList();
            break;
        case "Suivi":
            sortedList = listCmdLivres.OrderBy(o => o.Suivi).ToList();
            break;
        case "n° de document":
            sortedList = listCmdLivres.OrderBy(o => o.IdLivreDvd).ToList();
            break;
        case "n° de commande":
            sortedList = listCmdLivres.OrderBy(o => o.Id).ToList();
            break;
        case "Montant":
            sortedList = listCmdLivres.OrderBy(o => o.Montant).ToList();
            break;
    }
    ChargeDgvCmdLivres(sortedList);
}

```

## Modifier une commande de livre

### VIEW

Après sélection d'un livre sur la DataGridView, l'action du bouton modifier, une commande est valorisé depuis la sélection dans la dataGridView.

```

//Action du btn modifier dans commande de livre
1 référence | hedi-k, il y a 6 jours | 1 auteur, 3 modifications
private void btnModifierCmdLivre_Click(object sender, EventArgs e)
{
    //contrôle si une ligne est sélectionnée.
    if (dgvCmdLivre.CurrentCell != null)
    {
        try
        {
            //booléen qui va influencer le contrôleur choisi.
            cmdLivreOuDvdModif = true;
            CommandeDocument cmd = (CommandeDocument)bdgCommandeLivresListe.List[bdgCommandeLivresListe.Position];
            //Utilise cet indice pour valoriser le suivi d'une commande
            indiceSuivi = int.Parse(cmd.IdSuivi);
            //Gestion de l'affichage
            grbCmdLivre2.Enabled = cmdLivreOuDvdModif;
            cbxSuivi.Enabled = cmdLivreOuDvdModif;
            txbCmdLivreNumCmd.Enabled = !cmdLivreOuDvdModif;
            txbCmdLivreNumCmd.Text = cmd.Id;
            txbCmdLivreMontantCmd.Text = cmd.Montant.ToString();
            txbCmdLivreNbExCmd.Text = cmd.NbExemplaire.ToString();
            dtpCmdLivreDateCmd.Value = cmd.DateCommande;
        }
        catch
        {
        }
    }
    else { MessageBox.Show("Selectionner une commande"); }
}

```

Ensute il suffit de modifier la commande et de sélectionner valider pour appeler le contrôleur. Comme la variable cmdLivreOuDvdModif est à true c'est le contrôleur lié à la modification qui est sollicité.

```
//Action du btn valider, il envoi la cmd de livre à la bdd
1 référence | hedi-k, il y a 4 jours | 1 auteur, 5 modifications
private void btnCmdLivreValider_Click(object sender, EventArgs e)
{
    CommandeDocument cmdLivre = SuperCmdLivre();
    //Si la commande est valorisé ou que l'on ne modifie pas une commande
    if (cmdLivre != null && cmdLivreOuDvdModif == false)...
    if (cmdLivre != null && cmdLivreOuDvdModif == true)
    {
        if (GestionSuivi(cmdLivre, indiceSuivi))
        {
            if (controller.ModifierDocuments(cmdLivre))
            {
                MessageBox.Show("Commande modifiée.");
                cmdLivreOuDvdModif = false;
                VideCmdLivre();
                tabOngletCommandeLivre_Enter(null, null);
            }
        }
    }
}
```

A noter la fonction [GestionSuivi\(\)](#) qui prends en paramètre la commande et l'ancien indice (quand on clique sur modifier)

Cette fonction répond aux consignes :

- une commande livrée ou réglée ne peut pas revenir à une étape précédente
- une commande ne peut pas être réglée si elle n'est pas livrée.

```
1 référence | hedi-k, il y a 4 jours | 1 auteur, 5 modifications
private bool GestionSuivi(CommandeDocument cmd, int ancienSuiv)
{
    //Contrôle que le suivit de commande est cohérent en cours (1) => relance (2) => livré (3) => réglé (5)
    //livré = 3 moins réglé = 5 résultat = 2 si autre que livré résultat supérieur à 2
    if (int.Parse(cmd.IdSuivi) >= ancienSuiv && (int.Parse(cmd.IdSuivi) - ancienSuiv < 3))
    {
        return true;
    }
    else
    {
        MessageBox.Show("erreur du suivi sélectionné");
        return false;
    }
}
```

Premier test l'idSuivi actuel doit être plus grand ou égale à l'idSuivi précédent, ce qui bloque le passage à un état précédent comme livré à en cours.

Et le second test, le fait d'empêcher que le nouvel l'idSuivi moins l'ancien l'idSuivi ne dépasse pas 3 permet d'empêcher de passer de en cours à réglé. C'est pour cela que j'ai donné 5 comme id au labelle réglé dans la table suivi.  $5 - 1 = 4$  et  $5 - 2 = 3 \Rightarrow$  livré – en cours et livré – relance sont impossible.

## CONTROLLER

Le contrôleur fonctionne toujours pareil, il accepte un type générique transmet à access.

```
//Modifie un livre, un dvd ou une revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocuments<T>(T unDocument)
{
    return access.ModifierDocument(unDocument);
}
```

## DAL

Pour la modification, il faut ajouter le cas CommandeDocument et changer le verbe envoyé, cela devient PUT. Cette fois l'id attendu n'est pas celui du document mais celui de la commande à ne pas confondre.

```
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonModifierDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de modification
        string id = "";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
            case Dvd dvd:
                id = dvd.Id;
                break;
            case Revue revue:
                id = revue.Id;
                break;
            case CommandeDocument uneCmd:
                id = uneCmd.Id; ←
                break;
        }
        List<Object> liste = TraitementRecup<Object>(PUT, typeof(T).Name.ToLower() + "/" + id + "/" + jsonModifierDocument.Replace(" ", "-"));
        return (liste != null);
    }
    catch (Exception ex)
```

## API

Ensuite dans l'API, [Mediatekdocuments.php](#)

```
// traitement suivant le verbe HTTP utilisé
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $controle->get($stable, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controle->post($stable, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') { ←
    $controle->put($stable, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controle->delete($stable, $champs);
}
```

## [Controle.php](#)

```
public function put($stable, $id, $champs) {
    if ($stable == "livre") {
        $result = $this->accessBDD->modifiLivre($id, $champs);
    } elseif ($stable == "dvd") {
        $result = $this->accessBDD->modiDvd($id, $champs);
    } elseif ($stable == "revue") {
        $result = $this->accessBDD->modifiRevue($id, $champs);
    } elseif ($stable == "commandedocument") {
        $result = $this->accessBDD->modifiCommandeLivreOuDvd($id, $champs); ←
    }
```

## AcessBDD.php

```
public function modifiCommandeLivreOuDvd($id, $champs) {
    $champsCommande = [
        "dateCommande" => $champs["DateCommande"],
        "montant" => $champs["Montant"]
    ];

    $champsCommandeDocument = [
        "nbExemplaire" => $champs["NbExemplaire"],
        "idLivreDvd" => $champs["IdLivreDvd"],
        "idSuivi" => $champs["IdSuivi"]
    ];

    $result = $this->updateOne("commande", $id, $champsCommande);
    if ($result == null || $result == false) {
        return null;
    }

    return $this->updateOne("commandedocument", $id, $champsCommandeDocument);
}
```

Le trigger qui se déclenche si une commande passe à l'étape "livrée" et crée autant de tuples dans la table "Exemplaire" que nécessaires.

```
CREATE TRIGGER CreerExemplaire
AFTER UPDATE ON commandedocument
FOR EACH ROW
BEGIN
    DECLARE laDate DATE;
    DECLARE compteurBoucle INTEGER;
    DECLARE compteurExemplaire INTEGER;
    DECLARE nbExemplaire INTEGER;

    SET compteurBoucle = 0;
    SET nbExemplaire = NEW.nbExemplaire;
    SELECT COUNT(*) INTO compteurExemplaire FROM exemplaire;

    IF (NEW.idSuivi > 2 AND NEW.idSuivi != OLD.idSuivi) THEN
        SELECT dateCommande INTO laDate FROM commande WHERE id = NEW.id;
        WHILE compteurBoucle < nbExemplaire DO
            SET compteurExemplaire = compteurExemplaire + 1;
            INSERT INTO exemplaire (`id`, `numero`, `dateAchat`, `photo`, `idEtat`)
            VALUES (NEW.idLivreDvd, compteurExemplaire, laDate, '', '00001');
            SET compteurBoucle = compteurBoucle + 1;
        END WHILE;
    END IF;
END //
```

Concrètement si une commande à un nouvel idSuivi plus grand que 2 donc 3 (livré). Une boucle va créer autant d'exemplaire que le nombre d'exemplaire de la commande. Tout les exemplaires sont identiques sauf pour leur numéro d'exemplaire qui est incrémenté de 1 à chaque tout de boucle.

## Supprimer une commande de livre

### VIEW

Pour supprimer on à pour consigne de permettre la suppression d'une commande uniquement si elle n'est pas encore livrée.

De plus il faut créer un trigger, qui va effacer le tuple correspondant de la table commande après avoir effacer le tuple de la table commandedocument.

Ce trigger fonctionne exactement comme les précédent ce n'est que les tables qui changent.

```
1 DELIMITER //
2 CREATE TRIGGER supprimerUneCommande
3     AFTER DELETE on commandedocument
4     FOR EACH ROW
5 BEGIN
6     DELETE FROM commande
7     WHERE id = OLD.id;
8 END
9 //
10 DELIMITER ;
```

Le bouton supprimer fait deux contrôles, il s'assure qu'une commande soit sélectionnée et que son idSuivi soit inférieur à 3 car cela correspond à une commande livré. Si c'est valide, la commande est transmis au contrôleur.

```
//Action qui supprime une commande
1 référence | hedi-k, il y a 6 jours | 1 auteur, 3 modifications
private void btnSupprimer_Click_1(object sender, EventArgs e)
{
    //vérification selection
    if (dgvCmdLivre.CurrentCell != null)
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            CommandeDocument uneCommande = (CommandeDocument)bdgCommandeLivresListe.List[bdgCommandeLivresListe.Position];
            //Contrôle sur le numéro d'id, 3 = livré 4 = réglé.
            if (int.Parse(uneCommande.IdSuivi) < 3)
            {
                if (controller.SupprimerDocument(uneCommande))
                {
                    MessageBox.Show("Commande supprimé.");
                    tabOngletCommandeLivre_Enter(null, null);
                }
            }
        }
    }
}
```

## CONTROLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access.

```
//Supprime un livre, dvd ou revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool SupprimerDocument<T>(T unDocument)
{
    return access.SupprimerDocument(unDocument);
}
```

## DAL

C'est la même méthode pour supprimer que les précédentes on ajoute le cas d'une CommandeDocument dans la méthode de suppression.

```
        break;
    case CommandeDocument uneCmd:
        id = uneCmd.Id;
        break;
}
List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + "Id\":" + id + "}");
return (liste != null);
}
catch (Exception ex)
```

Ensuite dans l'API on suit le même cheminement, [Mediatekdocuments.php => Controle.php => AccesBDD.php](#) jusqu'à la fonction `delete()`

## Ajouter une commande de dvd

Le fonctionnement est identique à celui d'une commande de livre.

## VIEW

Dans l'onglet dvd, on recherche un dvd dans un premier temps.

```
//Action du bouton recherche d'un dvd sur l'ID
1 référence | hedi-k, il y a 6 jours | 1 auteur, 1 modification
private void btnRechCmdDvd_Click(object sender, EventArgs e)
{
    Dvd dvd = lesDvd.Find(x => x.Id.Equals(txbCmdDvdNum.Text));
    if (dvd != null)
    {
        AfficheCommandeDvdInfos(dvd);
    }
    else{ MessageBox.Show("numéro introuvable");}
}
```

Une fois un dvd sélectionné, l'action du bouton ajouter autorise l'usage du groupBox pour l'ajout.

```

//Action du bouton Ajouter une commande de dvd
1 référence | hedi-k, il y a 4 jours | 1 auteur, 2 modifications
private void btnAjouterCmdDvd_Click(object sender, EventArgs e)
{
    if (txbCmdDvdTitre.Text != "")
    {
        grbCmdDvd2.Enabled = true;
        txbCmdDvdNumCmd.Enabled = true;
        cbxSuiviDvd.SelectedIndex = 0;
        cbxSuiviDvd.Enabled = false;
    }
}
//Action du bouton Valider une commande de dvd

```

On entre le nécessaire pour une commande de dvd et on clique sur valider.

```

//Action du bouton Valider une commande de dvd
1 référence | hedi-k, il y a 4 jours | 1 auteur, 2 modifications
private void btnCmdDvdValider_Click(object sender, EventArgs e)
{
    CommandeDocument cmdDvd = SuperCmdDvd();
    if (cmdDvd != null && cmdLivreOuDvdModif == false)
        if (controller.EnvoiDocuments(cmdDvd))
    {
        MessageBox.Show("Dvd commandé");
        VideCmdDvd();
        tabOngletCommandeDvd_Enter(null, null);
    }
}

```

SuperCmDvd() va vérifier les entrées.

```

private CommandeDocument SuperCmdDvd()
{
    try
    {
        CommandeDocument cmdValorise = ValoriseCommandeDvd();
        if (int.Parse(txbCmdDvdNumCmd.Text) > 999 && int.Parse(txbCmdDvdNumCmd.Text) < 2000)
        {
            if (double.Parse(txbCmdDvdMontantCmd.Text) > 0)
            {
                if (int.Parse(txbCmdDvdnbExCmd.Text) > 0)
                {
                    if (cmdValorise.IdSuivi != null)
                    {
                        CommandeDocument testId = listCmdDvds.Find(x => x.Id.Equals(cmdValorise.Id));
                        if (testId == null || cmdLivreOuDvdModif == true)
                        {
                            return cmdValorise;
                        }
                    }
                }
            }
        }
    }
}

```

Et ValoriseCommande() crée l'objet.

```

private CommandeDocument ValoriseCommandeDvd()
{
    try
    {
        string id = FormaterId(txbCmdDvdNumCmd.Text);
        DateTime dateCommande = dtpCmdDvdDateCmd.Value;
        double montant = double.Parse(txbCmdDvdMontantCmd.Text);
        int nbExemplaire = int.Parse(txbCmdDvdnbExCmd.Text);
        string idSuivi = GetIdSuivi(cbxSuiviDvd.Text);
        string suivi = null;
        String idLivreDvd = txbCmdDvdId.Text;

        CommandeDocument commandeValorise = new CommandeDocument(id, dateCommande, montant, nbExemplaire, idSuivi, suivi, idLivreDvd);
        return commandeValorise;
    }
    catch (Exception ex) { return null; }
}

```

## CONTROLLER

C'est toujours le même qui est appelé, il est capable de gérer un objet CommandeDocument.

```
//Cette méthode envoi à l'accès livre, dvd et Revue.  
//En C#, T est un paramètre de type générique.  
//Ce qui implique quelle va traiter indifféremment un objet Livre/Dvd/revue  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool EnvoiDocuments<T>(T unDocument)  
{  
    return access.CreerDocument(unDocument);  
}
```

## DAL

C'est toujours le même qui est appelé.

```
//Envoie l'objet en paramètre à l'API  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool CreerDocument<T>(T unDocument)  
{  
    //Convertit en json l'objet en paramètre  
    String jsonCreerDocument = JsonConvert.SerializeObject(unDocument);  
    try  
    {  
        //Appel TraitementRecup avec en paramètre POST, le nom du type d'objet et le json  
        List<Object> liste = TraitementRecup<Object>(POST, typeof(T).Name.ToLower() + "/" + jsonCreerDocument.Replace(" ", "-"));  
        return (liste != null);  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
    }  
    return false;  
}
```

## API

Le cheminement dans l'API est identique à celui pour l'ajout d'une commande de livre.

Mediatekdocuments.php => Controle.php => AccesBDD.php => ajoutCommandeLivreOuDvd()

```
public function ajoutCommandeLivreOuDvd($champs) {  
    $champsCommande = [  
        "id" => $champs["Id"],  
        "dateCommande" => $champs["DateCommande"],  
        "montant" => $champs["Montant"]  
    ];  
  
    $champsCommandeDocument = [  
        "id" => $champs["Id"],  
        "nbExemplaire" => $champs["NbExemplaire"],  
        "idLivreDvd" => $champs["IdLivreDvd"],  
        "idSuivi" => $champs["IdSuivi"]  
    ];  
  
    $result = $this->insertOne("commande", $champsCommande);  
    if ($result == null || $result == false) {  
        return null;  
    }  
    return $this->insertOne("commandedocument", $champsCommandeDocument);  
}
```

## Modifier une commande DVD

### VIEW

Après sélection d'un DVD sur la DataGridView, l'action du bouton modifier, une commande est valorisée depuis la sélection.

```
//Action du bouton modifier dans commande de dvd
1 référence | hedi-k, il y a 6 jours | 1 auteur, 1 modification
private void btnModifierCmdDvd_Click(object sender, EventArgs e)
{
    try
    {
        if (dgvCmdDvd.CurrentCell != null)
        {
            cmdLivreOuDvdModif = true;
            CommandeDocument cmd = (CommandeDocument)bdgCommandeDvdsListe.List[bdgCommandeDvdsListe.Position];
            indiceSuivi = int.Parse(cmd.IdSuivi);
            grbCmdDvd2.Enabled = cmdLivreOuDvdModif;
            cbxSuiviDvd.Enabled = cmdLivreOuDvdModif;
            txbCmdDvdNumCmd.Enabled = !cmdLivreOuDvdModif;
            txbCmdDvdNumCmd.Text = cmd.Id;
            txbCmdDvdMontantCmd.Text = cmd.Montant.ToString();
            txbCmdDvnbExCmd.Text = cmd.NbExemplaire.ToString();
            dtpCmdDvdDateCmd.Value = cmd.DateCommande;
        }
    }
    catch (Exception ex) { }
}
```

Ensuite cliquer sur bouton Valider effectue les vérifications nécessaire et envoi au contrôleur.

```
//Action du bouton Valider une commande de dvd
1 référence | hedi-k, il y a 4 jours | 1 auteur, 2 modifications
private void btnCmdDvdValider_Click(object sender, EventArgs e)
{
    CommandeDocument cmdDvd = SuperCmdDvd();
    if (cmdDvd != null && cmdLivreOuDvdModif == false)
        if (controller.EnvoiDocuments(cmdDvd))...
    if (cmdDvd != null && cmdLivreOuDvdModif == true)
    {
        if (GestionSuivi(cmdDvd, indiceSuivi))
        {
            if (controller.ModifierDocuments(cmdDvd))
            {
                MessageBox.Show("Commande modifiée.");
                VideCmdDvd();
                tabOngletCommandeDvd_Enter(null, null);
            }
        }
    }
}
```

### CONTROLLER

Le contrôleur fonctionne toujours pareil, il accepte un type générique transmet à access.

```
//Modifie un livre, un dvd ou une revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocuments<T>(T unDocument)
{
    return access.ModifierDocument(unDocument);
}
```

## DAL

C'est comme pour la modification d'une commande de livre.

```
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonModifierDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de modification
        string id = "";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
            case Dvd dvd:
                id = dvd.Id;
                break;
            case Revue revue:
                id = revue.Id;
                break;
            case CommandeDocument uneCmd:
                id = uneCmd.Id; 
                break;
        }
        List<Object> liste = TraitementRecup<Object>(PUT, typeof(T).Name.ToLower() + "/" + id + "/" + jsonModifierDocument.Replace(" ", "-"));
        return (liste != null);
    }
    catch (Exception ex)
```

## API

Le cheminement dans l'API est identique à celui pour la modification d'une commande de livre.  
`Mediatekdocuments.php => Controle.php => AccesBDD.php => modifiCommandeLivreOuDvd()`

```
public function modifiCommandeLivreOuDvd($id, $champs) {
    $champsCommande = [
        "dateCommande" => $champs["DateCommande"],
        "montant" => $champs["Montant"]
    ];

    $champsCommandeDocument = [
        "nbExemplaire" => $champs["NbExemplaire"],
        "idLivreDvd" => $champs["IdLivreDvd"],
        "idSuivi" => $champs["IdSuivi"]
    ];

    $result = $this->updateOne("commande", $id, $champsCommande);
    if ($result == null || $result == false) {
        return null;
    }

    return $this->updateOne("commandedocument", $id, $champsCommandeDocument);
}
```

## Supprimer une commande de DVD

### VIEW

Après contrôle le contrôleur est appelé.

```
//Action du bouton supprimer pour une commande de dvd
1 référence | hedi-k, il y a 6 jours | 1 auteur, 1 modification
private void btnSupprimerCmdDvd_Click(object sender, EventArgs e)
{
    if (dgvCmdDvd.CurrentCell != null)      //vérification selection
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //CommandeDocument uneCommande = (CommandeDocument)bdgCommandeDvdsListe.List[bdgDvdListe.Position];
            CommandeDocument uneCommande = (CommandeDocument)bdgCommandeDvdsListe.List[bdgCommandeDvdsListe.Position];
            //Contrôle sur le numéro d'id, 3 = livré 4 = réglé.
            if (int.Parse(uneCommande.IdSuivi) < 3)
            {
                if (controller.SupprimerDocument(uneCommande))
                {
                    MessageBox.Show("Commande supprimé.");
                    tabOngletCommandeDvd_Enter(null, null);
                }
            }
        }
    }
}
```

### CONTROLLER

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access.

```
//Supprime un livre, dvd ou revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool SupprimerDocument<T>(T unDocument)
{
    return access.SupprimerDocument(unDocument);
}
```

### DAL

C'est la même méthode pour supprimer que les précédents on ajoute le cas d'une CommandeDocument dans la méthode de suppression.

```
        break;
    case CommandeDocument uneCmd:
        id = uneCmd.Id;
        break;
}
List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + "Id\":" + id + "}");
return (liste != null);
}
catch (Exception ex)
```

## Tâche 2 : gérer les commandes de revues

Tâches à réaliser

- Créer un onglet (ou une fenêtre) 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.

- La charte graphique doit correspondre à l'existant.
- Dans toutes les listes, permettre le tri sur les colonnes.
- 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 début d'abonnement.
- 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.
- Permettre de supprimer une commande de revue uniquement 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 début d'abonnement). Pour cela, créer et utiliser la méthode 'ParutionDansAbonnement' qui reçoit en paramètre 3 dates (date commande, date début 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.
- Toutes les sécurités seront mises en place pour éviter des erreurs de manipulation.
- Créer une méthode 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 début d'abonnement) triée sur la date dans l'ordre chronologique.

## Classes nécessaire aux commandes

Cette fois la commande d'une revue correspond à un abonnement alors je crée la classe abonnement qui va hériter de commande en me référant au MCD.

```
27 références | hedi-k, il y a 4 jours | 1 auteur, 1 modification
public class Abonnement : Commande
{
    5 références | hedi-k, il y a 4 jours | 1 auteur, 1 modification
    public DateTime DateFinAbonnement { get; }
    6 références | hedi-k, il y a 4 jours | 1 auteur, 1 modification
    public string IdRevue { get; }

    1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
    public Abonnement(string id, DateTime dateCommande, double montant, DateTime dateFinAbonnement, string idRevue)
        : base(id, dateCommande, montant)
    {
        DateFinAbonnement = dateFinAbonnement;
        IdRevue = idRevue;
    }
}
```

## Commande d'une revue

Le procédé est relativement identique sauf que l'on ne va plus traiter des commandes document mais des objets de type abonnement

## Recherche d'une revue

L'action du bouton recherche me permet de sélectionner une revue.

```

//Action du bouton Recherche dans commande Revue
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
private void btnRechCmdRevue_Click(object sender, EventArgs e)
{
    Revue revue = lesRevues.Find(x => x.Id.Equals(txBCmdRevueNum.Text));
    if (revue != null)
    {
        AfficheCommandeRevueInfos(revue);
    }
    else{ MessageBox.Show("numéro introuvable"); }
}

```

Une fois une revue sélectionné, l'action du bouton ajouter me permet d'accéder au grpBox de commande de revue.

```

}
//Action du bouton Ajouter une commande de revue
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
private void btnAjouterCmdRevue_Click(object sender, EventArgs e)
{
    if (txBCmdRevueTitre.Text != null)
    {
        grpCmdRevue2.Enabled = true;
    }
}
//Action du bouton valider une revue
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification

```

Commande de Revue

Numéro de commande :

Date de commande :  vendredi 1 mars

Montant :

Date de fin abonnement :  vendredi 1 mars

Le bouton valider va transmettre au contrôleur.

La différence est le type de données que l'on traite ici, ce sont des dates pour la valorisation.

```

//Action du bouton valider une revue
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
private void btnCmdRevueValider_Click(object sender, EventArgs e)
{
    Abonnement cmdRevue = SuperCmdRevue();
    if (cmdRevue != null)
    {
        if (controller.EnvoiDocuments(cmdRevue))
        {
            MessageBox.Show("Abonnement pris");
            VideCmdRevue();
            tabOngletCommandeRevue_Enter(null, null);
        }
    }
}

```

## CONTROLLER

C'est toujours le même qui est appelé, il est capable de gérer un objet Abonnement comme un type générique.

```
//Cette méthode envoi à l'accès livre, dvd et Revue.  
//En C#, T est un paramètre de type générique.  
//Ce qui implique quelle va traiter indifféremment un objet Livre/Dvd/revue  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool EnvoiDocuments<T>(T unDocument)  
{  
    return access.CreerDocument(unDocument);  
}
```

## DAL

Comme pour les précédents,

```
//Envoi l'objet en paramètre à l'API  
1 référence | 0 modification | 0 auteur, 0 modification  
public bool CreerDocument<T>(T unDocument)  
{  
    //Convertit en json l'objet en paramètre  
    String jsonCreerDocument = JsonConvert.SerializeObject(unDocument);  
    try  
    {  
        //Appel TraitementRecup avec en paramètre POST, le nom du type d'objet et le json  
        List<Object> liste = TraitementRecup<Object>(POST, typeof(T).Name.ToLower() + "/" + jsonCreerDocument.Replace(" ", "-"));  
        return (liste != null);  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
    }  
    return false;  
}
```

## API

Au sein de l'API, [Mediatekdocuments.php](#) récupère les paramètres et en fonction du verbe HTTP utilisé.

```
// traitement suivant le verbe HTTP utilisé  
if ($_SERVER['REQUEST_METHOD'] === 'GET') {  
    $controle->get($table, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    $controle->post($table, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {  
    $controle->put($table, $id, $champs);  
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {  
    $controle->delete($table, $champs);  
}
```



## [Controle.php](#)

```

public function post($table, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->ajoutLivre($champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->ajoutDvd($champs);
    } elseif ($table == "revue") {
        $result = $this->accessBDD->ajoutRevue($champs);
    } elseif ($table == "commandedocument") {
        $result = $this->accessBDD->ajoutCommandeLivreOuDvd($champs);
    } elseif ($table == "abonnement") { ←
        $result = $this->accessBDD->ajoutCommandeAbonnementRevue($champs);
    } else {

```

Dans le fichier [AccesBDD.php](#) on construit les requêtes SQL

```

public function ajoutCommandeAbonnementRevue($champs) {
    $champsCommande = [
        "id" => $champs["Id"],
        "dateCommande" => $champs["DateCommande"],
        "montant" => $champs["Montant"]
    ];

    $champsAbonnement = [
        "id" => $champs["Id"],
        "dateFinAbonnement" => $champs["DateFinAbonnement"],
        "idRevue" => $champs["IdRevue"],
    ];

    $result = $this->insertOne("commande", $champsCommande);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->insertOne("abonnement", $champsAbonnement);
}

```

## Supprimer une commande de revue

Pour supprimer une commande de revue il est demandé dans les consignes qu'aucun exemplaire ne soit rattaché. Donc de faire une vérification sur les dates. Il faut faire une méthode 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

```
//Compare les dates
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
private bool ParutionEntreCmdEtAbonnement(DateTime dateCmd, DateTime dateFinAbo, DateTime dateParu)
{
    //retourne faux si une parution est en cours.
    if (dateCmd < dateParu && dateFinAbo > dateParu)
    {
        return false;
    }
    else
    { return true; }
}
```

Donc l'idée est de chercher si une revue à un exemplaire. Si c'est le cas de faire appel à cette fonction qui va comparer les dates de l'exemplaire pour savoir si un abonnement est en cours.

Donc a chaque revue sélectionnée, je parcours la liste des exemplaires.

(La liste `lesExemplaires` et le controller `GetExemplairesRevue()` sont fournis avec l'application, je ne les détailleras pas.)

```
//Récupère la liste des exemplaire pour un id donné. Si trouve l exemplaire parcour la liste.
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
private bool ParcourExemplaire(DateTime dateCmd, DateTime dateFInAbo, string idDocument)
{
    lesExemplaires = controller.GetExemplairesRevue(idDocument);
    foreach (Exemplaire exemplaire in lesExemplaires)
    {
        if (ParutionEntreCmdEtAbonnement(dateCmd, dateFInAbo, exemplaire.DateAchat))
        {
            return false;
        }
    }
    return true;
}
```

Donc l'action du bouton suppression est :

```
//Action du bouton Supprimer pour une commande de Revue
1 référence | hedi-k, il y a 2 jours | 1 auteur, 2 modifications
private void btnSupprimerCmdRevue_Click(object sender, EventArgs e)
{
    if (dgvCmdRevue.CurrentCell != null)      //vérification selection
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            Abonnement uneCmd = (Abonnement)bdgCommandeRevueListe.List[bdgCommandeRevueListe.Position];
            //Ou parcourExemplaire est nulle alors supprime
            if (ParcourExemplaire(uneCmd.DateCommande, uneCmd.DateFinAbonnement, uneCmd.IdRevue))
            {
                controller.SupprimerDocument(uneCmd);
                MessageBox.Show("Commande supprimé.");
                tabOngletCommandeRevue_Enter(null, null);
            }
            else{ MessageBox.Show("Une parution est en cours.");}
        }
        else{ MessageBox.Show("selectionner une Commande !");}
    }
}
```

## CONTROLLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access.

```
//Supprime un livre, dvd ou revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool SupprimerDocument<T>(T unDocument)
{
    return access.SupprimerDocument(unDocument);
}
```

## DAL

C'est la même méthode pour supprimer que les précédents on ajoute le cas d'un Abonnement dans la méthode de suppression.

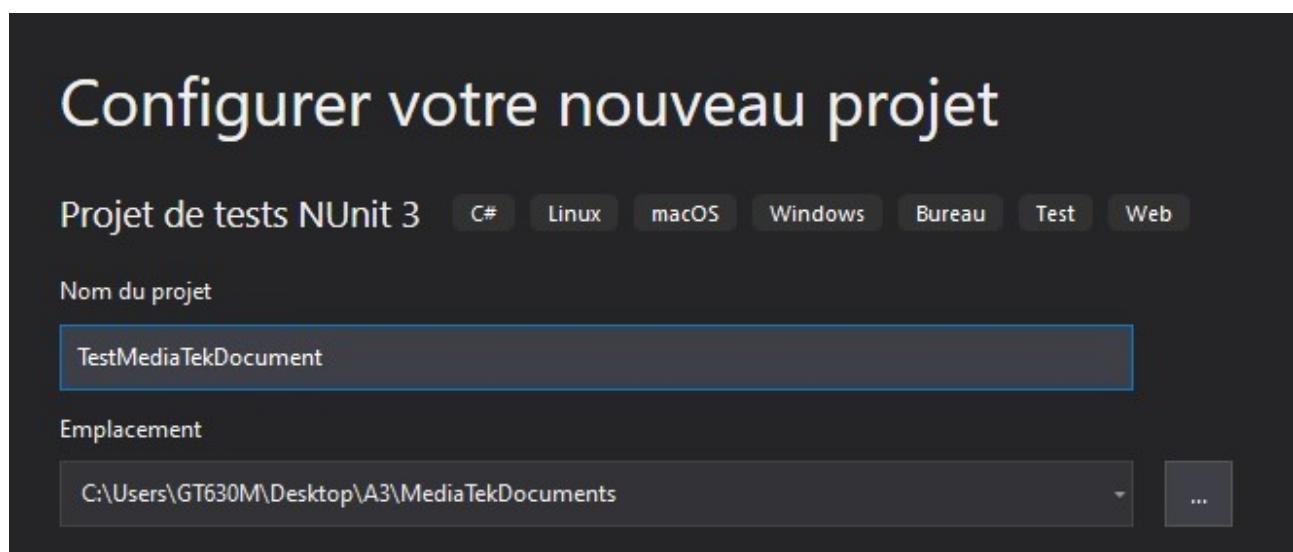
```
        break;
    case Abonnement unAbo:
        id = unAbo.Id;
        break;
}
List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + "Id\":" + id + "}");
return (liste != null);
```

Ensuite dans l'API on suit le même cheminement, [Mediatekdocuments.php](#) => [Controle.php](#) => [AccesBDD.php](#) jusqu'à la fonction `delete()`

## Test unitaire

Il est demandé dans les consignes de faire un test unitaire de la méthode de contrôle pour la suppression.

Je configure le projet de test.



Ensuite je le lie à mon projet

## Gestionnaire de références - TestMediaTekDocuments

?

X

Projets		Rechercher (Ctrl+E)
Solution	Nom	Chemin d'accès
Projets partagés	<input checked="" type="checkbox"/> MediaTekDocuments	C:\Users\GT630M\Desktop\MediaTekDocuments
Parcourir		

Je ne peux pas directement tester ma méthode car elle est en privée. La passer en public est possible mais ce n'est pas une bonne pratique.

Alors je la copie dans la classe de test.

```

using NUnit.Framework;
using System;

namespace TestMediaTekDocuments
{
    [SetUp]
    public void Setup()
    {
    }

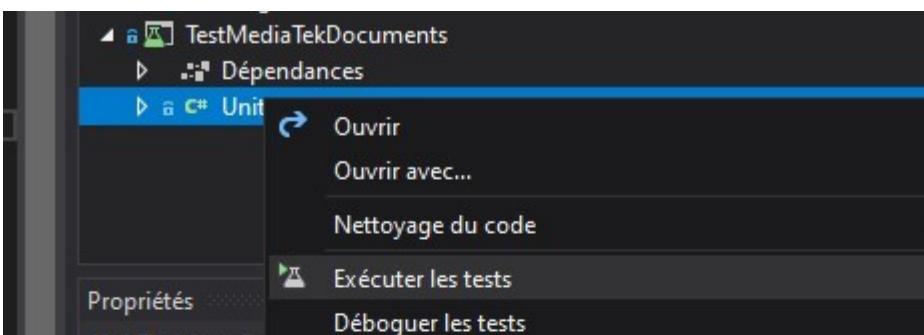
    [Test]
    public void Test1()
    {
        DateTime dateCommande = new DateTime(2024, 01, 01);
        DateTime dateFinAbonnement = new DateTime(2025, 01, 01);
        DateTime dateParution = new DateTime(2024, 02, 02);

        bool resultat = ParutionEntreCmdEtAbonnement(dateCommande, dateFinAbonnement, dateParution);
        Assert.IsFalse(resultat);
        Assert.Pass("essai");
    }

    //Compare les dates
    private bool ParutionEntreCmdEtAbonnement(DateTime dateCmd, DateTime dateFinAbo, DateTime dateParu)
    {
        if (dateCmd < dateParu && dateFinAbo > dateParu)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
}

```

Ensuite je fais le test.



Sortie

Afficher la sortie à partir de : Build

```
L'opération de génération a démarré...
1>----- Début de la génération : Projet : MediaTekDocuments, Configuration : Debug Any CPU -----
1>C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\Microsoft.
1> MediaTekDocuments -> C:\Users\GT630M\Desktop\A3\MediaTekDocuments\MediaTekDocuments\bin\Debu
===== Génération : 1 a réussi, 0 a échoué, 1 à jour, 0 a été ignoré =====
|
```



A la console ce n'est pas le plus jolie pour lire un teste mais je ne sais pas pourquoi j'ai un bug et cela ne fonctionne pas si je procède avec l'explorateur de test.

### Abonnement se terminant

Il est demandé de créer une méthode qui permet d'obtenir la liste des revues dont l'abonnement se termine dans moins de 30 jours. Dès l'ouverture de l'application.

Je fais une fonction qui va récupérer les liste des revues et commandes de revues (abonnement). Elle va parcourir la liste des commandes de revue et soustraire 30 jours à la date de fin d'abonnement. Si le résultat est inférieur à la date du jour alors il reste moins de 30 jours à cette abonnement. Donc sur cette abonnement on valorise une revue avec l'id qu'ils ont en commun afin de pouvoir récupérer le titre de la revue.

Et on valorise une variable avec un message qui contient le titre et la date de fin.

```
//Contrôle abonnement - 30jours
1 référence | hedi-k, il y a 4 jours | 1 auteur, 1 modification
private void AbonnementsSurLaFIn()
{
    string a = "Les abonnements qui se finissent sous moins de 30 jours sont : \n";
    lesRevues = controller.GetAllRevues(); ;
    listCmdRevues = controller.GetAllCommandeRevues();
    foreach (Abonnement aboRevue in listCmdRevues)
    {
        DateTime dateMoins30Jours = aboRevue.DateFinAbonnement.AddDays(-30);
        if (dateMoins30Jours < DateTime.Now)
        {
            Revue revue = lesRevues.Find(x => x.Id.Equals(aboRevue.IdRevue));
            a = a + "titre : " + revue.Titre + " date de fin : " + aboRevue.DateFinAbonnement.ToString() + "\n";
        }
    }
    MessageBox.Show(a);
}
```

Il suffit plus qu'appeler cette fonction au lancement de l'application.

```
2 références | hedi-k, il y a 20 jours | 1 auteur, 1 modification
internal FrmMediatek()
{
    InitializeComponent();
    this.controller = new FrmMediatekController();
    AbonnementsSurLaFIn();
}
```

Mission terminé, je met à jour mon kanban et passe à la mission 3.

The Kanban board displays tasks for Mission 3 across three columns: To do, In progress, and Done.

- To do:** 10 items (all Draft).
  - Mission 4 : mettre en place des authentifications
  - Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 1 : corriger des problèmes de sécurité
  - Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 2 : contrôler la qualité
  - Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 3 : intégrer des logs
  - Mission 6 : tester et documenter Tâche 1 : gérer les tests
- In progress:** 1 item (Draft).
  - Mission 3 : gérer le suivi de l'état des exemplaires tâche
- Done:** 4 items (all Draft).
  - Mission 1 : gérer les documents tâche1 Dans les onglets actuels (Livres, Dvd, Revues), ajouter les fonctionnalités (boutons) qui permettent d'ajouter, de modifier ou de supprimer un document.
  - Mission 1 : gérer les documents tâche2 Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Document, idem pour LivresDvd.
  - Mission 2 : gérer les commandes Tâche 1 : gérer les commandes de livres ou de DVD
  - Mission 2 : gérer les commandes Tâche 2 : gérer les commandes de revues

## Mission 3 : gérer le suivi de l'état des exemplaires

Tâches à réaliser

Agrandir la fenêtre en hauteur.

— Dans l'onglet Livres, partie basse, ajouter la liste des exemplaires du livre sélectionné. Cette liste d'exemplaires doit contenir les colonnes suivantes :

numéro d'exemplaire, date achat et libellé de l'état. La liste doit être triée par date d'achat, dans l'ordre inverse de la chronologie, et le clic sur une colonne doit permettre le tri sur la colonne. Sur la sélection d'un exemplaire, il doit être possible de changer son état.

— Le principe est le même pour les DVD.

— Dans l'onglet "Parutions des revues", liste des parutions, remplacer la colonne "Photo" par "Etat". Permettre aussi le changement d'état.

— Permettre de supprimer un exemplaire.

Je modifie mon interface et déplace la position des boutons pour que cela correspond à la demande. Je compte afficher les exemplaires dans une data grid view et utiliser une comboBox pour modifier les états.

## Exemplaires Livres

**VIEW**

## Gestion des documents de la médiathèque

Livres DVD Revues Parutions des revues Commande Livres Commande DVD Commande Revues

Recherches

Saisir le titre ou la partie d'un titre :

Ou sélectionner le genre :  X

Saisir un numéro de document :

**Rechercher**

Ou sélectionner le public :  X

**Ajouter**

**Supprimer**

**Modifier**

Ou sélectionner le rayon :  X

Id	Titre	Auteur	Collection	Genre	Public	Rayon
00102	a	a	a	Bande dessinée	Adultes	DVD films
01002	AB	a	a	Aventures	Adultes	DVD films
00017	Catastrophes au Brésil	Philippe Masson		Policier	Ados	Jeunesse romans
00007	Dans les coulisses du musée	Kate Atkinson		Roman	Tous publics	Littérature étrangère
00003	Et je danse aussi	Anne-Laure Bondoux		Comédie	Tous publics	Littérature française
00019	Guide Vert - Iles Canaries		Guide Vert	Voyages	Tous publics	Voyages
00020	Guide Vert - Irlande		Guide Vert	Voyages	Tous publics	Voyages
00008	Histoire du juif errant	Jean d'Ormesson		Roman	Adultes	Littérature française

Informations détaillées

Numéro de document :

Code ISBN :

Image :

Titre :

Auteur(e) :

Collection :

Genre :

Public :

Rayon :

Chemin de l'image :

Gestion des exemplaires

	n° Exemplaire	DateAchat	Etat
▶	1	27/02/2024	inutilisable
	3	27/02/2024	neuf

Etat de l'exemplaire:

**Modifier**

**Supprimer**

Dans un premier temps je dois récupérer les états qui sont dans la table état.  
Je crée une nouvelle liste d'états.

```
private List<Etat> lesEtats = new List<Etat>();
```

A l'ouverture de l'onglet livre la liste est chargée avec les états issu de la BDD.

```
2 références (neufs, il y a 2 jours) | 1 auteur, 5 modifications
private void TabLivres_Enter(object sender, EventArgs e)
{
    lesLivres = controller.GetAllLivres();
    RemplirComboCategorie(controller.GetAllGenres(), bdgGenres, cbxLivresGenres);
    RemplirComboCategorie(controller.GetAllPublics(), bdgPublics, cbxLivresPublics);
    RemplirComboCategorie(controller.GetAllRayons(), bdgRayons, cbxLivresRayons);
    RemplirLivresListeComplete();
    lesEtats = controller.GetAllEtats();
    PresChargeDGVExemplaireLivre();
    listCmdLivres = controller.GetAllCommandeLivres();
}
```

Je crée une fonction qui me permet de remplir les comboBox pour la modification des états.

```
3 références | hedi-k, il y a 2 jours | 1 auteur, 1 modification
public void RemplirComboEtat(List<Etat> lesEtats, BindingSource bdg, ComboBox cbx)
{
    bdg.DataSource = lesEtats;
    cbx.DataSource = bdg;
    cbx.DisplayMember = "Libelle"; //Pour forcer l'affichage des libelles sans modifier le modèle Etat
    if (cbx.Items.Count > 0)
    {
        cbx.SelectedIndex = -1;
    }
}
//Retourne l'ID de l'état sélectionné dans les cbxEtat
```

Et la fonction qui va me donner l'id de l'état en fonction de son label sélectionné dans la comboBox

```
//Retourne l'ID de l'état sélectionné dans les cbxEtat
3 références | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private string GetIdEtat(string unEtat)
{
    List<Etat> uneListe = controller.GetAllEtats();
    foreach (Etat Etat in uneListe)
    {
        if (Etat.Libelle == unEtat)
        {
            return Etat.Id;
        }
    }
    return null;
}
```

Je fais une première méthode qui a pour but de récupérer la liste des exemplaires en fonction du livre sélectionné. J'utilise la textBox qui contient l'id du livre pour faire une recherche sur l'id des exemplaires qu'ils ont en commun.

```
//Rempli la combo charge la liste des exemplaire et appelle pour la dgv.
4 références | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void PresChargeDGVExemplaireLivre()
{
    if (txbLivresNumero.Text != null)
    {
        lesExemplaires = controller.GetExemplairesRevue(txbLivresNumero.Text);
        RemplirComboEtat(lesEtats, bdgEtats, cbxLivreEtat);
        ChargeDGVExemplaireLivre(lesExemplaires, lesEtats);
    }
}
```

Ensuite [ChargeDGVExemplaireLivre\(\)](#) va remplir la data grid view. Avec tout les exemplaires de la liste si il y en a. Et pour chaque exemplaire va associer l'id de l'état avec son label.

```

//Rempli la dgv des exemplaire du livre sélectionné dans l'onglet livre
2 références | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void ChargeDGVExemplaireLivre(List<Exemplaire> exemplaires, List<Etat> lesEtats)
{
    if (exemplaires != null)
    {
        foreach (var exemplaire in exemplaires)
        {
            // Rechercher le libellé correspondant à l'idEtat de l'exemplaire
            var etat = lesEtats.FirstOrDefault(e => e.Id == exemplaire.IdEtat);
            // Si un état correspondant est trouvé, remplacer l'idEtat par le libellé
            if (etat != null)
            {
                exemplaire.IdEtat = etat.Libelle;
            }
        }
        bdgExemplairesListe.DataSource = exemplaires;
        dgvLivreExemplaire.DataSource = bdgExemplairesListe;
        dgvLivreExemplaire.Columns["photo"].Visible = false;
        dgvLivreExemplaire.Columns["id"].Visible = false;
        dgvLivreExemplaire.Columns[0].HeaderText = "n° Exemplaire";
        dgvLivreExemplaire.Columns[3].HeaderText = "Etat";
    }
}
//+rie sur la DGVExemplaire

```

Une fois la liste des exemplaires chargée, il est possible de sélectionner un exemplaire pour lui modifier son état en sélectionnant un nouvel état via la comboBox et en cliquant sur valider.

Pour créer l'objet exemplaire, j'utilise cette fois la fonction DataBoundItem qui me permet de récupérer les données d'un objet dans une data grid view.  
(créer un exemplaire via son id en parcourant la liste des exemplaires est possible mais c'est pour essayer autre chose)

```

//Action du btn modifier pour l'état d'un exemplaire
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void btnLivreExemplaireModifier_Click(object sender, EventArgs e)
{
    if (dgvLivreExemplaire.CurrentCell != null)
    {
        if (MessageBox.Show("Modifier ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //Récupère dans la variable les données de l'objet sélectionné dans la dgv
            DataGridViewRow row = dgvLivreExemplaire.SelectedRows[0];
            //DataBoundItem permet l'accès aux données de l'objet
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            exemplaire.IdEtat = GetIdEtat(cbxLivreEtat.Text);
            if (exemplaire.IdEtat != null)
            {
                if (controller.ModifierDocuments(exemplaire))
                {
                    PresChargeDGVExemplaireLivre();
                }
            }
            else { MessageBox.Show("Selectionner un Etat pour le modifier."); }
        }
    }
}

```

## CONTROLLE

Dans le contrôleur c'est toujours la même fonction est appelée et qui transmet à l'access. Comme elle attend un type générique elle peut traiter un exemplaire

```
//Modifie un livre, un dvd ou une revue
1 référence | 0 modification | 0 auteur, 0 modification
public bool ModifierDocuments<T>(T unDocument)
{
    return access.ModifierDocument(unDocument);
}
```

## DAL

Pour la modification d'un exemplaire il suffit d'ajouter la possibilité. Attention un exemplaire à pour identifiant son numéro car l'id est commun a tout les exemplaire pour un même livre. De plus un numéro est un int alors il faut le convertir en string pour qu'il puisse être traité.

```
        break;
    case Exemplaire exemplaire:
        id = exemplaire.Numero.ToString(); // C'est un INT donc faut le convertir en string
        break;
}
List<Object> liste = TraitementRecup<Object>(PUT, typeof(T).Name.ToLower() + "/" + id + "/" + jsonModifierDocument.Replace(" ", "-"));
return (liste != null);
}
catch (Exception ex)
```

## API

Ensuite dans l'API on suit le même cheminement, [Mediatekdocuments.php](#) => [Controle.php](#) =>

```
public function put($table, $id, $champs) {
    if ($table == "livre") {
        $result = $this->accessBDD->modifiLivre($id, $champs);
    } elseif ($table == "dvd") {
        $result = $this->accessBDD->modiDvd($id, $champs);
    } elseif ($table == "revue") {
        $result = $this->accessBDD->modifiRevue($id, $champs);
    } elseif ($table == "commandedocument") {
        $result = $this->accessBDD->modifiCommandeLivreOuDvd($id, $champs);
    } elseif ($table == "exemplaire") { 
        $result = $this->accessBDD->modifiExemplaire($id, $champs);
    } else {
        $result = $this->accessBDD->updateOne($table, $id, $champs);
    }
    if ($result == null || $result == false) {
        $this->reponse(400, "requete invalide");
    } else {
        $this->reponse(200, "OK");
    }
}
```

## [AccesBDD.php](#)

```
/*
public function modifiExemplaire($id, $champs) {
    $champsModif = [
        "idEtat" => $champs["IdEtat"]
    ];
    return $this->updateOneBis("exemplaire", $id, $champsModif);
}
```

## Suppression

La suppression fonctionne de la même manière, elle appelle le contrôleur

```
//Action du btn supprimer pour un exemplaire
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void btnLivreExemplaireSupprimer_Click(object sender, EventArgs e)
{
    if (dgvLivreExemplaire.CurrentCell != null)
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //Récupère dans la variable les données de l'objet sélectionné dans la dgv
            DataGridViewRow row = dgvLivreExemplaire.SelectedRows[0];
            //DataBoundItem permet l'accès aux données de l'objet
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            if (exemplaire.Numero != null)
            {
                if (controller.SupprimerDocument(exemplaire))
                {
                    PresChargeDGVExemplaireLivre();
                }
            }
            else { MessageBox.Show("Selectionner un Exemplaire pour le supprimer."); }
        }
    }
}
```

Le contrôleur appelle [SupprimerDocument\(\)](#) dans access. La il faut ajouter le cas d'un exemplaire mais les exemplaires ne sont pas identifiés avec « Id » mais « Numero » alors il faut ajouter une variable pour préciser ce cas. Ou modifier l'API pour un exemplaire comme j'ai fais pour la modification d'un exemplaire.

```
        break;
    case Exemplaire exemplaire:
        id = exemplaire.Numero.ToString(); // C'est in INT donc faut le convertir en string
        Id = "Numero"; ←
        break;
    }
    List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + Id + ":" + id + "}");
    return (liste != null);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

Et l'API remonte jusqu'à la fonction [delete\(\)](#) de [AccesBDD.php](#)

## Exemplaires DVD

Pour la gestion des exemplaires de DVD, le fonctionnement est identique.  
J'ai une data grid view qui contient les exemplaires de dvd.

```
//Rempli la dgv des exemplaire du dvd sélectionné dans l onglet dvd
2 références | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void ChargeDGVExemplaireDvd(List<Exemplaire> exemplaires, List<Etat> lesEtats)
{
    if (exemplaires != null)
    {
        foreach (var exemplaire in exemplaires)
        {
            // Rechercher le libellé correspondant à l'idEtat de l'exemplaire
            var etat = lesEtats.FirstOrDefault(e => e.Id == exemplaire.IdEtat);
            // Si un état correspondant est trouvé, remplacer l'idEtat par le libellé
            if (etat != null)
            {
                exemplaire.IdEtat = etat.Libelle;
            }
        }
        bdgExemplairesListe.DataSource = exemplaires;
        dgvDvdExemplaire.DataSource = bdgExemplairesListe;
        dgvDvdExemplaire.Columns["photo"].Visible = false;
        dgvDvdExemplaire.Columns["id"].Visible = false;
        dgvDvdExemplaire.Columns[0].HeaderText = "n° Exemplaire";
        dgvDvdExemplaire.Columns[3].HeaderText = "Etat";
    }
}
```

L'action du bouton modifier fonctionne avec le même contrôler et dal que celui d'un exemplaire de livre.

```
//Modifie l état d'un exemplaire
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void btnDvdExemplaireModifier_Click(object sender, EventArgs e)
{
    if (dgvDvdExemplaire.CurrentCell != null)
    {
        if (MessageBox.Show("Modifier ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            DataGridViewRow row = dgvDvdExemplaire.SelectedRows[0];
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            exemplaire.IdEtat = GetIdEtat(cbxLivreEtat.Text);
            if (exemplaire.IdEtat != null)
            {
                if (controller.ModifierDocuments(exemplaire))
                {
                    PresChargeDGVExemplaireDvd();
                }
            }
            else { MessageBox.Show("Selectionner un Etat pour le modifier."); }
        }
    }
}
```

Controleur => Dal => API => function modifiExemplaire()

Il en est de même pour la fonction supprimer.

```

//Action du btn supprimer pour un exemplaire de dvd
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void btnDvdExemplaireSupprimer_Click(object sender, EventArgs e)
{
    if (dgvDvdExemplaire.CurrentCell != null)
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            DataGridViewRow row = dgvDvdExemplaire.SelectedRows[0];
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            if (exemplaire.Numero != null)
            {
                if (controller.SupprimerDocument(exemplaire))
                {
                    PresChargeDGVEExemplaireDvd();
                }
            }
            else { MessageBox.Show("Selectionner un Exemplaire pour le supprimer."); }
        }
    }
}

```

Controleur => Dal => API => function delete()

## Exemplaires revues

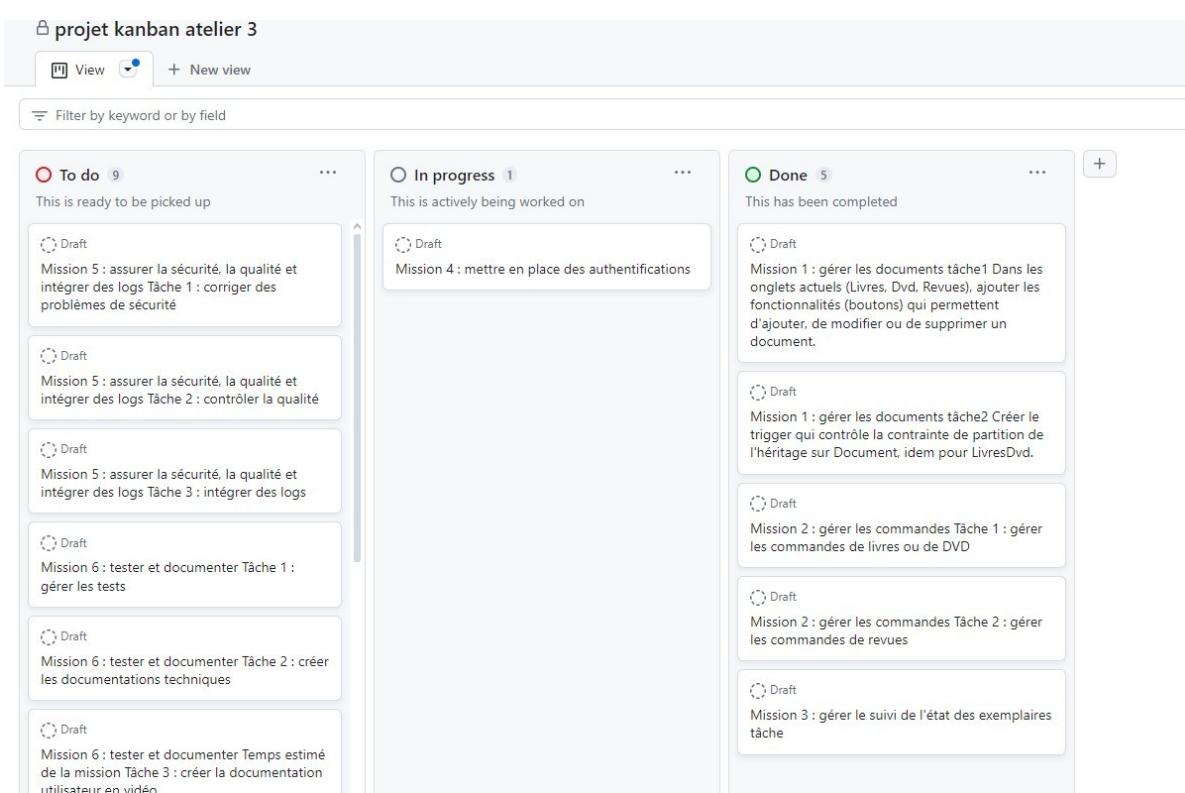
Pour les revues il est imposé de mettre la gestion dans l'onglet Parutions revues

Numéro	DateAchat	Etat
3	27/02/2024	neuf
1	12/12/2023	inutilisable

A part cela ce sont le même type d'objets Exemplaire alors cela fonctionne de la même manière que pour un livre ou un exemplaire.

```
//Action du bouton modifier un exemplaire pour une revue
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void btnRevueExemplaireModifier_Click(object sender, EventArgs e)
{
    if (dgvReceptionExemplairesListe.CurrentCell != null)
    {
        if (MessageBox.Show("Modifier ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            DataGridViewRow row = dgvReceptionExemplairesListe.SelectedRows[0];
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            exemplaire.IdEtat = GetByIdEtat(cbxRevuEtat.Text);
            if (exemplaire.IdEtat != null)
            {
                if (controller.ModifierDocuments(exemplaire))
                {
                    AfficheReceptionExemplairesRevue();
                }
            }
            else { MessageBox.Show("Selectionner un Etat pour le modifier."); }
        }
    }
}
//Action du btn supprimer pour un exemplaire pour une revue
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
private void btnRevueExemplaireSupprimer_Click(object sender, EventArgs e)
{
    if (dgvReceptionExemplairesListe.CurrentCell != null)
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            DataGridViewRow row = dgvReceptionExemplairesListe.SelectedRows[0];
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            if (exemplaire.Numero != null)
            {
                if (controller.SupprimerDocument(exemplaire))
                {
                    AfficheReceptionExemplairesRevue();
                }
            }
            else { MessageBox.Show("Selectionner un Exemplaire pour le supprimer."); }
        }
    }
}
```

Une fois cela finis, la mission 3 est terminée. Je mets à jour mon kaban et passe à la prochaine.



## Mission 4 : mettre en place des authentications

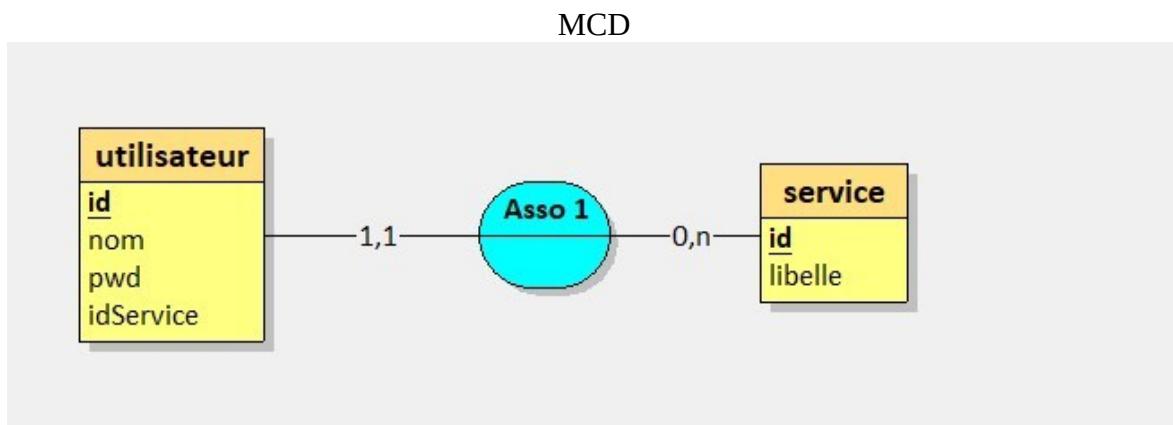
### Tâches à réaliser

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.

- Ajouter une première fenêtre d'authentication. Faire en sorte que l'application démarre sur cette fenêtre.
- Suivant le type de personne authentifiée, empêcher certains accès en rendant invisibles ou inactifs certains onglets ou objets graphiques.
- Dans le cas du service Culture qui n'a accès à rien, afficher un message précisant que les droits ne sont pas susants pour accéder à cette application, puis fermer l'application.
- Faire en sorte que l'alerte de n d'abonnement n'apparaisse que pour les personnes concernées (qui gèrent les commandes)

### Base de données

En prenant en compte les consignes et les informations du cahier des charges, je crée deux tables supplémentaires.



Je crée la table utilisateur, le champs qui mérite l'attention est pwd, il va contenir les mots de passe des utilisateurs, je lui attribue une taille de 64 caractères car c'est le minimum pour contenir un mot de passe d'un hachage SHA-256 que je compte utiliser.

The screenshot shows the MySQL Workbench interface with the 'Structure' tab selected. A SQL query window displays the creation of the 'utilisateur' table:

```
CREATE TABLE utilisateur(
    id VARCHAR(50) NOT NULL,
    nom VARCHAR(50) NOT NULL,
    pwd VARCHAR(64) NOT NULL,
    PRIMARY key(id),
    idService VARCHAR(50)
);
```

Ensuite la table service.

The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: mediatek86
- Table: utilisateur
- Toolbar buttons: Parcourir, Structure, SQL, Rechercher, Insérer, Exporter
- SQL Editor content:

```
1 CREATE TABLE service(
2     id VARCHAR(50),
3     libelle VARCHAR(50),
4     primary key(id)
5 );
```

Et je fais le lien entre ces deux tables. Comme un utilisateur appartient forcément à un service.

The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: mediatek86
- Table: utilisateur
- Toolbar buttons: Parcourir, Structure, SQL, Rechercher, Insérer, Exporter
- SQL Editor content:

```
1 ALTER TABLE utilisateur
2 ADD FOREIGN KEY (idService) REFERENCES service(id);
```

Je remplis la table service avec les informations du cahier des charges et des consignes.

The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: mediatek86
- Table: service
- Toolbar buttons: Parcourir, Structure, SQL, Rechercher, Insérer, Exporter, Importer, Privilèges, Opérations
- SQL Editor content:

```
1 INSERT INTO service(id, libelle) VALUES('01','admin'),('02','administratif'),('03','prêts'),('04','culture');
```

Et j'ajoute des utilisateurs dans la table utilisateur.

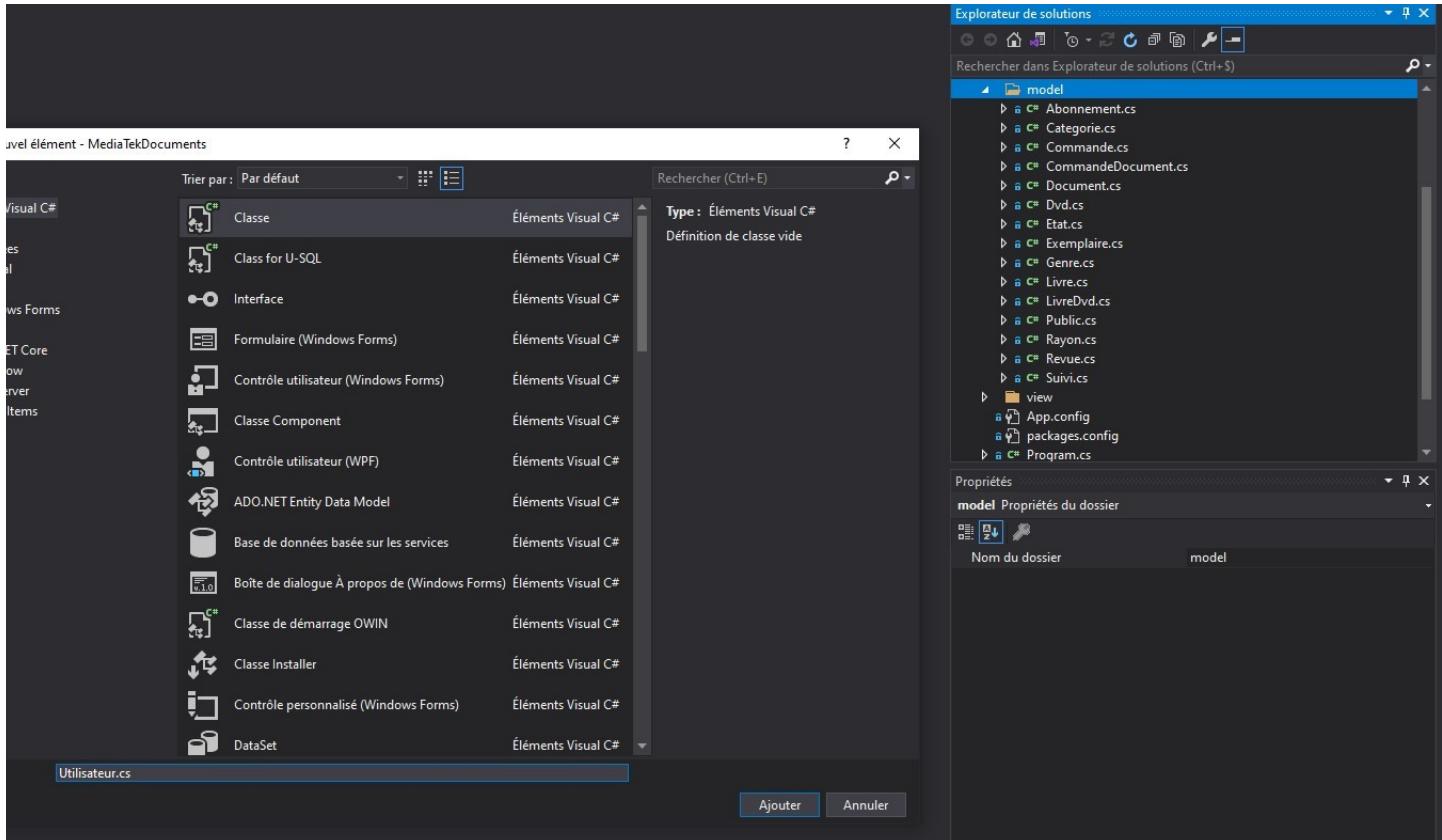
The screenshot shows the MySQL Workbench interface with the following details:

- Server: MySQL:3306
- Database: mediatek86
- Toolbar buttons: Structure, SQL, Rechercher, Requête, Exporter, Importer, Opérations, Privilèges
- SQL Editor content:

```
1 INSERT INTO utilisateur (id, nom, pwd, idService) VALUES ('101','admin', SHA2('mdp',256), '01');
```

## Model

Je crée la classe utilisateur qui correspond à la table utilisateur, elle va me servir pour l'authentification et l'attribution des droits dans l'application.

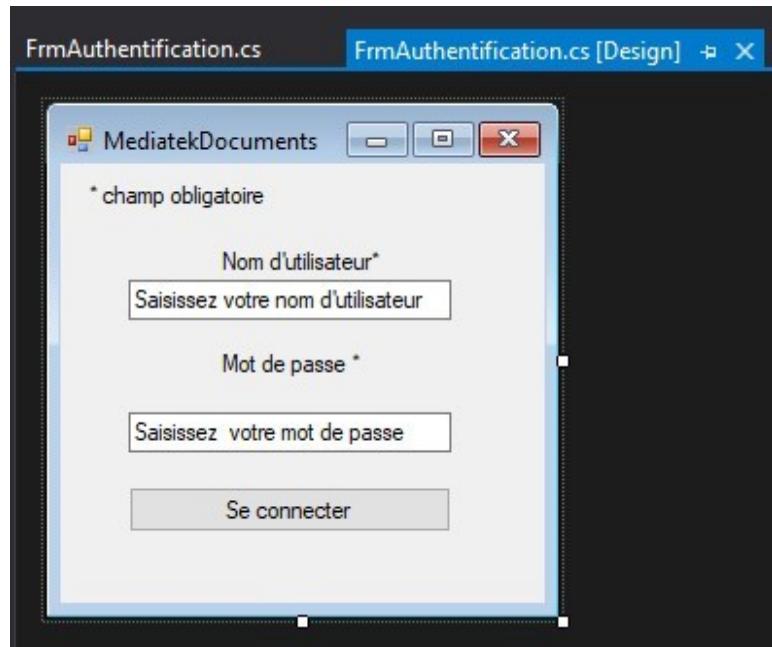


Son contenu correspond à la table de la base de données.

```
6
7     namespace MediaTekDocuments.model
8     {
9         public class Utilisateur
10        {
11            public string Id { get; }
12            public string Nom { get; }
13            public string Pwd { get; }
14            public string IdService { get; }
15
16            public Utilisateur (string id, string nom, string pwd, string idService)
17            {
18                Id = id;
19                Nom = nom;
20                Pwd = pwd;
21                IdService = idService;
22            }
23        }
24    }
```

## VIEW

Je créer une nouvelle vue que je nomme FrmAuthentication. Avec une interface simple, deux textBox et un bouton.



Pour forcer son démarrage au lancement de l'application il suffit de l'appeler dans classe Program.

```
static class Program
{
    /// <summary>
    /// Point d'entrée principal de l'application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new FrmAuthentication());
    }
}
```



Son fonctionnement est assez basique, il suffit d'entrer un nom d'utilisateur et un mot de passe, la combinaison de ses deux champs va valoriser un objet Utilisateur qui va être envoyé au contrôleur. Si le contrôleur retourne un utilisateur avec ses paramètres, cela va activer la vue principal FrmMediatek avec cet utilisateur en paramètre.

```

//Action du bouton
1 référence | 0 modification | 0 auteur, 0 modification
private void button1_Click(object sender, EventArgs e)
{
    String login = txtLogin.Text;
    String pwd = txtPwd.Text;

    Utilisateur utilisateur = new Utilisateur("", login, pwd, "");
    Utilisateur utilisateurConnecte = controller.Authentification(utilisateur);
    //Si le bon couple mot de passe utilisateur est entré il sera retourné et l'application va se lancer
    if (utilisateurConnecte != null)
    {
        this.Hide();
        FrmMediatek frm = new FrmMediatek(utilisateurConnecte);
        frm.ShowDialog();
    }
    else { MessageBox.Show("Authentification incorrecte !"); }

}
//Pour masquer les lettres que l'on entre comme mot de passe.
1 référence | 0 modification | 0 auteur, 0 modification
private void txtPwd_TextChanged(object sender, EventArgs e)
{
    txtPwd.PasswordChar = '*';
}

```

## CONTROLLER

Je crée un contrôleur dédié pour cette vue, son rôle n'est que d'envoyer à l'access l'utilisateur de la vue et éventuellement d'en retourner un.

```

namespace MediaTekDocuments.controller
{
    3 références | 0 modification | 0 auteur, 0 modification
    class FrmAauthenticationController
    {
        private readonly Access access;

        1 référence | 0 modification | 0 auteur, 0 modification
        public FrmAauthenticationController()
        {
            access = Access.GetInstance();
        }
        //reçoit un utilisateur de la vue et l'envoi au DAL.
        1 référence | 0 modification | 0 auteur, 0 modification
        public Utilisateur Authentication(Utilisateur utilisateur)
        {
            return access.Authentication(utilisateur);
        }
    }
}

```

## DAL

Dans [Access.cs](#) le principe est identique aux autres pour communiquer à l'API, je crée un JSON des données de l'utilisateur.

J'appelle la méthode GET car mon but ici est de contrôler et récupérer si l'utilisateur existe. Je nomme authentication la table pour la gestion dans l'API.

```

//Méthode pour l'authentification
1 référence | 0 modification | 0 auteur, 0 modification
public Utilisateur Authentification (Utilisateur utilisateur)
{
    String jsonAuthentification = JsonConvert.SerializeObject(utilisateur);
    try
    {
        List<Utilisateur> liste = TraitementRecup<Utilisateur>(GET,"authentification/"+ jsonAuthentification);
        if (liste != null)
        {
            //Il y a qu'un utilisateur qui peut être retourné donc indice 0
            return liste[0];
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return null;
}

```

## API

Dans mediatekdocuments.php rien ne change, on appelle le GET.

```

// traitement suivant le verbe HTTP utilisé
if ($_SERVER['REQUEST_METHOD'] === 'GET') { ←
    $controle->get($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controle->post($table, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
    $controle->put($table, $id, $champs);
} else if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    $controle->delete($table, $champs);
}

```

Dans le contrôleur il vient de préciser la nouvelle « table » qui n'en est pas une.

```

*/
public function get($table, $champs) {
    $result = null;
    if ($champs == "") {
        $result = $this->accessBDD->selectAll($table);
    } elseif ($table == "authentification") { ←
        $result = $this->accessBDD->authentification($champs);
    } else {
        $result = $this->accessBDD->select($table, $champs);
    }
    if (gettype($result) != "array" && ($result == false || $result == null)) {
        $this->reponse(400, "requête invalide");
    } else {
        $this->reponse(200, "OK", $result);
    }
}

```

Et dans **accessBDD** il faut créer la fonction qui va faire la requête.

Elle va récupérer le nom de l'utilisateur envoyé et hasher son mot de passe. Le tout sera envoyé à la BDD sous forme d'un select retourner l'utilisateur qui correspond.

```

public function authentification($champs) {
    $nom = $champs["Nom"];
    $pwd = hash('sha256', $champs["Pwd"]);

    $req = "SELECT * FROM utilisateur WHERE nom = '$nom' AND pwd = '$pwd';";
    return $this->conn->query($req);

}

}

```

Si le couple nom d'utilisateur et mot de passe correspond à un de ceux de la table utilisateur il sera retourné à la vue d'authentification.

Cette vue va lancer la seconde avec l'utilisateur en paramètre.

```

//Remplace FrmMediatek mais nécessaire pour récupérer l'utilisateur
1 référence | 0 modification | 0 auteur, 0 modification
public FrmMediatek(Utilisateur utilisateur)
{
    InitializeComponent();
    this.controller = new FrmMediatekController();
    Permission(utilisateur);
}

```

De la je crée une nouvelle fonction `Permission()` qui attend un utilisateur en paramètre.

Son but est de gérer l'affichage en fonction de l'utilisateur en paramètre et d'afficher et d'afficher l'alerte de fin d'abonnement que pour les administratif. Je fais le contrôle de service sur l'idService que contiennent les utilisateurs.

```

//Contrôle les accès des utilisateur
1 référence | 0 modification | 0 auteur, 0 modification
private void Permission(Utilisateur utilisateur)
{
    int service = int.Parse(utilisateur.IdService);
    switch (service)
    {
        case 4://service culture
            MessageBox.Show(" les droits ne sont pas suffisants pour accéder à cette application");
            Application.Exit();
            break;
        case 3://service prêt
            tabOngletCommandeLivre.Enabled = false;
            tabOngletCommandeDvd.Enabled = false;
            tabOngletCommandeRevue.Enabled = false;

            btnAjoutLivre.Enabled = false;
            btnAjoutDvd.Enabled = false;
            btnAjoutRevue.Enabled = false;
            btnSupprimerLivre.Enabled = false;
            btnSupprimerDvd.Enabled = false;
            btnSupprimerRevue.Enabled = false;
            btnModifier.Enabled = false;
            btnModifDvd.Enabled = false;
            btnModifierRevue.Enabled = false;

            groupBox1.Enabled = false;
            groupBox2.Enabled = false;
            grpReceptionExemplaire.Enabled = false;

            dgvCmdLivre.Visible = false;
            dgvCmdDvd.Visible = false;
            dgvCmdRevue.Visible = false;
            break;
        case 2://service administratif
            AbonnementsSurLaFIn();
            break;
        case 1://ADMIN
            break;
    }
}

```

Une fois cela fini, je mets à jour mon kanban et passe à la prochaine mission.



## Mission 5 : assurer la sécurité, la qualité et intégrer des logs

### Tâche 1 : corriger des problèmes de sécurité

Tâches à réaliser

Problème 1 : L'accès à l'API se fait en authentification basique, avec le couple « login:pwd » en dur dans le code de l'application. Le but est de sécuriser cette information.

Problème 2 : L'accès à l'API depuis son adresse sans mettre de paramètres donne la liste des fichiers contenus dans le dossier de l'API. Le but est d'avoir un retour d'erreur.

## Problème 1

Comme le demande les consignes, je crée un issue sur ce problème dans mon dépôt GitHub. Et je crée une seconde branche.

The screenshot shows a GitHub issue creation interface. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. Below the navigation bar, there's a title field containing "Problème1" with an orange arrow pointing to it. Underneath the title is a large text area labeled "Add a description". Inside this area, there's a note: "L'accès à l'API se fait avec une authentification basique, couple login:pwd en dure dans le code de l'application il faut sécuriser cette information". An orange arrow points to the word "pwd". At the bottom right of the text area is a green "Submit new issue" button, and at the bottom left is a note that "Markdown is supported".

Depuis mon IDE j'effectue un « fetch » pour récupérer le contenu du dépôt et je migre sur cette nouvelle branche.

(Il est inutile de faire un fetch dans mon cas car j'ai déjà le contenu mais ça simule le fait que je travail en coopération)

A screenshot of a PowerShell developer terminal window. The title bar says "PowerShell dév". The command history shows: "PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git fetch origin", "git checkout 1-problème1", "From https://github.com/hedi-k/Atelier3", "\* [new branch] 1-problème1 -> origin/1-problème1", and "Switched to a new branch '1-problème1'".

Pour que l'application puisse se connecter à l'API jusqu'à maintenant elle trouve ses informations de connexions dans le fichier [Access.cs](#)

```
private Access()
{
    String authenticationString;
    try
    {
        authenticationString = "admin:adminpwd"; // Orange arrow points here
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}
```

Il faut donc mettre ces informations dans un fichier de configuration et appeler son contenu depuis Access.

Donc à l'image de l'application habilitation qu'il est recommandé de suivre, je crée une nouvelle section de configuration qui va contenir les informations pour se connecter dans le fichier **App.config**

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
    <add name="MediaTekDocuments.Properties.Settings.mEDIATEK86ConnectionString"
         connectionString="admin:adminpwd"
         />
  </connectionStrings>
  <startup>
```



Maintenant il va falloir appeler les informations contenu ici depuis le fichier Access. Par l'intermédiaire du fichier Settings.

J'ajoute une variable qui contient le chemin.

```
//ajout
private static readonly string connectionName = "MediaTekDocuments.Properties.Settings.mEDIATEK86ConnectionString";
```

Et j'entre les informations de connexions contenu. Via la méthode `GetConnectionStringByName()`

```
1 référence | hedi-k, Il y a 21 heures | 1 auteur, 3 modifications
private Access()
{
    String authenticationString;
    try
    {
        authenticationString = GetConnectionStringByName(connectionName);
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}
```

```
//Ajout
1 référence | hedi-k, Il y a 21 heures | 1 auteur, 1 modification
static string GetConnectionStringByName(string name)
{
    string returnValue = null;
    //Récupération des paramètres de la chaîne de connexion à partir de la configuration de l'application
    ConnectionStringSettings settings = ConfigurationManager.ConnectionStrings[name];
    if (settings != null)
        returnValue = settings.ConnectionString;
    return returnValue;
}
```

Une fois cela fait, je propose une résolution du problème sur GitHub. Donc je commit et push sur la seconde branche ma solution et ensuite je fais une « pull request » pour proposer ma solution.

### Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff com

The screenshot shows the GitHub interface for creating a pull request. At the top, it says "base: main" and "compare: 1-problème1". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below this, there's a title field containing "Solution" with an orange arrow pointing to it. The description field contains the text "log et mdp sont dans le fichier App.config comme dans l'application habilitation" with another orange arrow pointing to it. There are "Write" and "Preview" tabs, and a rich text editor toolbar above the description area. Below the description, it says "Markdown is supported" and "Paste, drop, or click to add files". On the right, there's a large green "Create pull request" button with an orange arrow pointing to it. At the bottom, there are links for "1 commit" and "2 files changed", each with an orange arrow pointing to it. A note at the bottom left says "Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)".

Ensuite il me reste à accepter (ou non) la solution proposer pour la faire fusionner avec la branche principal.

## Solution #2

The screenshot shows a GitHub pull request page for a repository named 'Atelier3'. The pull request is titled 'hedi-k wants to merge 1 commit into main from 1-problème1'. A warning message at the top states: '⚠ The head ref may contain hidden characters: "1-problème1\u00E8me1"'.

Below the warning, there are tabs for 'Conversation' (0), 'Commits' (1), 'Checks' (0), and 'Files changed' (2). A comment from 'hedi-k' is visible, stating: 'log et mdp sont dans le fichier App.config comme dans l'application habilitation'. Below this comment is a link to an issue titled 'Problème1 #1'.

A large modal window is open, titled 'Merge pull request'. It contains several items:

- 'Require approval from specific reviewers before merging': A note says 'Rulesets ensure specific people approve pull requests before they're merged.' with a 'Add rule' button.
- 'Continuous integration has not been set up': A note says 'GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.'
- 'This branch has no conflicts with the base branch': A note says 'Merging can be performed automatically.'

At the bottom of the modal, there is a green 'Merge pull request' button and a note: 'You can also open this in GitHub Desktop or view command line instructions.'

Below the modal, there is a placeholder for a comment: 'Add a comment'.

Une fois la solution fusionnée dans la branche principale, il ne me reste plus qu'à retourner sur cette branche avec mon IDE.

Et de mettre à jour ma branche principale avec les modifications effectuées.

The screenshot shows a PowerShell developer window. The command history is as follows:

```
PowerShell développeur
+ PowerShell développeur
=====
** Visual Studio 2019 Developer PowerShell v16.11.33
** Copyright (c) 2021 Microsoft Corporation
=====
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git branch
* 1-problème1
  main
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments>
```

```

PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments> git pull
Updating e2f96c3..afdfc90
Fast-forward
 MediaTekDocuments/App.config    |  5 +++
 MediaTekDocuments/dal/Access.cs | 19 ++++++-----
 2 files changed, 23 insertions(+), 1 deletion(-)
PS C:\Users\GT630M\Desktop\A3\MediaTekDocuments>

```

## Problème 2

Actuellement si on entre l'adresse de l'API on obtient son contenu.

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize

## Index of /rest\_mediatekdocuments

	Name	Last modified	Size	Description
	<a href="#">Parent Directory</a>		-	
	<a href="#">AccessBDD.php</a>	2024-03-02 15:06	19K	
	<a href="#">ConnexionPDO.php</a>	2024-02-13 10:58	2.0K	
	<a href="#">Controle.php</a>	2024-03-01 16:25	4.3K	
	<a href="#">htaccess.save</a>	2023-08-16 16:03	304	
	<a href="#">mediatek86.sql</a>	2023-08-16 16:03	15K	
	<a href="#">mediatekdocuments.php</a>	2024-02-28 19:49	1.5K	
	<a href="#">nbproject/</a>	2024-02-08 15:17	-	

Pour palier à cela il suffit d'ajouter dans le fichier .htaccess de l'API la règle associé à cette erreur.

```

E: > wamp64 > www > rest_mediatekdocuments > .htaccess
1 RewriteEngine on
2 RewriteRule ^([a-zA-Z]+)$ mediatekdocuments.php?table=$1
3 RewriteRule ^([a-zA-Z]+)/(.*)$ mediatekdocuments.php?table=$1&champs=$2
4 RewriteRule ^([a-zA-Z])/([a-zA-Z0-9]+)/(.*)$ mediatekdocuments.php?table=$1&id=$2&champs=$3
5 RewriteRule ^$ mediatekdocuments.php?error=404 ←
6
7
8

```

Cette nouvelle règle s'applique si `^$` cela correspond à une URL vide.

Si cela arrive, on redirige la requête sur le fichier `mediatekdocuments.php`.  
Et on définit le paramètre `error` à 404.

Donc dans l'API il ne reste qu'à créer la condition pour cette erreur.

```

// Contrôle de l'authentification
if (!isset($_SERVER['PHP_AUTH_USER']) || (isset($_SERVER['PHP_AUTH_USER']) &&
    !($_SERVER['PHP_AUTH_USER'] == 'admin' && ($_SERVER['PHP_AUTH_PW'] == 'adminpwd')))) {
    $controler->unauthorized();
} else {
    if(isset($_GET['error'])&& $_GET['error']==404){ ←
        echo ("La page n'existe pas");
        exit();
}

```

Si on a une erreur et que cette erreur correspond à 404, on retourne un message et on quitte l'API.

The screenshot shows a Postman interface. At the top, there's a header bar with 'localhost/rest\_mEDIATEkdocuments/' and a green 'GET' button. Below it is a table for 'Query Params' with one row for 'Key'. Underneath is a large empty area for the response. At the bottom, tabs for 'Body', 'Cookies', 'Headers (7)', and 'Test Results' are visible, along with buttons for 'Pretty', 'Raw', 'Preview', and 'Visualize'. The 'Preview' tab is selected, displaying the text 'La page n'existe pas'.

Une fois cette tâche réalisé, je met à jour mon kanban et passe à la prochaine.

The screenshot shows a Kanban board with three columns: 'To do', 'In progress', and 'Done'. The 'To do' column has a card with 'Draft' and 'Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 3 : intégrer des logs'. The 'In progress' column has a card with 'Draft' and 'Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 2 : contrôler la qualité'. The 'Done' column has a card with 'Draft' and 'Mission 5 : assurer la sécurité, la qualité et intégrer des logs Tâche 1 : corriger des problèmes de sécurité'.

## Tâche 2 : contrôler la qualité

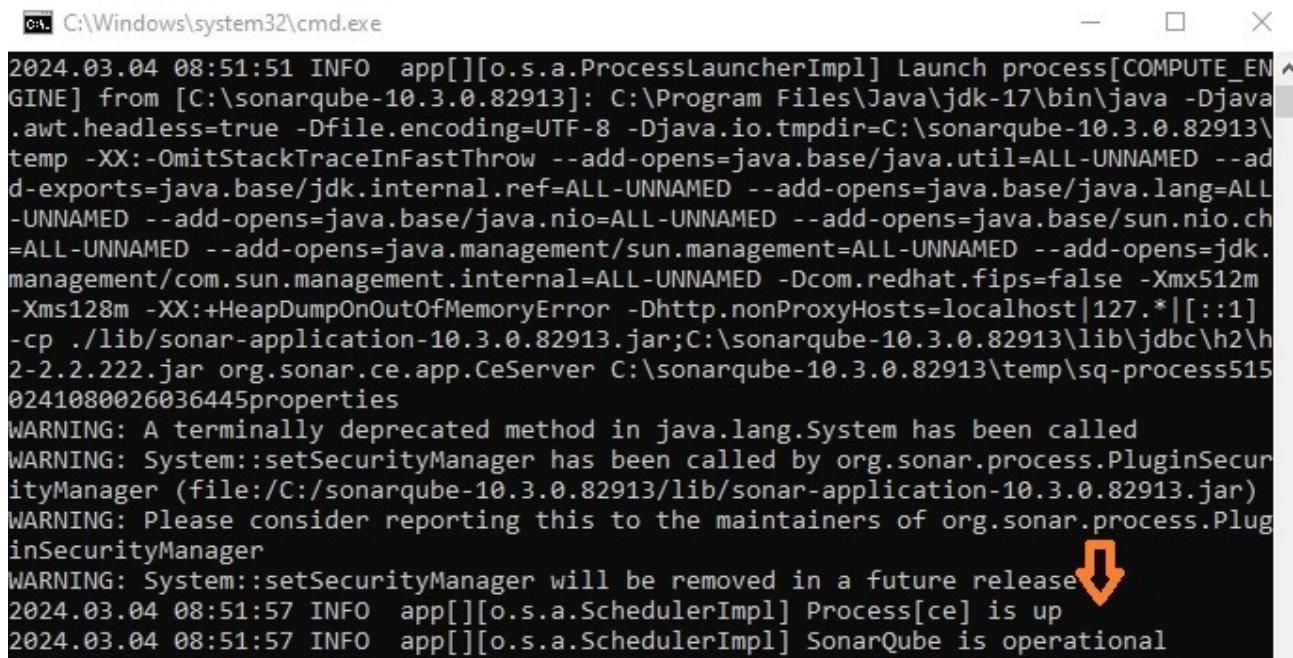
Tâches à réaliser

Contrôler que Sonarlint est configuré dans Visual Studio.

- Mettre en place les serveurs SonarQube et Jenkins.
- Faire le lien entre l'application C# et SonarQube via Jenkins pour l'intégration continue du suivi de qualité.
- Corriger les problèmes relevés par Sonarlint dans le code ajouté (excepté les problèmes qui ne doivent pas être corrigés, comme les noms des méthodes événementielles qui commencent par une minuscule).
- Contrôler les messages dans SonarQube

## Configuration SonarQube

Dans un premier temps je lance le serveur SonarQube.



```
C:\Windows\system32\cmd.exe
2024.03.04 08:51:51 INFO app[][o.s.a.ProcessLauncherImpl] Launch process[COMPUTE_ENGIN
GINE] from [C:\sonarqube-10.3.0.82913]: C:\Program Files\Java\jdk-17\bin\java -Djava
.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\sonarqube-10.3.0.82913\t
emp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --ad
d-exports=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.lang=ALL
-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch
=ALL-UNNAMED --add-opens=java.management/sun.management=ALL-UNNAMED --add-opens=jdk.
management/com.sun.management.internal=ALL-UNNAMED -Dcom.redhat.fips=false -Xmx512m
-Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.*|[::1]
-cp ./lib/sonar-application-10.3.0.82913.jar;C:\sonarqube-10.3.0.82913\lib\jdbc\h2\h
2-2.2.222.jar org.sonar.ce.app.CeServer C:\sonarqube-10.3.0.82913\temp\sq-process515
0241080026036445properties
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by org.sonar.process.PluginSecur
ityManager (file:/C:/sonarqube-10.3.0.82913/lib/sonar-application-10.3.0.82913.jar)
WARNING: Please consider reporting this to the maintainers of org.sonar.process.Plugin
SecurityManager
WARNING: System::setSecurityManager will be removed in a future release 
2024.03.04 08:51:57 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up
2024.03.04 08:51:57 INFO app[][o.s.a.SchedulerImpl] SonarQube is operational
```

Une fois le serveur lancé, je crée un nouveau projet que je vais lier à mon application.

## Create a local project

Project display name \*

 ✓

Up to 255 characters. Some scanners might override the value you provide.

Project key \*

 ✓

The project key is a unique identifier for your project. It may contain up to 41 characters. Allowed characters are alphanumeric, '-' (dash), '\_' (underscore), '.' (period) and at least one non-digit.

Main branch name \*

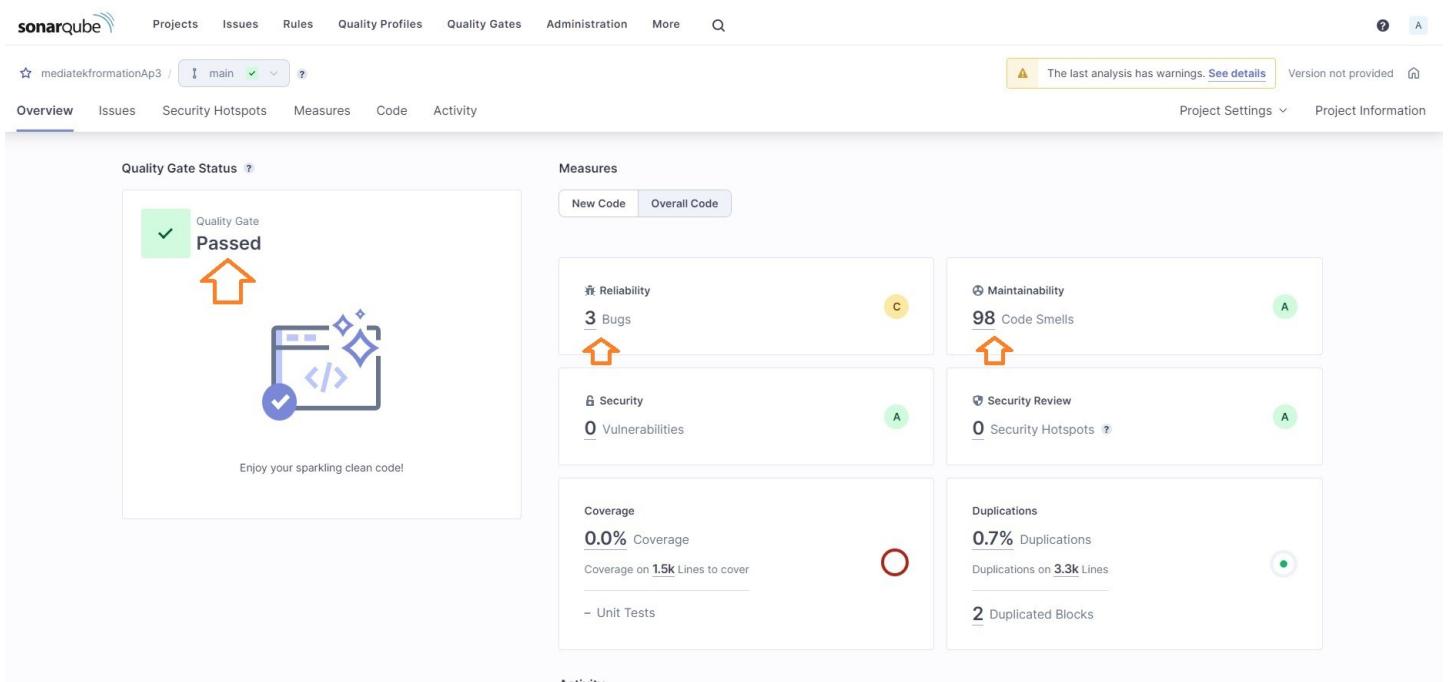
The name of your project's default branch [Learn More](#)

[Next](#)

Une fois le projet créé et configuré, j'exécute un scanne par SonarQube de l'application.

```
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AY4IkrHAHRLrB3E22Xif
INFO: Analysis total time: 11.327 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 12.765s
INFO: Final Memory: 21M/96M
INFO: -----
The SonarScanner CLI has finished
09:26:56.077 Post-processing succeeded.
```

A partir de maintenant j'ai accès au résultat du scan sur l'interface de SonarQube. Et la bonne nouvelle, mon projet est « validé » par sonarQube. Bien qu'il m'indique trois bugs et 98 smells qui correspondent à des défauts mineurs.



sonarQube Projects Issues Rules Quality Profiles Quality Gates Administration More Q

mediatekformationAp3 / main ✓ ?

The last analysis has warnings. See details Version not provided ⓘ

Overview Issues Security Hotspots Measures Code Activity Project Settings ⓘ Project Information

Quality Gate Status ?

Quality Gate Passed

Enjoy your sparkling clean code!

Measures

New Code Overall Code

Reliability	Maintainability
3 Bugs	98 Code Smells
House icon	House icon
Security	Security Review
0 Vulnerabilities	0 Security Hotspots ⓘ
House icon	House icon
C	A

Coverage	Duplications
0.0% Coverage	0.7% Duplications
Coverage on 1.5k Lines to cover	Duplications on 3.3k Lines
- Unit Tests	2 Duplicated Blocks
O	G

Activity

Pour la corrections des bugs comme ils ne sont qu'au nombre de trois et identiques, j'effectue la correction directement.

SonarQube m'indique qu'un de mes test ne fonctionne pas. Comme un entier ne peux pas être null, je vais le tester sur sa valeur pour corriger ce problème

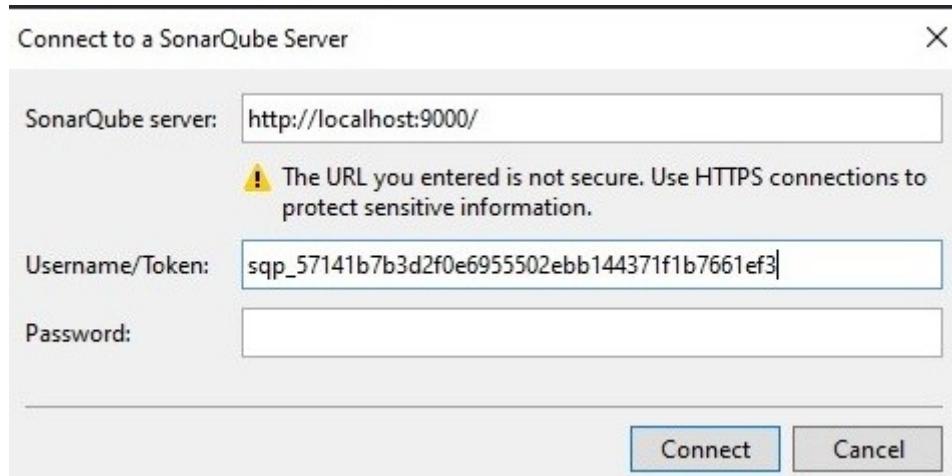
The screenshot shows the SonarQube interface with two highlighted issues:

- Intentionality issue**:  
Change this condition so that it does not always evaluate to 'True'. Some code paths are unreachable. Why is this an issue?  
cwe pitfall ... +  
Open Not assigned Reliability Bug Major 15min effort • 5 days ago
- Consistency issue**:  
Le résultat de l'expression est toujours 'true', car une valeur de type 'int' n'est jamais égale à 'null' du type 'int?' Why is this an issue?  
No tags +  
Open Not assigned Maintainability Code Smell Major ROSLYN 0min effort • 5 days ago

```
private void btnLivreExemplaireSupprimer_Click(object sender, EventArgs e)
{
    if (dgvLivreExemplaire.CurrentCell != null)
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //Récupère dans la variable les données de l'objet sélectionné dans la dgv
            DataGridViewRow row = dgvLivreExemplaire.SelectedRows[0];
            //DataBoundItem permet l'accès aux données de l'objet
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            if (exemplaire.Numero != null) ←
            {
                if (controller.SupprimerDocument(exemplaire))
                {
                    PresChargeDGVExemplaireLivre();
                }
            }
        }
    }
}
```

```
private void btnLivreExemplaireSupprimer_Click(object sender, EventArgs e)
{
    if (dgvLivreExemplaire.CurrentCell != null)
    {
        if (MessageBox.Show("Supprimer ?", "Confirmer", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //Récupère dans la variable les données de l'objet sélectionné dans la dgv
            DataGridViewRow row = dgvLivreExemplaire.SelectedRows[0];
            //DataBoundItem permet l'accès aux données de l'objet
            Exemplaire exemplaire = row.DataBoundItem as Exemplaire;
            if (exemplaire.Numero > -1) ←
            {
                if (controller.SupprimerDocument(exemplaire))
                {
                    PresChargeDGVExemplaireLivre();
                }
            }
        }
    }
}
```

Pour les codes smell, ils sont trop nombreux pour être corrigé depuis cette interface. Donc je lie SonarQube avec l'extension SonarLint du l'IDE.



Maintenant l'IDE va m'indiquer les erreurs de SonarQube. Pour les identifier elles commencent par un S. Il ne me reste qu'à les sélectionner et les corriger.

Exemple d'une variable oubliée.

Code	Description	Projet	Fichier	Li...	État de la su
S1075	Refactor your code not to use hardcoded absolute paths or URLs.	MediaTekDocuments	Access.cs	21	Actif
S1481	Remove the unused local variable 'jsonSupprimerDocument'.	MediaTekDocuments	Access.cs	246	Actif
S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	240	Actif
S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	252	Actif
S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	264	Actif
S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	284	Actif

Ou d'une condition qui peut être simplifiée.

```
//Supprime le document en paramètre
1 référence | hedi-k, il y a 2 jours | 1 auteur, 1 modification
public bool SupprimerDocument<T>(T unDocument)
{
    //Convertit en json l'objet en paramètre
    String jsonSupprimerDocument = JsonConvert.SerializeObject(unDocument);
    try
    {
        //L'id est nécessaire pour l'API en cas de suppression
        string id = "", Id="Id";
        switch (unDocument)
        {
            case Livre livre:
                id = livre.Id;
                break;
            case Dvd dvd:
                id = dvd.Id;
                break;
            case Revue revue:
                id = revue.Id;
                break;
            case CommandeDocument uneCmd:
                id = uneCmd.Id;
                break;
            case Abonnement unAbo:
                id = unAbo.Id;
                break;
            case Exemplaire exemplaire:
                id = exemplaire.Numero.ToString(); // C'est en INT donc faut le convertir en string
                Id = "Numero";
                break;
        }
        List<Object> liste = TraitementRecup<Object>(DELETE, typeof(T).Name.ToLower() + "/{" + Id + ":" + id + "}");
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

Code	Description	Projet	Fichier	Li...	État de la s
⚠ S1075	Refactor your code not to use hardcoded absolute paths or URLs.	MediaTekDocuments	Access.cs	21	Actif
⚠ S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	240	Actif
⚠ S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	252	Actif
⚠ S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	264	Actif
⚠ S1066	Merge this if statement with the enclosing one.	MediaTekDocuments	FrmAjout.cs	284	Actif

```

}
//Action du bouton Ajouter. Envoi un livre, un dvd ou une revue dans la bdd
1 référence | hedi-k, il y a 2 jours | 1 auteur, 8 modifications
private void btnAjouter_Click(object sender, EventArgs e)
{
    switch (quelEnvoi)
    {
        case "livre":
            Livre livre = SuperLivre();
            if (livre != null)
            {
                if (controller.EnvoiDocuments(livre)) // l'api return true si l'ajout a fonctionné
                {
                    MessageBox.Show("Livre ajouté");
                    Vide();
                }
            }
            break;
    }
}

```

```

}
//Action du bouton Ajouter. Envoi un livre, un dvd ou une revue dans la bdd
1 référence | hedi-k, il y a 2 jours | 1 auteur, 8 modifications
private void btnAjouter_Click(object sender, EventArgs e)
{
    switch (quelEnvoi) IDE0059 Assigment inutile d'une valeur à 'ex'
    {
        case "livre":
            Livre livre = SuperLivre();
            if (livre != null)
            {
                try
                {
                    controller.EnvoiDocuments(livre);
                }
                catch (Exception ex) { return null; }
                catch (Exception) { return null; }
                Vide();
            }
            break;
    }
}

```

Beaucoup ne sont pas de « vrais » erreurs mais plus des conventions de noms ou des variables que je garde pour des tests comme les ex dans mes try catch qu'il me demande de supprimer.

## CONFIGURATION JENKINS

Il est demandé de mettre en place jenkins pour l'intégration continue.

La configuration du serveur jenkins pour qu'il communique avec le serveur SonarQube repose sur des tokens que l'on crée dans sonarQube et donnent à Jenkins.

Il faut dans un premier temps créer un projet dans jenkins et lui associer les informations de SonarQube pour qu'ils puissent communiquer

The screenshot shows the Jenkins dashboard. On the left, there are links for 'Nouveau Item', 'Utilisateurs', 'Historique des constructions', 'Relations entre les builds', 'Vérifier les empreintes numériques', and 'Administrer Jenkins'. In the center, there is a table titled 'Nom du projet' with columns 'S', 'M', and 'Nom du projet'. Two projects are listed: 'Pipeline-AP3' and 'Pipeline-mediatekformation'. An orange arrow points to the 'Pipeline-AP3' row. At the bottom, there are dropdown menus for 'File d'attente des constructions' and 'Icône: S M L', and a link 'Étapes du build'.

### Étapes du build

The screenshot shows the configuration for a 'Execute a command line batch Windows' step. It includes a 'Commande' section with the command: `SonarScanner.MSBuild.exe begin /k:"MediaTekDocuments" /d:sonar.token="sqp_57141b7b3d2f0e6955502ebb144371f1b7661ef3"  
MSBuild.exe C:\Users\GT630M\Desktop\A3\MediaTekDocuments /t:Rebuild  
SonarScanner.MSBuild.exe end /d:sonar.token="sqp_57141b7b3d2f0e6955502ebb144371f1b7661ef3"`. There is an 'Avancé' button at the bottom left and 'Sauvegarder' and 'Appliquer' buttons at the bottom right.

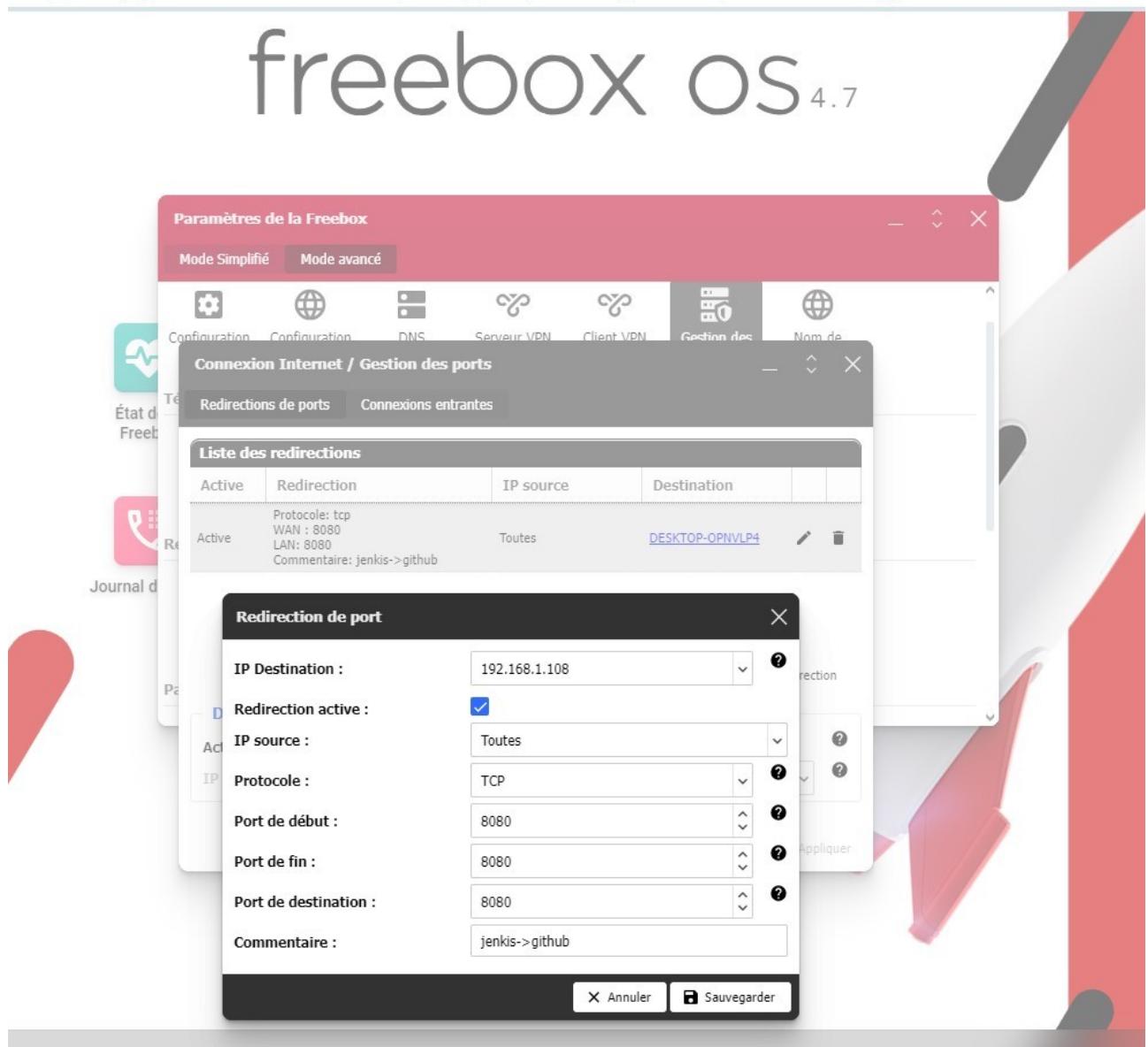
Une fois que ces deux serveurs communiquent, il faut faire un lien avec Github (webhook)

The screenshot shows the 'Ce qui déclenche le build' (What triggers the build) section of a Jenkins job configuration. It includes options for 'Construire après le build sur d'autres projets', 'Construire périodiquement', 'GitHub hook trigger for GITScm polling' (which is checked), and 'Scrutation de l'outil de gestion de version'. Below this is the 'Environnements de Build' (Build environments) section with the option 'Delete workspace before build starts'.

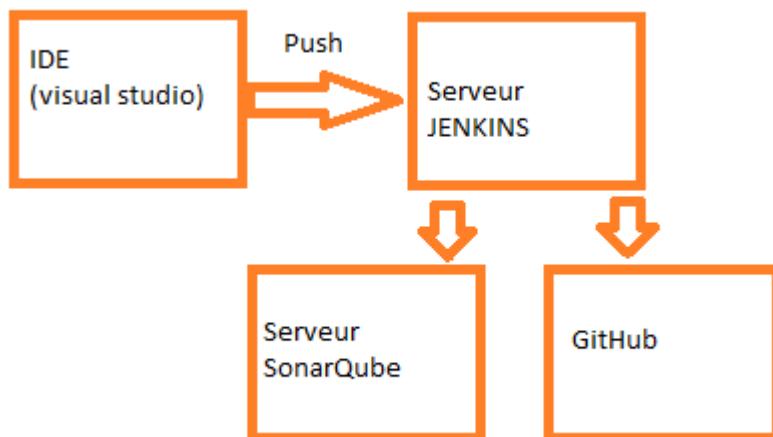
Ce lien va nécessiter que  
GitHube puisse

communiquer avec le serveur jenkins donc il faut autoriser les connexions externes (internet) à communiquer avec ce serveur.

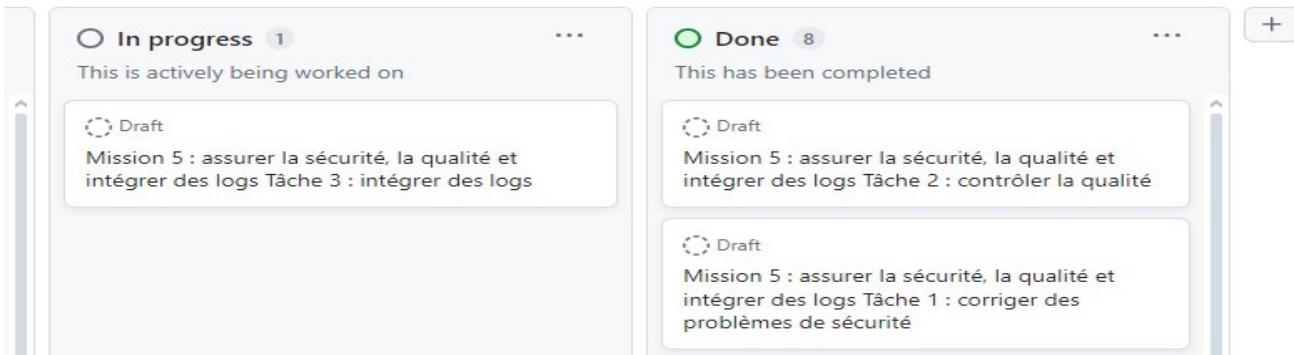
Pour moi qui suis chez free, l'ouverture d'un port dans ma box donne .



Le résultat est que toute correction dans Visual Studio, poussée dans GitHub, sera visible dans SonarQube après une demande de build dans Jenkins.



Une fois cette tâche terminé, je mets à jour mon kanban.

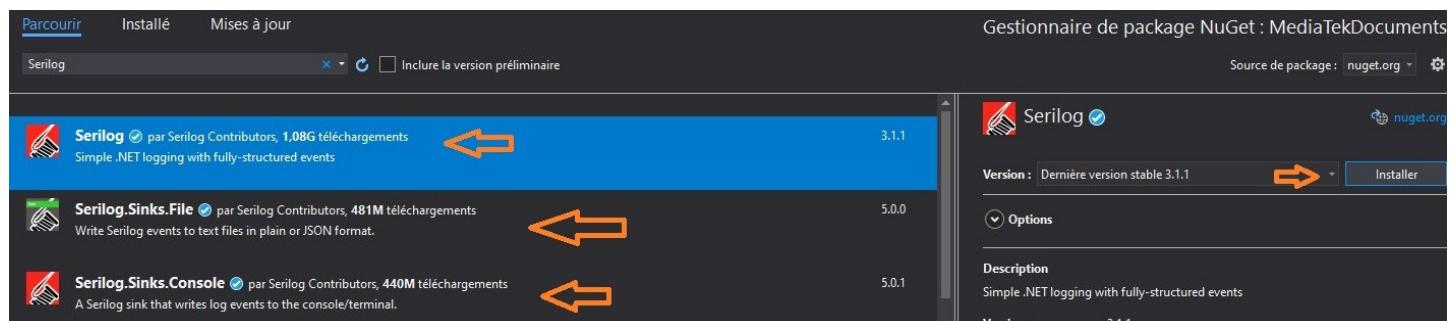


### Tâche 3 : intégrer des logs

#### Tâches à réaliser

Dans la classe Access, ajouter le code de configuration des logs et des logs au niveau de chaque affichage console (à enregistrer dans un fichier de logs)

Pour réaliser cette tâche, il faut dans un premier temps installer les NuGet ( NuGet est le gestionnaire de paquets de la plate forme de développement Microsoft .NET) nécessaire pour les logs.

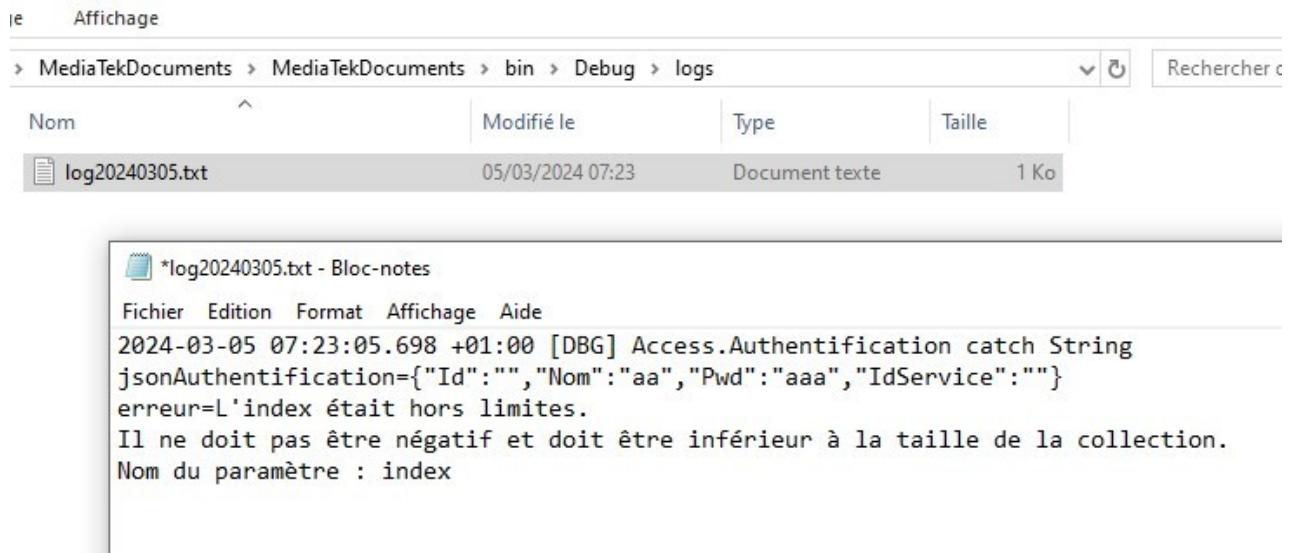


Ensuite Il faut intégrer dans le fichier access l'usage des logs.

```
1 référence | hedi-k, il y a 2 jours | 1 auteur, 3 modifications
private Access()
{
    String authenticationString;
    try
    {
        //Objet qui va gérer les logs
        Log.Logger = new LoggerConfiguration()
            //Si aucun "MinimumLevel" n'est fixé, il est par défaut à "Information".
            .MinimumLevel.Verbose()
            //Log dans la console
            .WriteTo.Console()
            //Log dans un fichier txt
            .WriteTo.File("logs/log.txt",
                //Interval entre chaque nouveaux fichier de log
                rollingInterval: RollingInterval.Day)
            .CreateLogger();

        authenticationString = GetConnectionStringByName(connectionName);
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        //On remplace les "simples" console.WriteLine par des logs
        Log.Fatal("Access.Access catch connectionString={0} erreur={1}", connectionName, e.Message);
        Environment.Exit(0);
    }
}
```

Le fait de remplacer les `Console.WriteLine()` par des `Log.niveau_d_erreur` va écrire le contenu du log dans la console et dans un fichier de log dédié. Exemple dans access si on entre des mauvaises informations de mot de passe, cela va me produire le fichier suivant :



Affichage

MediaTekDocuments > MediaTekDocuments > bin > Debug > logs

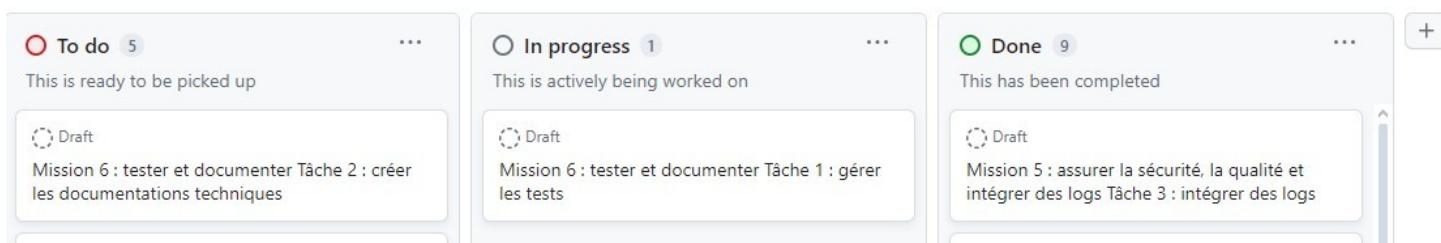
Nom	Modifié le	Type	Taille
log20240305.txt	05/03/2024 07:23	Document texte	1 Ko

```
*log20240305.txt - Bloc-notes
Fichier Edition Format Affichage Aide
2024-03-05 07:23:05.698 +01:00 [DBG] Access.Authentication catch String
jsonAuthentication={"Id":"","Nom":"aa","Pwd":"aaa","IdService":""}
erreur=L'index était hors limites.
Il ne doit pas être négatif et doit être inférieur à la taille de la collection.
Nom du paramètre : index
```

Et je procède comme cela pour tout les try catch du fichier access exemple

```
public bool CreerExemplaire(Exemplaire exemplaire)
{
    String jsonExemplaire = JsonConvert.SerializeObject(exemplaire, new CustomDateTimeConverter());
    try
    {
        // récupération soit d'une liste vide (requête ok) soit de null (erreur)
        List<Exemplaire> liste = TraitementRecup<Exemplaire>(POST, "exemplaire/" + jsonExemplaire);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Log.Debug("Access.CreaterExemplaire catch String jsonExemplaire={0} erreur={1}", jsonExemplaire, ex.Message);
    }
    return false;
}
```

Une fois cela fini, je mets à jour mon kanban et je passe à la mission suivante.



## Mission 6 : tester et documenter

### Tâche 1 : gérer les tests

Tâches à réaliser

Tests unitaires :

Écrire les tests unitaires sur les classes du package Model (en plus du test unitaire écrit précédemment).

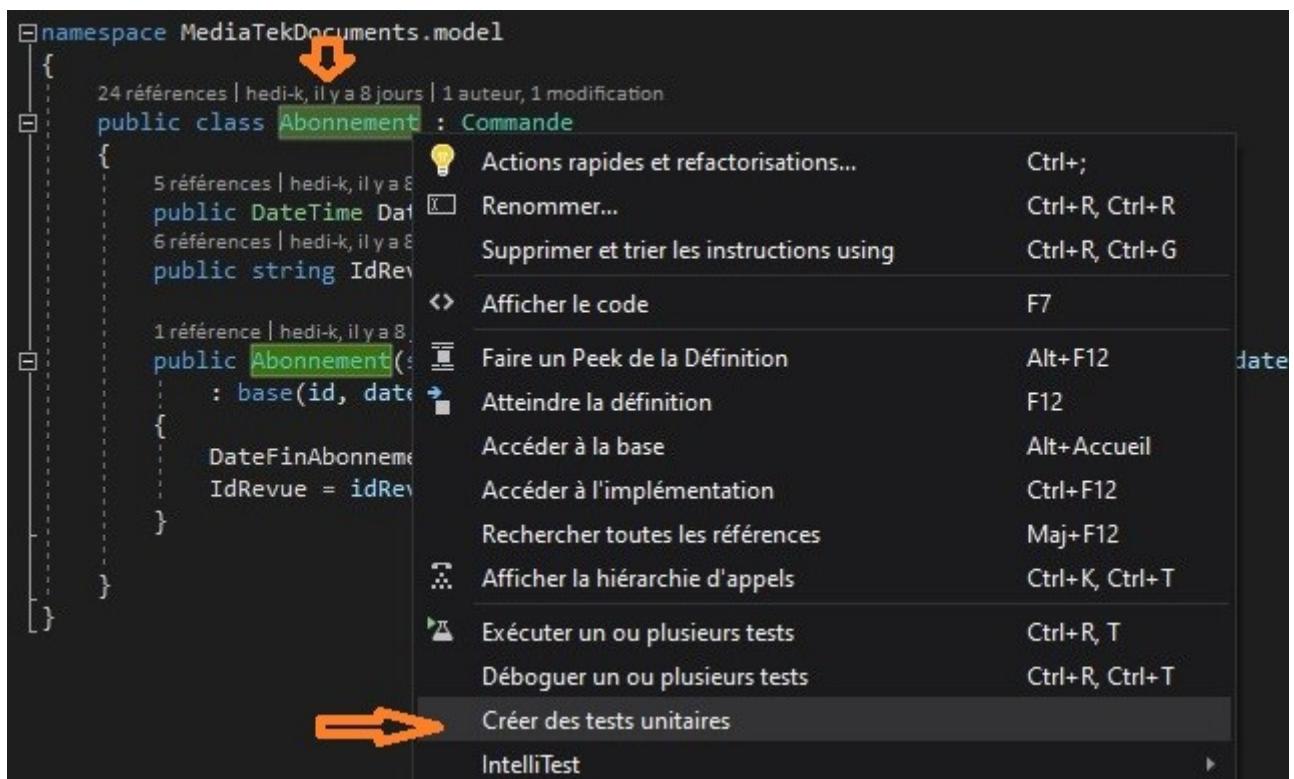
Tests fonctionnels :

Dans l'application C#, écrire les tests fonctionnels sur les recherches dans l'onglet des livres

Construire une collection de tests dans Postman pour contrôler les fonctionnalités de l'API d'accès à la BDD.

### Tests unitaires

Pour réaliser le test de la classe Abonnement je sélectionne la classe et fait générer la classe de test automatiquement.



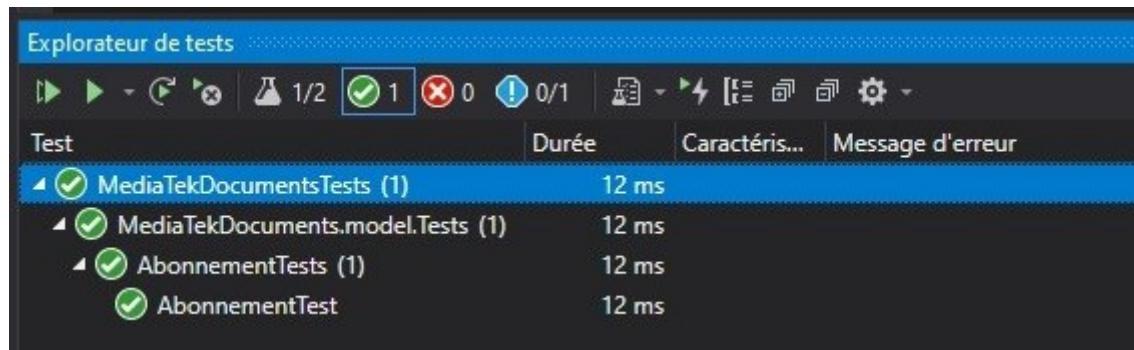
Ensuite je valorise un abonnement et je contrôle que chaque membre de la classe abonnement contient la valeur valorisée appropriée .

```

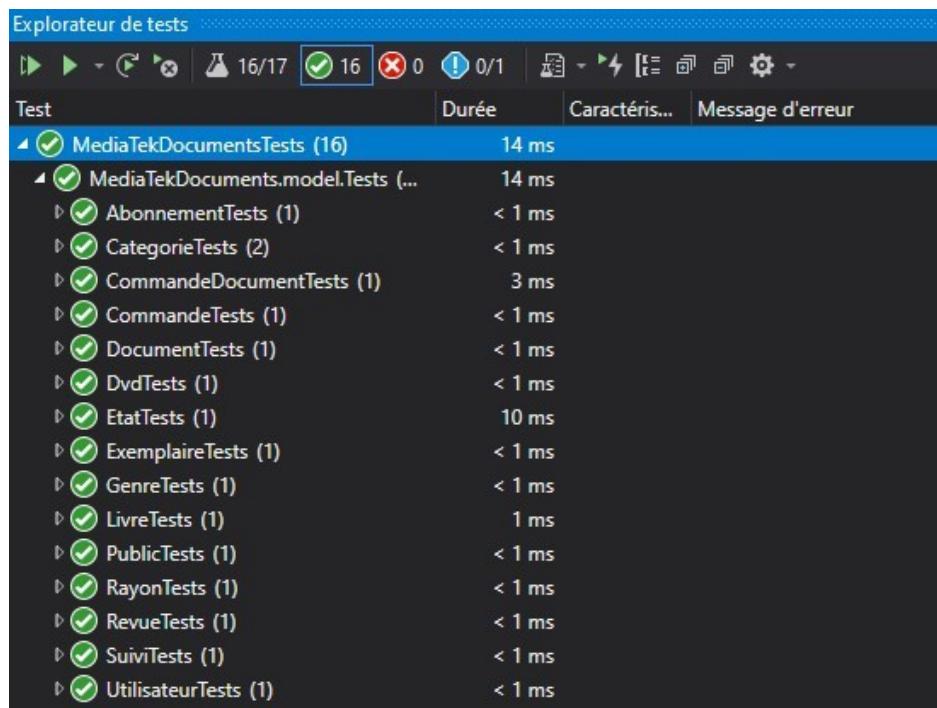
namespace MediaTekDocuments.model.Tests
{
    [TestClass()]
    0 références | 0 modification | 0 auteur, 0 modification
    public class AbonnementTests
    {
        private const string id = "01"; ← Valorise
        private static DateTime dateDebutAbo = new DateTime(2020, 01, 01);
        private const double montant = 2;
        private static DateTime dateFinAbo = new DateTime(2021, 01, 01);
        private const string idRevue = "02";
        private static readonly Abonnement abo = new Abonnement(id, dateDebutAbo, montant, dateFinAbo, idRevue);
        [TestMethod()]
        0|0 références | 0 modification | 0 auteur, 0 modification
        public void AbonnementTest()
        {
            Assert.AreEqual(id, abo.Id, "Devrait réussir => id valorisé"); ← contrôle
            Assert.AreEqual(dateDebutAbo, abo.DateCommande, "Devrait réussir => date début valorisée");
            Assert.AreEqual(montant, abo.Montant, "Devrait réussir => montant valorisé");
            Assert.AreEqual(dateFinAbo, abo.DateFinAbonnement, "Devrait réussir => date de fin valorisée");
            Assert.AreEqual(idRevue, abo.IdRevue, "Devrait réussir => id revue valorisé");
        }
    }
}

```

Et j'effectue le test. A noter quand générant la classe de test automatiquement, mon explorateur de test ne bug pas contrairement au test sur les dates d'abonnement que j'ai généré manuellement précédemment.

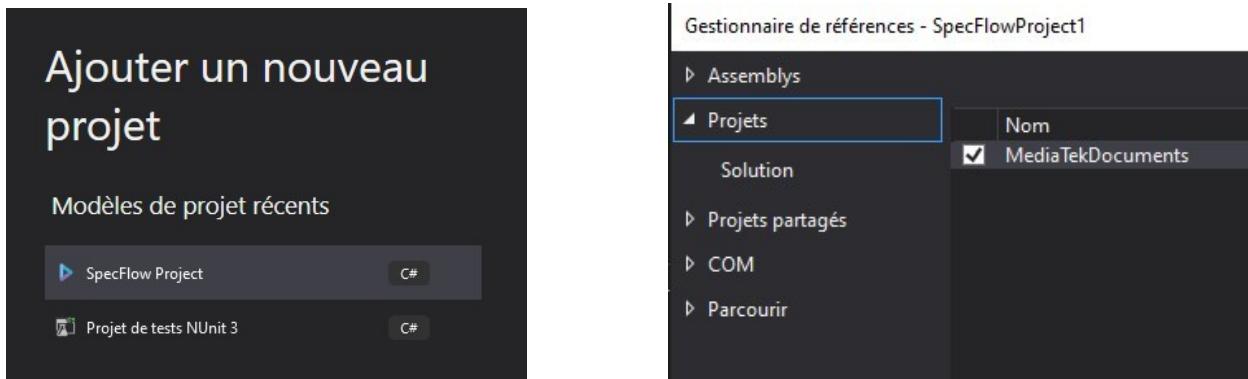


J'effectue cela pour toutes les classes du model, à l'exception de la classe LivreDvd qui est une classe abstraite donc elle ne peut pas être instanciée. Les résultats détaillés se trouvent dans le plan tests. Qui est en annexe dans le depot GitHub



## Tests fonctionnels

Pour réaliser les tests fonctionnel j'utilise SpecFlow. Il faut ajouter un nouveau projet SpecFlow et le lié à l'application.



Après une configuration et l'ajout d'un certains nombre de paramètres on peut utiliser SpecFlow.

Son fonctionnement repose sur des scénarios écrit en langage Gherkin qui seront codés et exécutés en C#.

Exemple pour un scénario de recherche de titre.

Je saisie les paramètres voulu (la recherche du titre )et attendu (le titre trouvé) dans Features.

```
Feature: recherche par titre
  Scenario: recherche par titre
    Given je saisie "La planète des singes"
    Then Il doit apparaître dans le titre des infos détaillé "La planète des singes"
```

Ensuite dans Steps j'écris le code lié à ce scenario.

```
[Given(@"je saisie ""([^\"]*)""")]
0 références | 0 modification | 0 auteur, 0 modification
public void GivenJeSaisieLeTitreLaPlaneteDesSingesDansTxbLivresTitreRecherche(string valeur)
{
  TabControl tabOngletsApplication = (TabControl)frmMediatek.Controls["tabOngletsApplication"];
  frmMediatek.Visible = true;
  tabOngletsApplication.SelectedTab = (TabPage)tabOngletsApplication.Controls["tabLivres"];
  TextBox txbLivresTitreRecherche = (TextBox)frmMediatek.Controls["tabOngletsApplication"].Controls["tabLivres"].Co
  txbLivresTitreRecherche.Text = valeur;
}
```

L'image est tronqué pour être lisible, la ligne qui manque est :

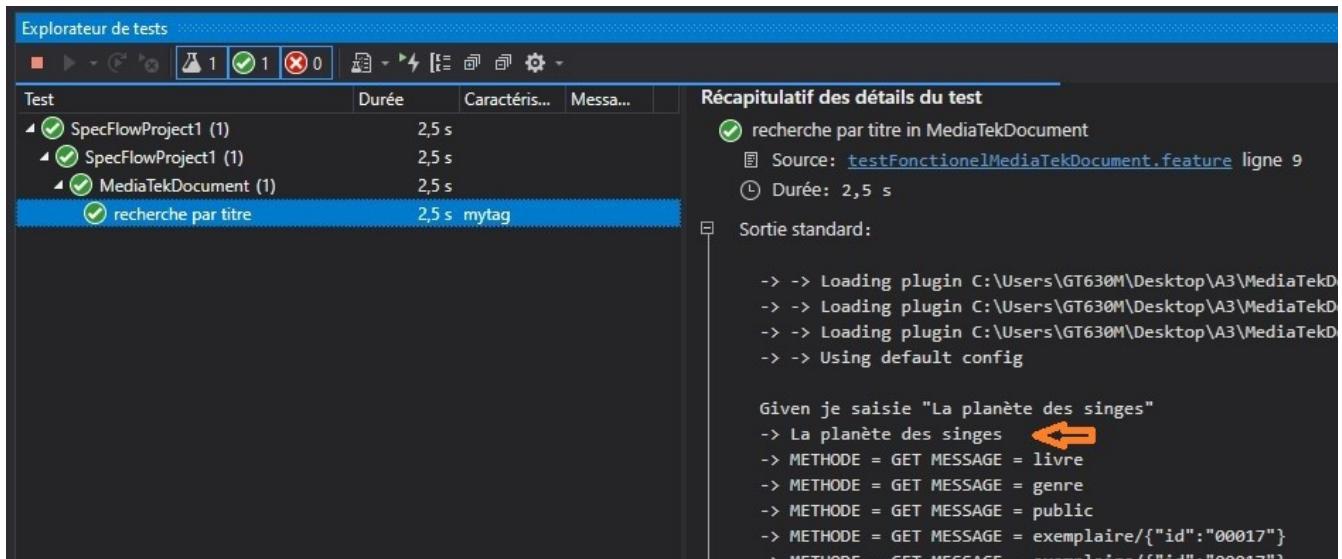
```
TextBox txbLivresTitreRecherche =
(TextBox)frmMediatek.Controls["tabOngletsApplication"].Controls["tabLivres"].Controls["grpLi
vresRecherche"].Controls["txbLivresTitreRecherche"];
```

Le principe repose sur la sélection des objets graphique en précisant toute leur arborescence. Ici pour avoir accès à la textBox de recherche de titre, il faut préciser qu'elle appartient à un groupBox qui lui même appartient à un onglet etc..

```
[Then(@"Il doit apparaître dans le titre des infos détaillé ""([^\"]*)""")]
0 références | 0 modification | 0 auteur, 0 modification
public void ThenIlDoitApparaitreDansLeTitreDesInfosDetailleTxbLivresTitre(string resultat)
{
    TextBox txbLivresTitre = (TextBox)frmMediatek.Controls["tabOngletsApplication"].Controls["tabLivre"];
    Assert.That(txbLivresTitre.Text, Is.EqualTo(resultat));
}
```

Après avoir écrit un titre, je suis sensé trouvé ce titre dans la textBox des informations détaillé et ici je compare le résultat avec un assert. Le choix de cet Assert.that() n'est pas le plus pertinent mais pour des raisons de compatibilités avec ma configuration, je ne peux pas en utiliser un autre.

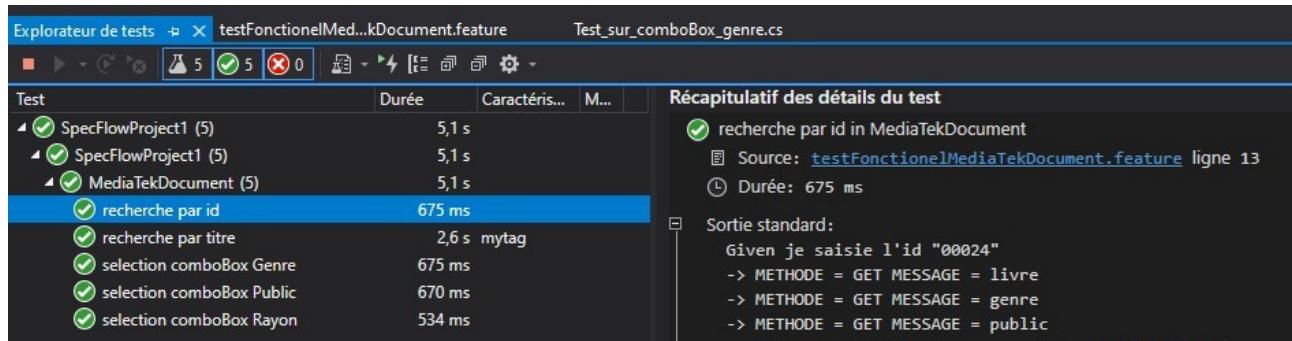
Ensuite on effectue le test et on récupère le resultat dans l explorateur de test.



```
-> METHODE = GET MESSAGE = exemplaire {"id": "00017"} ←
-> done: MediaTekDocumentSteps.GivenJeSaisieLeTitreLaPlaneteDesSingesDansTxbLivresTitreRecherche("La planète des si...") (2,2s)

Then Il doit apparaître dans le titre des infos détaillé "La planète des singes"
-> done: MediaTekDocumentSteps.ThenIlDoitApparaitreDansLeTitreDesInfosDetailleTxbLivresTitre("La planète des si...") (0,0s) ←
```

Je procède sur ce principe pour les autres tests



Les résultats détaillés se trouvent dans le plan tests.

Une fois cela fini, je mets à jour mon kanban et je passe à la mission suivante.

### Tests sur Postman

Pour effectuer les test sur Postman, il me suffit d'écrire les requêtes manuellement envoyé par l'application à l'API et de lire le résultat attendu.

Exemple pour la recherche des livres, un première exemple avec une authentification incorrecte donc qui me retourne une erreur

The screenshot shows a POST request to `localhost/rest_mEDIATEkdocuments/livre`. The Authorization tab is selected, showing "Basic Auth" type with "admin" for Username and "a" for Password. The response body is a JSON object:

```
1 {  
2   "code": 401,  
3   "message": "authentification incorrecte",  
4   "result": ""  
5 }
```

Et ensuite la même requête avec les bonnes informations de connexion

The screenshot shows a POST request to `localhost/rest_mEDIATEkdocuments/livre`. The Authorization tab is selected, showing "Basic Auth" type with "admin" for Username and "adminpwd" for Password. The response body is a JSON object:

```
1 {  
2   "code": 200,  
3   "message": "OK",  
4   "result": [  
5     {  
6       "id": "00102",  
7       "ISBN": "1111111111111",  
8       "author": "a"  
9     }  
10  ]  
11 }
```

Je fais la même chose pour tous mes GET, les résultats détaillés se trouvent dans le plan test.

Le plan test se trouve sur le dépôt GitHub en annexe

Une fois cela fait je met à jour mon kanban

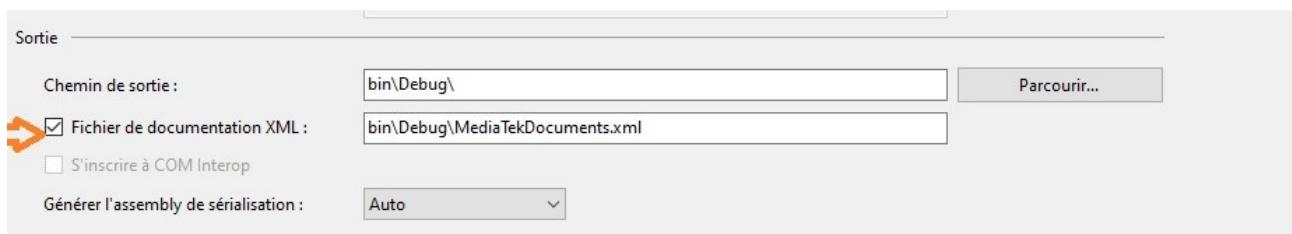


## Tâche 2 : créer les documentations techniques

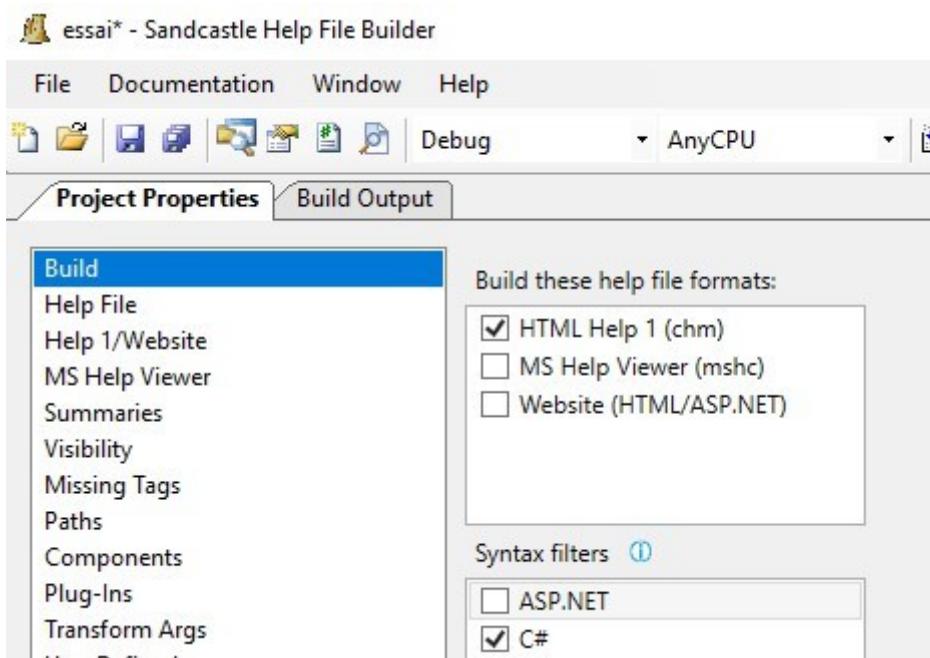
Tâches à réaliser

- Contrôler, dans chaque application, que les commentaires normalisés sont bien tous ajoutés et corrects.
- Générer les documentations techniques des 2 applications : sous format HTML pour l'API et sous format chm (HTML Help) pour l'application C#.
- Transférer les documentations dans les dépôts.

Pour l'application C# je convertis tout mes commentaires en commentaires normalisés  
Ensuite je crée un fichier xml de mon projet via les propriétés du projet dans l'IDE



Et via le logiciel sandCastle je convertis ce fichier xml en chm.



Ce qui me donne ma documentation technique. Elle se trouve dans le dépôt GitHub.

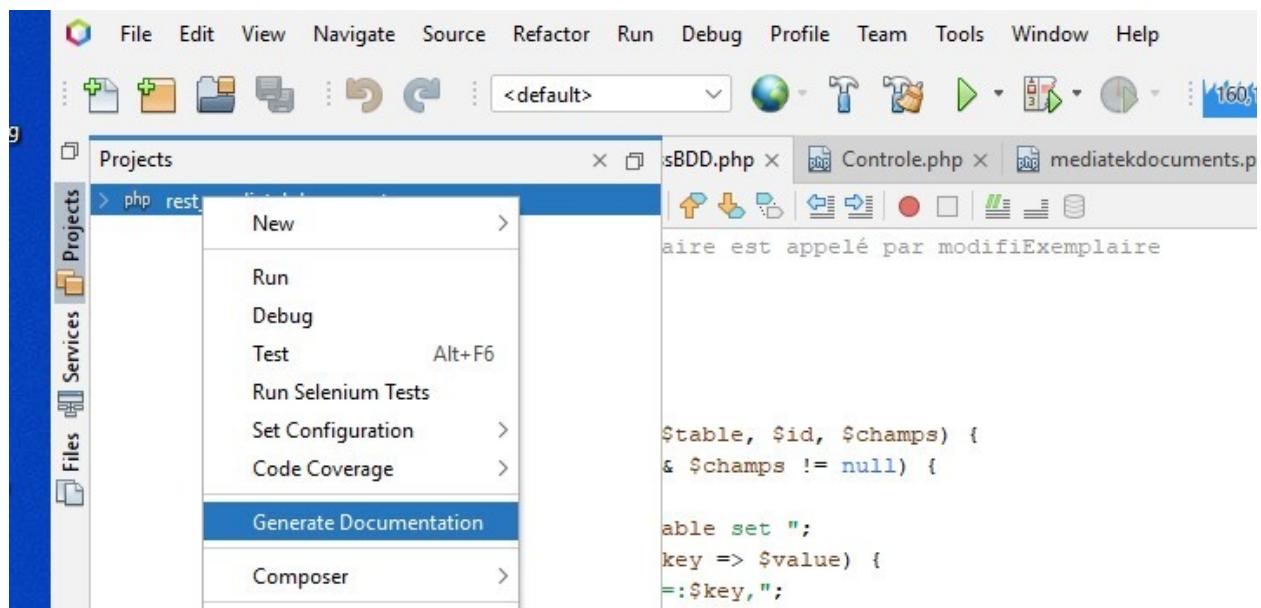
The screenshot shows a Java API documentation interface. On the left, there's a navigation bar with links to 'Sommaire', 'Index', and 'Rechercher'. Below it is a tree view of package structure:

- MediaTek Documents
- MediaTek Documents.controller
- MediaTek Documents.dal
- MediaTek Documents.manager
- MediaTek Documents.model
- MediaTek Documents.view
  - FrmAjout Classe
  - FrmAuthentification Classe
  - FmMediatek Classe
    - FmMediatek Constructeur
    - FmMediatek Propriétés
  - FmMediatek Méthodes
    - AbonnementsSurLaIn Méthode
    - AccesReceptionExemplaireGroupBox Méthode
    - AfficheCommandeDvdInfos Méthode
    - AfficheCommandeLivreInfos Méthode
    - AfficheCommandeRevueInfos Méthode

To the right of the tree view, there's a large text area containing detailed documentation for the selected class or method. It includes sections like 'Espace de nom:', 'Assembly:', and 'Syntaxe' (with C# code examples). The 'Syntaxe' section shows the following C# code:

```
C#
private void btnAjout(
    Object sender,
    EventArgs e
)
```

Pour la documentation de l'API j'utilise l'IDE NetBean qui me permet de la générer après une rapide configuration.



## rest\_mediatekdocuments

### Packages

Application

### Reports

Deprecated

Errors

Markers

### Indices

Files

## Documentation

### Table of Contents

#### Packages

P Application

#### Classes

C AccessBDD

Classe de construction des requêtes SQL à envoyer à la BDD

C ConnexionPDO

Classe de connexion et d'exécution des requêtes dans une BDD MySQL

C Controle

Contrôleur : reçoit et traite les demandes du point d'entrée

les documentations techniques se trouvent sur le dépôt GitHub du projet.

### Tâche 3 : créer la documentation utilisateur en vidéo

Tâches à réaliser

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

Pour se faire j'utilise le logiciel OBS Studio, cette documentation est téléchargeable depuis le GitHub du projet.

Une fois fait, je mets à jour mon kanban et passe à la prochaine mission.



### Mission 7 : déployer et gérer les sauvegardes de données

#### Tâche 1 : déployer le projet

Tâches à réaliser

## Mettre en ligne l'API :

Déployer l'API et la BDD. Pour cela, consultez l'article "API en ligne" sur le wiki de rest\_mEDIATEkdocuments, directement accessible avec ce lien :

[https://github.com/CNED-SLAM/rest\\_mEDIATEkdocuments/wiki/API-en-ligne](https://github.com/CNED-SLAM/rest_mEDIATEkdocuments/wiki/API-en-ligne).

Tester l'API avec Postman.

Créer un installateur pour l'application C# :

Modifier le code pour l'accès à l'API distante.

Créer l'installateur, tester son installation et l'utilisation de l'application installée.

L'envoyer vers le dépôt

## En ligne

Pour déployer mon API en ligne j'utilise un raspberry pi3 sur le quel j'installe un serveur ubuntu.

Une fois le serveur mis a jour, j'active openSSH installé par défaut sur ubuntu.

sudo systemctl start ssh

Ensuite je récupère mon adresse ip local

ip a

Avec cette information j'ouvre l'accès a cette adresse ip sur le port 22 (port par défaut pour ssh) dans ma box internet pour que je puisse communiquer de l'extérieure.

The screenshot shows the 'Connexion Internet / Gestion des ports' (Internet Connection / Port Management) window. On the left, there are configuration fields for 'IP Destination', 'Redirection active', 'IP source', 'Protocole' (set to TCP), 'Port de début' (set to 0), 'Port de fin' (set to 0), 'Port de destination' (set to 0), and 'Commentaire'. On the right, a table titled 'Liste des redirections' (List of redirections) displays two active rules:

Active	Redirection	IP source	Destination
Active	Protocole: tcp WAN : 8080 LAN: 8080 Commentaire: jenkins->github	Toutes	<a href="#">DESKTOP-OPNVLPI</a>
Active	Protocole: tcp WAN : 22 LAN: 22 Commentaire: API AP3	Toutes	<a href="#">ubuntu</a>

Buttons at the bottom right include 'Rafraîchir' (Refresh) and 'Ajouter une redirection' (Add a redirection).

Depuis mon pc, j'utilise putty pour accéder à mon serveur à distance. J'entre mon Ip avec le port 22 et j'arrive sur la page de log du serveur.

Il me suffit d'entrer mes informations de connexions configurés pendant l'installation d'ubuntu.

The screenshot shows the PuTTY Configuration window and a terminal session on a raspberry pi3 running Ubuntu. The PuTTY window shows the 'Session' tab with 'Host Name (or IP address)' set to '82.66.130.5' and 'Port' set to '22'. The terminal session shows system information:

```
pi3@ubuntu: ~
System information as of Sat Mar 9 17:12:59 CET 2024
System load:          0.08
Usage of /:           3.9% of 57.98GB
Memory usage:         28%
Swap usage:           0%
Temperature:          32.7 °C
Processes:            130
Users logged in:     1
IPv4 address for eth0: 192.168.1.147
IPv6 address for eth0: 2a01:e0a:lea:cf70:ba27:ebff:fe58:bb9e
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

0 updates can be applied immediately.

*** System restart required ***
Last login: Sat Mar 9 16:26:50 2024
pi3@ubuntu:~$
```

Depuis la console Putty j'installe LAMP qui est l'équivalent de wamp mais pour linux.  
sudo apt install apache2 php libapache2-mod-php mysql-server php-mysql

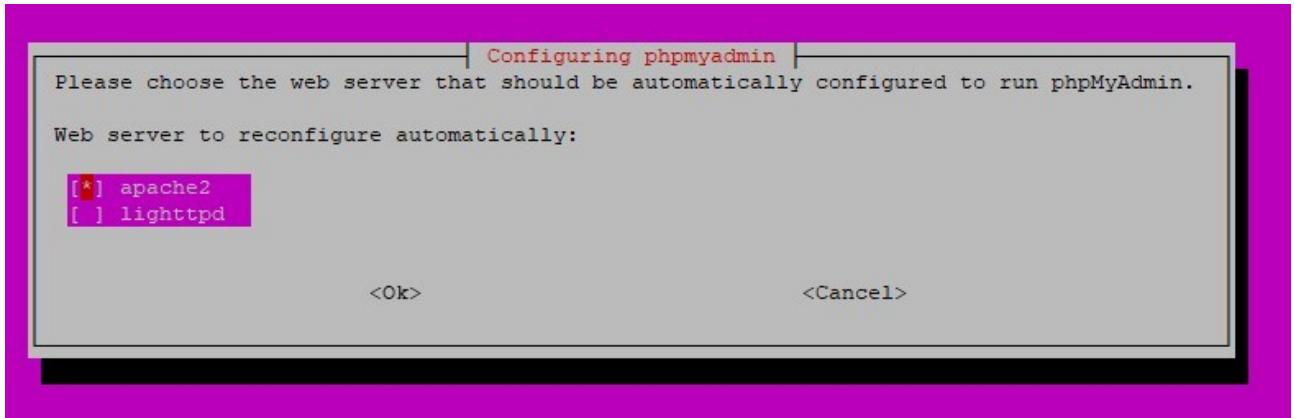
J'ouvre le port 80 qui est nécessaire pour la suite.  
Et j'essaie de communiquer avec mon serveur depuis mon navigateur web avec mon adresse IP



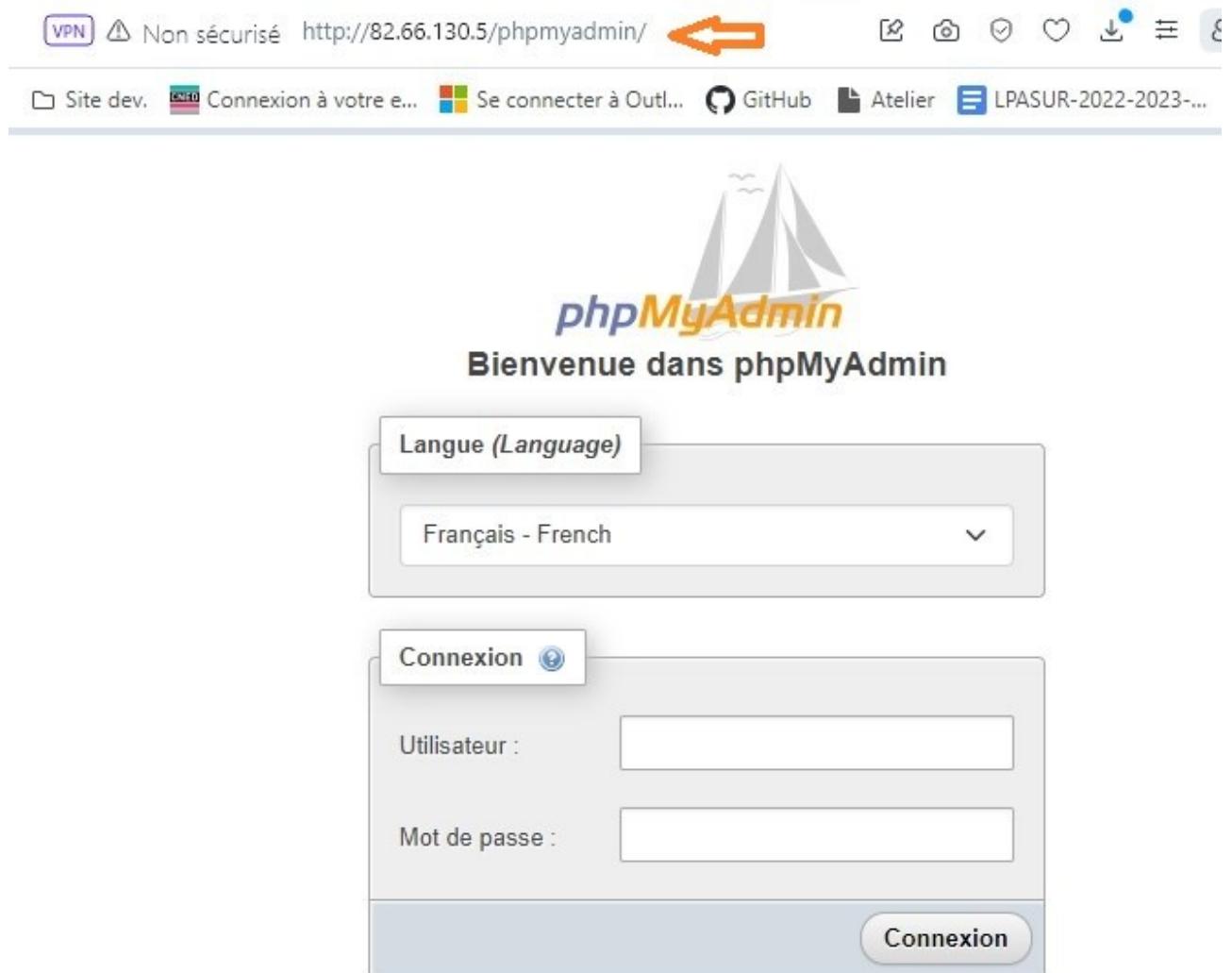
Une fois que je constate que cela fonctionne, j'installe phpMyAdmin.

sudo apt install phpmyadmin

J'associe bien le serveur Apach2 avec phpMyAdmin.



A partir de maintenant j'ai accès a phpMyAdmin depuis l'extérieur, il ne me reste plus qu'à configurer la base de données.



Je crée un utilisateur et la base de donnée associé.

```
sudo mysql
CREATE DATABASE mediatek86;
CREATE USER 'hedi'@'localhost' IDENTIFIED BY '*****';
GRANT ALL PRIVILEGES ON mediatek86.* TO 'hedi'@'localhost';
FLUSH PRIVILEGES;
QUIT;
```

Pour importer mon scripte SQL cette commande est obligatoire !

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Elle est utilisée dans MySQL pour autoriser l'enregistrement (logging) de fonctions créées par des utilisateurs dans les journaux binaires. Concrètement sans cela le script n'importe pas toutes les tables et trigger.

Une fois tout configuré et fonctionnel, j'importe ma base de donnée local sur mon serveur en ligne. Je modifie le fichier [Access.bdd](#) de l'API pour qu'il contienne les informations de connexion de cette base de donnée.

Ensuite avec FileZilla j'importe mon API dans un dossier dédié du serveur.

```

Site distant : /home/pi3/www/rest_mEDIATEKdocuments
[?] / 
    [?] home 
        [?] pi3 
            [?] .cache 
class AccessBDD { 

    public $login = "hedi"; 
    public $mdp = "mediatek86bdd"; 
    public $bd = "mediatek86"; 
    public $serveur = "127.0.0.1"; 
    public $port = "3306"; 
    public $conn = null; 
}

```

Pour la configuration d'Acpache2 je modifie  
apache2/sites-available/000-default.conf  
pour le lien symbolique

```

ServerAdmin webmaster@localhost
DocumentRoot /home/pi3/www
# DocumentRoot /var/www/html

```

apache2/apache2.conf  
pour le « vrai » chemin, autoriser le fichier .htaccess et tout le monde à accéder.

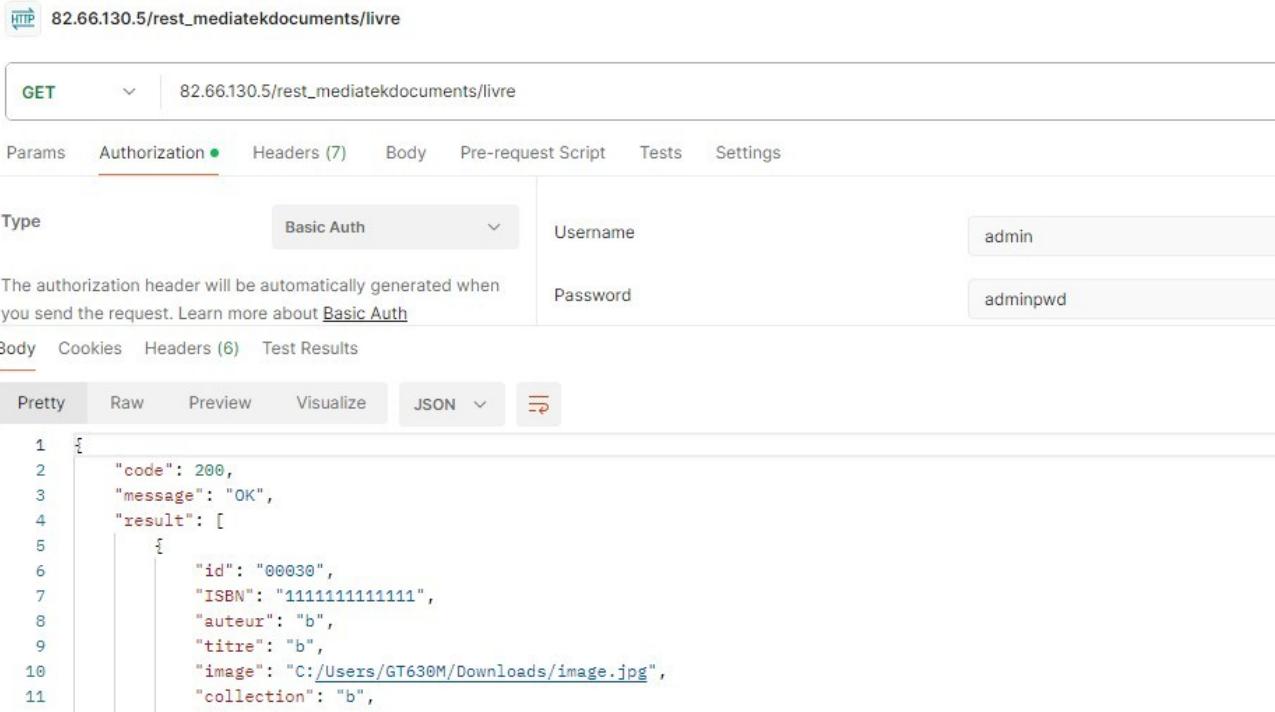
```

<Directory /home/pi3/www>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
<Directory /var/>

```

Et il faut aussi activer le module rewrite dans apache pour le fichier .htaccess.  
 sudo a2enmod rewrite

Ensuite je fais un essai sur postman pour m'assurer que tout fonctionne.



The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** 82.66.130.5/rest\_mEDIATEkdocuments/livre
- Headers:** Authorization (Basic Auth) - Username: admin, Password: adminpwd
- Body:** (Empty)
- JSON Response:**

```

1  {
2      "code": 200,
3      "message": "OK",
4      "result": [
5          {
6              "id": "00030",
7              "ISBN": "1111111111111",
8              "auteur": "b",
9              "titre": "b",
10             "image": "C:/Users/GT630M/Downloads/image.jpg",
11             "collection": "b",
...         }
...
]

```

Et dans le code de l'application je change l'adresse de l'API elle n'est plus local maintenant.

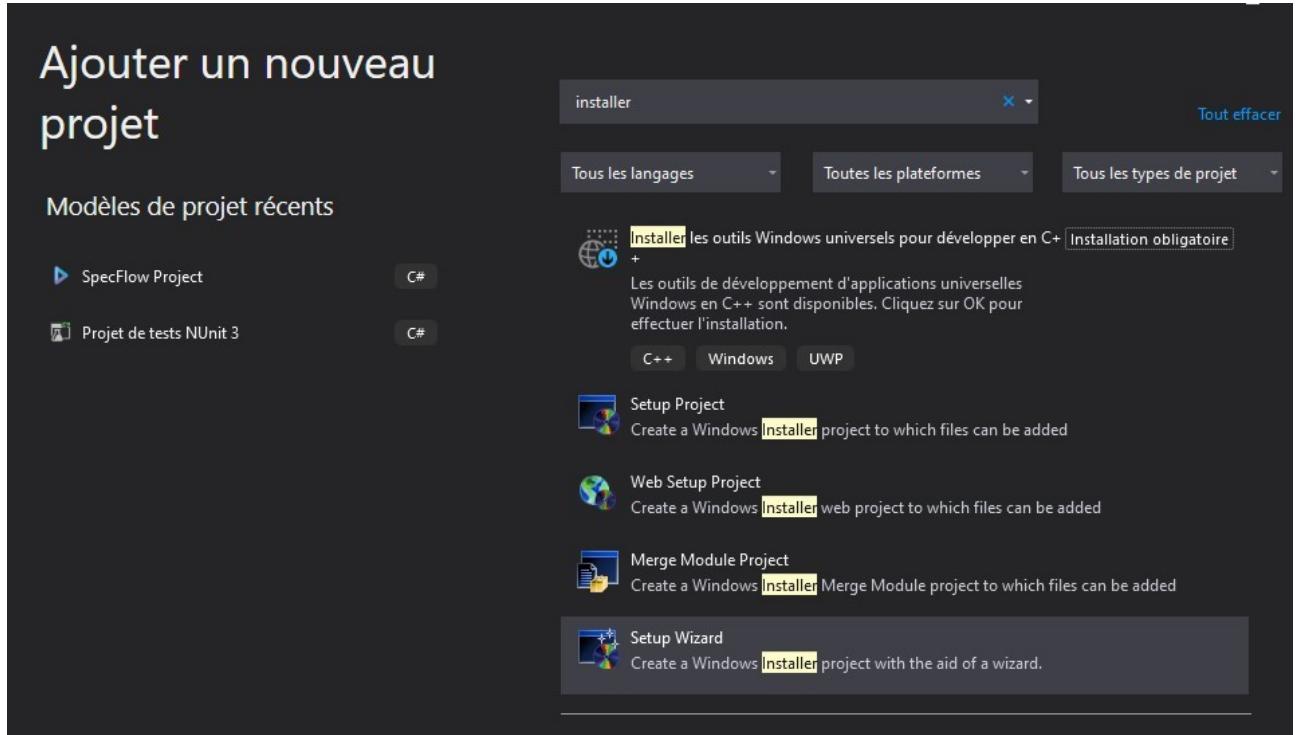
```

differences | newer | older | 1 job | 2 updates, 24 modifications
public class Access
{
    /// <summary>
    /// adresse de l'API
    /// </summary>
    private static readonly string uriApi = "http://82.66.130.5/rest_mEDIATEkdocuments/";
    /// <summary>
    /// instance unique de la classe
    /// </summary>
    ...
}

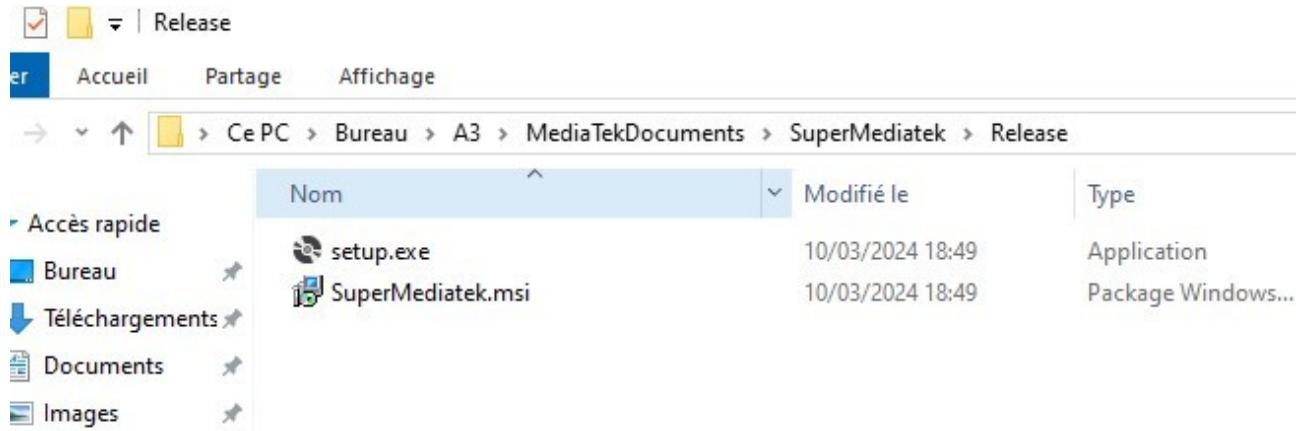
```

Afin de créer un installateur j'utilise l'extension Microsoft Visual Studio Installer Project.

Ensuite je crée un nouveau projet Setup Wizard



Après avoir configurer je généré le fichier.msi qui permet une installation.



## Tâche 2 : gérer les sauvegardes des données

Tâche à réaliser.

Une sauvegarde journalière automatisée doit être programmée pour la BDD  
La restauration pourra se faire manuellement, en exécutant le script de sauvegarde.

Pour réaliser cette tâche je crée un scripte qui va automatiser la sauvegarde de la bdd.

```
GNU nano 4.8
#!/bin/sh
#mémoriser la date dans une variable (date qui va permettre de construire le nom du fichier)
DATE=$(date -I)
LOGIN_COMPTE="pi3"
LOGIN_USER_BDD="hedi"
PWD_USER_BDD="mediatek86bdd"
NAME_BDD="mediatek86"
#Pour supprimer un éventuel précédent fichier commençant par "bdd" dans le dossier de sauvegarde,
# afin de ne garder qu'un fichier
find /home/$LOGIN_COMPTE/savebdd/bdd* -mtime -1 -exec rm {} \;
#Pour récupérer le script sql de la bdd et l'enregistrer dans un fichier zippé portant le nom
# de "bddbackup_" suivi de la date et l'extension
mysqldump -u $LOGIN_USER_BDD -p$PWD_USER_BDD --databases $NAME_BDD --single-transaction | gzip > /home/$
```

Ensuite je crée une tâche cron qui exécute ce scripte quotidiennement.

```
#sauvegarde de la bdd tout les jours
0 0 * * * /home/pi3/savebdd/scriptSaveBdd.sh
```

Projet fini, je mets a jour mon kaban !

## Bilan

J'ai « perdu » énormément de temps en réalisant les divers tâches pour les missions 1,2,3 car au fil du temps je me suis rendu compte qu'il est possible de factoriser des fonctions et donc je devais reprendre tout ce que j'avais fait depuis le début.

Les problèmes rencontrés pour faire fonctionner les tests sont les plus frustrant car ils sont le fait de compatibilités entre les différentes versions de l'ide, framework et autres outils. J'ai du essayer beaucoup de versions différents pour arriver à les faire fonctionner.

Je suis content de pouvoir réaliser une application, son API et sa BDD et d'avoir le tout hébergé en ligne par mes propres moyens !