2025

Mediatek Document

BTS SIO SLAM
ESTELLE FRIEDT

TABLE DES MATIERES

Contexte

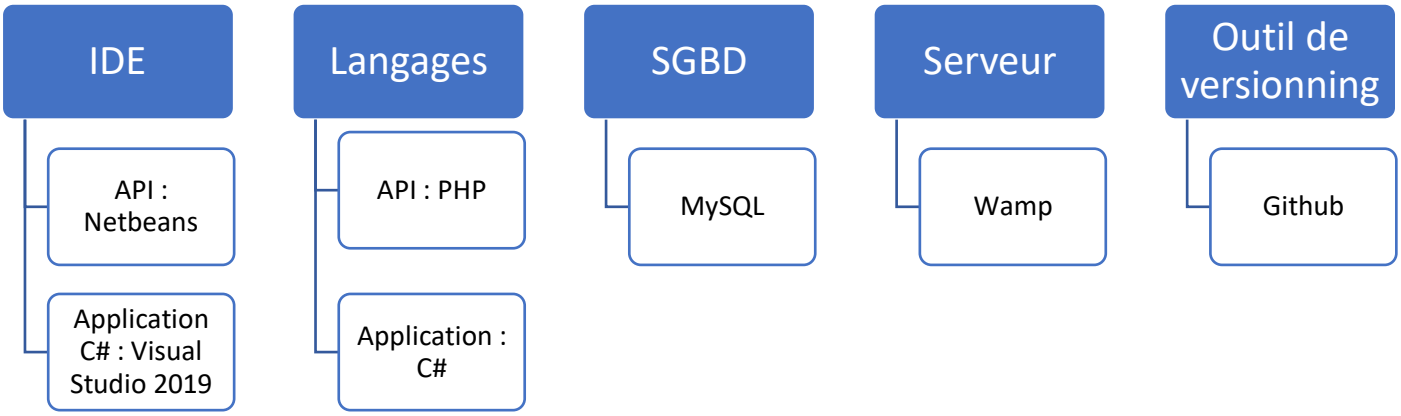
MediaTek86 est le réseau en charge de la gestion et de la coordination des médiathèques du département de la Vienne. Il facilite l'emprunt et la consultation de documents variés (livres, DVD, CD, revues) tout en développant des services numériques pour moderniser l'accès aux ressources culturelles.

Pour optimiser la gestion du catalogue et des commandes de documents, MediaTek86 a confié à InfoTechServices 86 le développement d'une application de bureau en C#, qui s'appuie sur une API REST en PHP et une base de données MySQL.

Fonctionnalités Actuelles :

- Consultation du catalogue avec recherche et filtrage (par nom, numéro, genre, public, rayon).
- Affichage des détails des documents (auteur, ISBN, synopsis, etc.).
- Ajout de nouvelles parutions de revues.

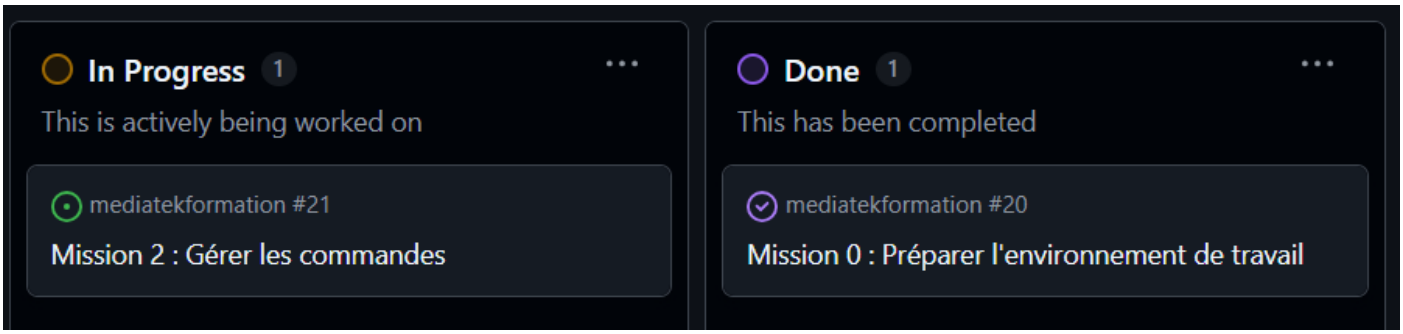
Langages et technologies utilisées



Commentaire

N'ayant pas eu la possibilité de travailler en équipe sur cet atelier je n'ai pas abordé les missions optionnelles. J'ai préféré me concentrer sur les missions principales (Missions 2, 4 , 5, 6 et 7) afin de bien comprendre le code existant et l'appel à l'API.

MISSION 2 : GERER LES COMMANDES

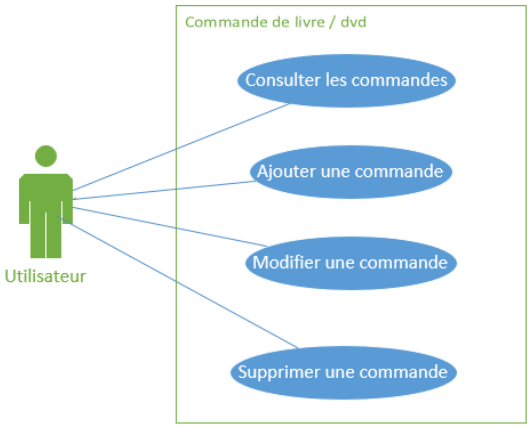


TACHE 1 : GERER LES COMMANDES DE LIVRES OU DE DVD

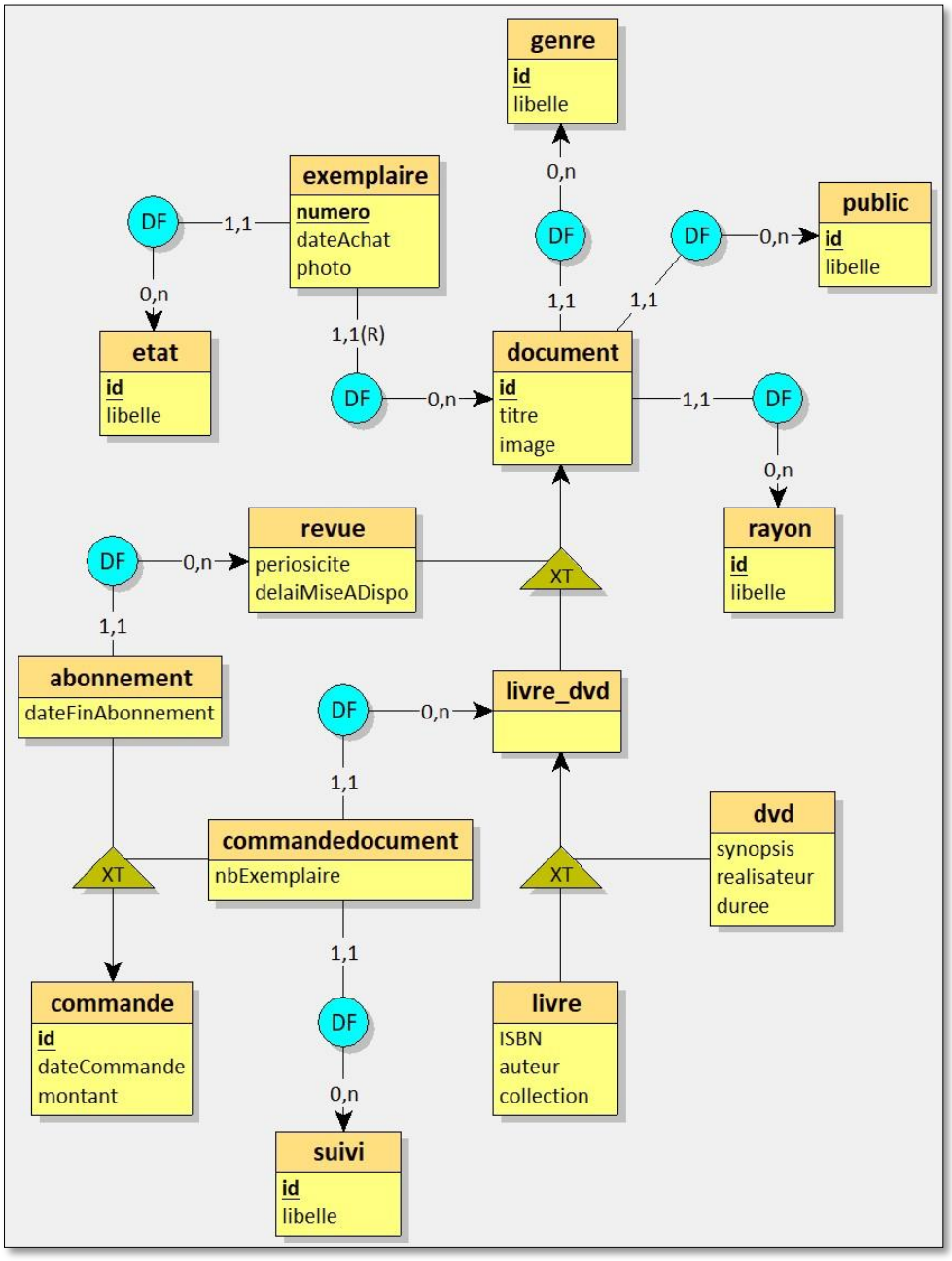
⋙ Dans la base de données, créer la table 'Suivi' qui contient les différentes étapes de suivi d'une commande de document de type livre ou dvd. Relier cette table à CommandeDocument.

⋙ 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 fille.
- 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.



Modification du modèle conceptuel de données avec ajout de l'entité « suivi » :



1.Modification de la base de donnée :

➢ Création de la table « suivi » et insertion du libellé

```sql
CREATE TABLE suivi(
   id VARCHAR(5) ,
   libelle VARCHAR(50) ,
   PRIMARY KEY(id)
);
```

```sql
INSERT INTO suivi VALUES
('1', 'en cours'),
('2', 'relancée'),
('3', 'livrée'),
('4', 'réglée');
```

➢ Ajout de la clé idsuivi dans la table **commandeDocument**

```sql
ALTER TABLE commandedocument
ADD COLUMN idsuivi VARCHAR(5) NOT NULL,
ADD FOREIGN KEY (idsuivi) REFERENCES suivi(id);
```

2. Création des nouvelles classes métier dans le dossier **« model »** :

➲ Commande :

```
public class Commande
{
    1 référence | 0 modification | 0 auteur, 0 modification
    public string Id { get; }
    1 référence | 0 modification | 0 auteur, 0 modification
    public DateTime DateCommande { get; set; }
    1 référence | 0 modification | 0 auteur, 0 modification
    public double Montant { get; }

    0 références | 0 modification | 0 auteur, 0 modification
    public Commande(string id, DateTime dateCommande, double montant)
    {
        Id = id;
        DateCommande = dateCommande;
        Montant = montant;
    }
}
```

➲ CommandeDocument :

```
/// <summary>
/// Classe CommandeDocument : Hérite de la classe Document
/// </summary>
1 référence | 0 modification | 0 auteur, 0 modification
public class CommandeDocument : Commande
{
    1 référence | 0 modification | 0 auteur, 0 modification
    public int NbExemplaire { get; }
    1 référence | 0 modification | 0 auteur, 0 modification
    public string IdLivreDvd { get; }
    1 référence | 0 modification | 0 auteur, 0 modification
    public int IdSuivi { get; }
    1 référence | 0 modification | 0 auteur, 0 modification
    public string Libelle { get; }

    0 références | 0 modification | 0 auteur, 0 modification
    public CommandeDocument(string id, DateTime dateCommande, double montant, int nbExemplaire, string idLivreDvd, int idSuivi, string libelle)
        : base (id, dateCommande, montant)
    {
        NbExemplaire = nbExemplaire;
        IdLivreDvd = idLivreDvd;
        IdSuivi = idSuivi;
        Libelle = libelle;
    }
}
```

➲ Suivi

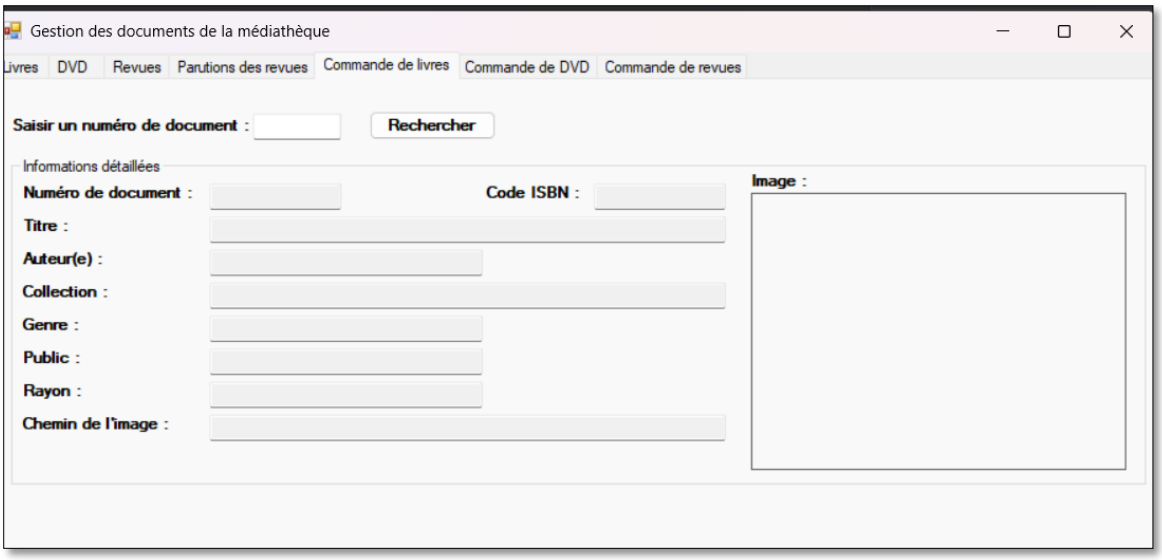
```
/// <summary>
/// Classe Suivi : Récupération des étapes de suivi d'une commande
/// </summary>
1 référence | 0 modification | 0 auteur, 0 modification
public class Suivi
{
    1 référence | 0 modification | 0 auteur, 0 modification
    public int Id { get; }
    1 référence | 0 modification | 0 auteur, 0 modification
    public string libelle { get; }

    0 références | 0 modification | 0 auteur, 0 modification
    public Suivi(int id, string libelle)
    {
        Id = id;
        this.libelle = libelle;
    }
}
```

Remplissage du combo pour le suivi de la commande :

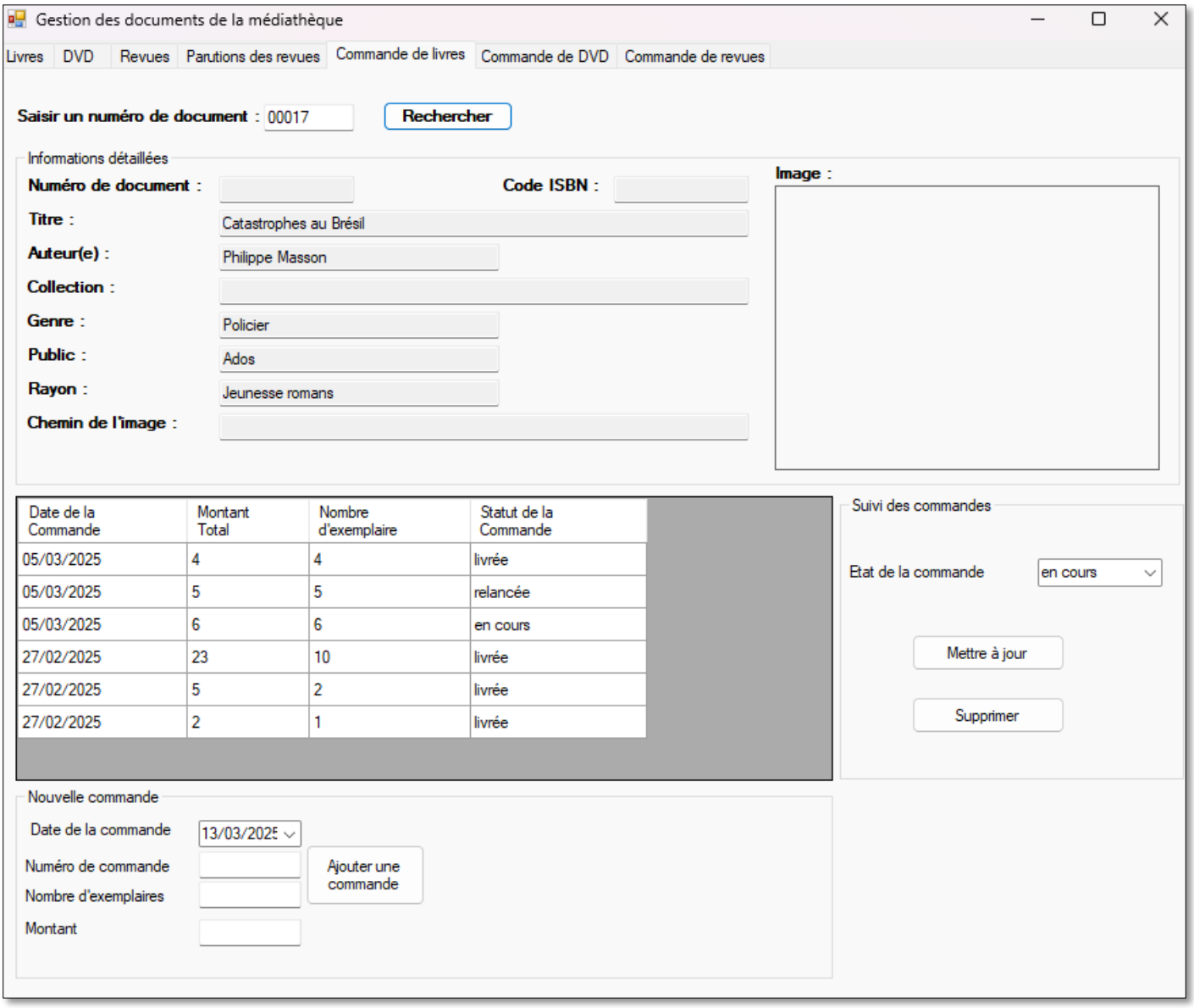
Dans **Access.cs** : Ajout de la méthode **GetAllSuivis()** pour récupérer toutes les étapes de suivi :

```
/// <summary>
/// Retourne toutes les étapes de suivi d'une commande
/// </summary>
/// <returns></returns>
0 références | 0 modification | 0 auteur, 0 modification
public List<Suivi> GetAllSuivis()
{
    IEnumerable<Suivi> lesSuivis = TraitementRecup<Suivi>(GET, "suivi",null);
    return new List<Suivi>(lesSuivis);
}
```

➲ Création des 2 nouveaux onglets : Commande de livres et commande de Dvd :



L'espace dédié à une nouvelle commande n'est accessible qu'une fois le numéro de document renseigné :



COTE VUE APPLICATION :

1. Recherche d'un livre par son numéro via le bouton Rechercher :

```
/// <summary>
/// recherche par numéro de livre saisi par l'utilisateur
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | STL, Il y a 46 minutes | 1 auteur, 3 modifications
private void btnRechercherCommandeLivre_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(txbNumDocCommandeLivre.Text))
    {
        // recherche un livre correspondant à l'id saisi
        Livre livre = lesLivres.Find(x => x.Id.Equals(txbNumDocCommandeLivre.Text));
        if (livre != null)
        {
            //affichage de ses informations
            AfficheInfosLivreRecherche(livre);
            dgvListeLivresCom.ClearSelection();
            cbxEtatCommandeLivre.SelectedIndex = 0;
        }
        else
        {
            MessageBox.Show("numéro introuvable");
        }
    }
}
```

Création de la méthode AfficheInfosLivreRecherche qui prends en paramètre un objet de type livre et permet de récupérer toutes ces informations :

```
/// <summary>
/// Affichage des informations du livre recherché
/// </summary>
/// <param name="livre">Objet livre contenant les informations de celui ci</param>
1 référence | STL, il y a 9 jours | 1 auteur, 2 modifications
private void AfficheInfosLivreRecherche(Livre livre)
{
    txbNumDocCommandeLivre.Text = livre.Id;
    txbCommandeLivreTitre.Text = livre.Titre;
    txbCommandeLivreAuteur.Text = livre.Auteur;
    txbCommandeLivreCodeISBN.Text = livre.Isbn;
    txbCommandeLivreCollection.Text = livre.Collection;
    txbCommandeLivreGenre.Text = livre.Genre;
    txbCommandeLivrePublic.Text = livre.Public;
    txbCommandeLivreRayon.Text = livre.Rayon;
    txbCommandeLivreUrlImage.Text = livre.Image;
    string image = livre.Image;
    try
    {
        pcbCommandeLivreImage.Image = Image.FromFile(image);
    }
    catch
    {
        pcbCommandeLivreImage = null;
    }
    // affiche la liste des commandes du livre
    AfficheLivreCommandes();
}
```

Création de la méthode AfficheLivreCommandes appelée dans la méthode précédente afin d'afficher les commandes associées à ce livre :

```
/// <summary>
/// Récupère et affiche les commandes d'un livre
/// </summary>
4 références | STL, Il y a 55 minutes | 1 auteur, 3 modifications
private void AfficheLivreCommandes()
{
    string idDocument = txbNumDocCommandeLivre.Text;
    //Appel au controleur pour récupérer les commandes liées à ce livre
    lesCommandesDocument = controller.GetCommandesDocument(idDocument);
    RemplirDgvCommandesLivres(lesCommandesDocument);
    // Si des commandes existent
    if (lesCommandesDocument.Count > 0)
    {
        ModifEnCoursComLivre(true, true);
    }
    else
    {
        MessageBox.Show("Aucune commande trouvée pour ce livre");
        ModifEnCoursComLivre(true, false);
    }
}
```

2. Création d'une nouvelle commande : Clic sur le bouton Ajouter :

```
1 référence | STL, Il y a 4 jours | 1 auteur, 5 modifications
private void btnAjoutCommandeLivre_Click(object sender, EventArgs e)
{
    // vérification des champs remplis
    if (!VerifierChampsCommande(txbNumCommandeLivre.Text, txbCommandeLivreNbEx.Text, txbCommandeLivreMontant.Text, dtpCommandeLivreDate.Value))
    {
        return;
    }
    // Vérification du suivi
    Suivi suivi = (Suivi)cbxEtatCommandeLivre.SelectedItem;
    if (suivi.Id != 1)
    {
        MessageBox.Show("Une nouvelle commande ne peut être créée qu'avec un état en cours");
        return;
    }
    try
    {
        string id = txbNumCommandeLivre.Text;
        string idLivreDvd = txbNumDocCommandeLivre.Text;

        // Si numero de commande existant déja
        if (CommandeExistante(id))
        {
            txbAbonnementNumCommande.Focus();
            return;
        }
        int nbExemplaire = int.Parse(txbCommandeLivreNbEx.Text);
        DateTime dateCommande = dtpCommandeLivreDate.Value;
        int idSuivi = suivi.Id;
        string libelleSuivi = suivi.Libelle;
        double montant = double.Parse(txbCommandeLivreMontant.Text);
        //Création
        CommandeDocument commandeDocument = new CommandeDocument(
            id, dateCommande, montant, nbExemplaire,
            idLivreDvd, idSuivi, libelleSuivi
        );
        if (controller.CreerCommandeDocument(commandeDocument))
        {
            AfficheLivreCommandes();
            MessageBox.Show("La commande numéro " + commandeDocument.Id + " pour le livre " + lesLivres.Find(o => o.Id == commandeDocument.IdLivr
            grpCommandeLivre.Controls.OfType<TextBox>().ToList().ForEach(txt => txt.Clear());
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Veuillez renseigner des valeurs correctes", "Information");
        ReinitialiserChampsComLivre();
```

3. Validation des champs de saisie

Une première méthode VerifierChampsCommande, est appelée pour contrôler que tous les champs requis ont été correctement renseignés par l'utilisateur. Cette méthode vérifie:

- La présence du numéro de commande
- La présence et la validité du nombre d'exemplaires
- La présence et la validité du montant
- Que la date de commande n'est pas antérieure à la date du jour

```
2 références | STL, Il y a 3 heures | 1 auteur, 1 modification
private bool VerifierChampsCommande(string commandeId, string nbExemplaire, string montant, DateTime dateCommande)
{
    // Vérification des champs vides
    if (string.IsNullOrWhiteSpace(commandeId))
    {
        MessageBox.Show("Numéro de commande obligatoire", "Information");
        return false;
    }

    if (string.IsNullOrWhiteSpace(nbExemplaire))
    {
        MessageBox.Show("Le nombre d'exemplaires est obligatoire", "Information");
        return false;
    }

    if (string.IsNullOrWhiteSpace(montant))
    {
        MessageBox.Show("Le montant est obligatoire", "Information");
        return false;
    }

    // Vérification des formats numériques
    if (!int.TryParse(nbExemplaire, out int nbEx) || nbEx <= 0)
    {
        MessageBox.Show("Le nombre d'exemplaires doit être un nombre entier ", "Information");
        return false;
    }

    if (!double.TryParse(montant, out double mt) || mt <= 0)
    {
        MessageBox.Show("Le montant doit être un nombre décimal positif", "Information");
        return false;
    }
    // Vérification de la date
    if (dateCommande <= DateTime.Now.Date)
    {
        MessageBox.Show("La date de commande ne peut pas être inférieure à la date d'aujourd'hui");
        return false;
    }
    return true;
}
```

Dans la seconde partie de btnAjoutCommandeLivre on récupère les valeurs saisies par l'utilisateur puis pour éviter les doublons, on vérifie que le numéro de commande saisi n'existe pas déjà à l'aide de la méthode CommandeExistante() qui :

- Prend en paramètre le numéro de commande saisi
- Récupère toutes les commandes existantes
- Utilise la méthode LINQ Exist() pour vérifier si le numéro est déjà utilisé
- Affiche un message d'avertissement si c'est le cas

Cette vérification permet de s'assurer que chaque commande est unique, ce qui évite les confusions et les erreurs dans la gestion des commandes.

```
/// <summary>
/// vérifie si l'identifiant renseigné existe déja
/// </summary>
/// <param name="id">identifiant de la commande</param>
/// <returns>true si la commande existe, sinon false</returns>
3 références | STL, il y a 5 jours | 1 auteur, 3 modifications
private bool CommandeExistante(string id)
{
    List<Commande> commandesExistantes = controller.GetAllCommandes();
    if (commandesExistantes.Exists(c => c.Id == id))
    {
        MessageBox.Show("Une commande avec ce numéro existe déjà", "Erreur");
        //si commande existe
        return true;
    }
    //si elle n'existe pas
    return false;
}
```

Une fois toutes ces étapes validées, l'appel au contrôleur pour la création d'une nouvelle commande est effectué et un message d'information s'affiche si l'opération à réussi, la liste des commandes est alors mise à jour.

4. Modification d'une commande :

La gestion de la modification du suivi des commandes est effectuée par la procédure évènementielle sous le bouton Modifier :

```
/// <summary>
/// Clic sur le bouton modifier / Gestion de l'état de la commande
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | STL, il y a 11 jours | 1 auteur, 1 modification
private void btnMAJSuiviCommandeLivre_Click(object sender, EventArgs e)
{
    ModifierSuiviCommandeLivre();
}
```

Création d'une nouvelle méthode ModifierSuiviCommandeLivre() :

```
/// <summary>
/// Modification de l'état d'une commande si son suivi actuel le permet
/// </summary>
1 référence | STL, Il y a 1 jour | 1 auteur, 2 modifications
private void ModifierSuiviCommandeLivre()
{
    // ligne sélectionnée
    if (dgvListeLivresCom.CurrentRow != null)
    {
        CommandeDocument commandeDocument = (CommandeDocument)bdgLivreCommandesListe[bdgLivreCommandesListe.Position];
        //récupération du suivi sélectionné par l'utilisateur dans le cbx
        Suivi nouvelEtat = (Suivi)cbxEtatCommandeLivre.SelectedItem;

        //Vérification des états de suivi
        if (commandeDocument.IdSuivi >= 3)
        {
            MessageBox.Show("Une commande livrée ou réglée ne peut pas revenir en arrière.", "Modification impossible");
            return;
        }

        if (nouvelEtat.Id == 4 && commandeDocument.IdSuivi != 3)
        {
            MessageBox.Show("Une commande ne peut être réglée que si elle est livrée.", "Modification impossible");
            return;
        }
        commandeDocument.IdSuivi = nouvelEtat.Id;

        if (controller.ModiferCommandeDocument(commandeDocument))
        {
            MessageBox.Show("Suivi mis à jour avec succès !");
            AfficheLivreCommandes();
        }
        else
        {
            MessageBox.Show("Erreur lors de la mise à jour du suivi.", "Erreur");
        }
```

- Récupération de l'objet CommandeDocument sélectionné dans le datagridview dgvListeLivresCom
- Récupération du suivi de commande sélectionné par l'utilisateur dans le combobox cbxEtatCommandeLivre et affectation à nouvelEtat.
- Vérifications des conditions de modifications : une commande livrée ou réglée ne peut pas revenir en arrière / une commande ne peut être réglée que si elle est réglée.
- Si toutes les conditions sont respectées le suivi de la commande est mis à jour avec le nouvel état sélectionné par l'utilisateur et un message de confirmation est affiché.

5. Suppression d'une commande :

```
/// <summary>
/// Suppression d'une commande de livre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | STL, Il y a 1 jour | 1 auteur, 3 modifications
private void btnSupprimerCommandeLivre_Click(object sender, EventArgs e)
{

    CommandeDocument commandeDocument = (CommandeDocument)bdgLivreCommandesListe[bdgLivreCommandesListe.Position];
    if (dgvListeLivresCom.CurrentCell != null)
    {
        // vérifie si la commande est livrée ou réglée
        if (commandeDocument.IdSuivi > 2)
            MessageBox.Show("Une commande livrée ou réglée ne peut etre supprimée");
        else if (MessageBox.Show("Voulez vous supprimer la commande " + commandeDocument.Id + " pour le livre " + lesLivres.Find(o => o.Id == commandeDocument.IdLivreDvd).Titre + " ?",
            "Validation suppresion", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //appel au controleur
            if (controller.SupprimerCommandeDocument(commandeDocument.Id))
            {
                AfficheLivreCommandes();
            }
            else
            {
                MessageBox.Show("erreur");
            }
        }
    }
    else
    {
        MessageBox.Show("la selection d'une commande est obligatoire");
    }
}
```

- Récupération de la commande sélectionnée dans le datagridview
- Vérification des conditions de suppression : si la commande est déjà livrée ou réglée, un message informe l'utilisateur qu'une commande avec ce suivi ne peut pas être supprimée.
- Affichage d'un message de confirmation pour que l'utilisateur valide la suppression, afin d'éviter toute erreur.
- Si toutes les étapes sont validées, la commande est supprimée et un message de confirmation est affiché.

Trigger qui réalise également la suppression dans la classe fille :

```
1  CREATE TRIGGER `SuppressionCommande` BEFORE DELETE ON `commande`
2   FOR EACH ROW BEGIN
3      DECLARE etatSuivi VARCHAR(5);
4
5      SELECT idSuivi INTO etatSuivi
6      FROM commandedocument
7      WHERE id = OLD.id
8      LIMIT 1;
9
10     IF etatSuivi = '3' OR etatSuivi = '4' THEN
11         SIGNAL SQLSTATE '45000'
12         SET MESSAGE_TEXT = 'Impossible de supprimer une commande livrée ou réglée';
13     ELSE
14         DELETE FROM commandedocument WHERE id = OLD.id;
15         DELETE FROM abonnement WHERE id = OLD.id;
16     END IF;
17 END
```

Trigger qui se déclenche lorsqu'une commande passe à l'étape « livrée » et crée autant de tuples dans la table Exemplaire que nécessaire :

```sql
CREATE TRIGGER `CreationExemplaire` AFTER UPDATE ON `commandedocument`
 FOR EACH ROW BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE dernierNumero INT DEFAULT 0;

    IF NEW.idSuivi = '3' AND OLD.idSuivi != '3' THEN
        SELECT IFNULL(MAX(numero), 0) INTO dernierNumero
        FROM exemplaire
        WHERE id = NEW.idLivreDvd;

        SET @dateCmd = (SELECT dateCommande FROM commande WHERE id = NEW.id);

        WHILE i < NEW.nbExemplaire DO
            SET dernierNumero = dernierNumero + 1;
            INSERT INTO exemplaire (id, numero, dateAchat, photo, idEtat)
            VALUES (NEW.idLivreDvd, dernierNumero, @dateCmd, '', '00001');
            SET i = i + 1;
        END WHILE;
    END IF;
END
```

ACCES AUX DONNEES :

Création de plusieurs méthodes pour :

➲ Récupérer les commandes de livres ou Dvd :

```csharp
/// <summary>
/// récupère les commandes pour un livre ou un dvd
/// </summary>
/// <param name="idDocument"></param>
/// <returns>Liste d'objet CommandeDocument</returns>
1 référence | STL, Il y a 1 jour | 1 auteur, 2 modifications
public List<CommandeDocument> GetCommandesDocument(string idDocument)
{
    String jsonIdDocument = convertToJson("idLivreDvd", idDocument);
    List<CommandeDocument> lesCommandesDoc = TraitementRecup<CommandeDocument>(GET, "commandedocument/" + jsonIdDocument, null);
    return lesCommandesDoc;
}
```

➲ Enregistrer une nouvelle commande de livres ou Dvd :

```
/// <summary>
/// Ecriture d'une nouvelle commande de document dans la base de données
/// </summary>
/// <param name="commandeDoc">commande de document à insérer</param>
/// <returns>true si l'insertion a pu se faire (retour != null)</returns>
1 référence | STL, il y a 17 jours | 1 auteur, 1 modification
public bool CreerCommandeDocument(CommandeDocument commandeDoc)
{
    String jsonCommandeDocument = JsonConvert.SerializeObject(commandeDoc, new CustomDateTimeConverter());
    try
    {
        List<CommandeDocument> liste = TraitementRecup<CommandeDocument>(POST, "commandedocument", "champs=" + jsonCommandeDocument);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

➲ Modifier le suivi d'une commande :

```
/// <summary>
/// Modification du suivi d'une commande
/// </summary>
/// <param name="commandeDocument"></param>
/// <returns>true si modification effectuée</returns>
1 référence | STL, il y a 1 jour | 1 auteur, 2 modifications
public bool ModifierCommandeDocument(CommandeDocument commandeDocument)
{
    try
    {
        string jsonCommandeIdSuivi = convertToJson("idSuivi", commandeDocument.IdSuivi);
        List<CommandeDocument> liste = TraitementRecup<CommandeDocument>(PUT, "commandedocument/" + commandeDocument.Id, "champs=" + jsonCommandeIdSuivi);
        return (liste != null);
    }
    catch (Exception ex)
    {


        Console.WriteLine(ex.Message);
    }
    return false;
}
```

➲ Supprimer une commande de livre ou Dvd :

```
/// <summary>
/// Suppression d'une commande
/// </summary>
/// <param name="idCommande"></param>
/// <returns></returns>
1 référence | STL, il y a 10 jours | 1 auteur, 2 modifications
public bool SupprimerCommandeDocument(string idCommande)
{
    try
    {
        string jsonIdCommande = convertToJson("id", idCommande);
        List<CommandeDocument> liste = TraitementRecup<CommandeDocument>(DELETE, "commandedocument/" + jsonIdCommande, null);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

DANS L'API :

Création de nouvelles méthodes :

➲ selectCommandesDocument récupère l'ensemble des commandes liées à un livre ou un DVD en effectuant une jointure entre les tables commandeDocument, commande et suivi. Les résultats sont ensuite triés par date de commande décroissante grâce à l'utilisation de order by.

```php
/**
 * récupère les commandes de livre ou dvd
 * @param array|null $champs
 * @return array|null
 */
private function selectCommandesDocument(?array $champs): ?array {
    if (empty($champs) || !isset($champs['idLivreDvd'])) {
        return null;
    }
    $champNecessaire['idLivreDvd'] = $champs['idLivreDvd'];
    $requete = "Select cd.id, c.dateCommande, c.montant, cd.nbExemplaire, cd.idLivreDvd, ";
    $requete .= "cd.idSuivi, s.libelle ";
    $requete .= "from commandedocument cd join commande c on cd.id=c.id ";
    $requete .= "join suivi s on cd.idSuivi=s.id ";
    $requete .= "where cd.idLivreDvd = :idLivreDvd ";
    $requete .= "order by c.dateCommande DESC";
    return $this->conn->queryBDD($requete, $champNecessaire);
}
```

➲insertCommande enregistre une commande de livre ou de DVD en ajoutant d'abord les informations générales de la commande dans la table commande, puis en enregistrant les détails spécifiques dans la table commandedocument.

```php
/**
 * Insertion d'une commande de livre ou dvd
 * @param array|null $champs
 * @return int|null
 */
private function insertCommande(?array $champs): ?int {

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

➲ updateSuiviCommande : modifie l'étape de suivi d'une commande en mettant à jour la valeur de idSuivi dans la table commandeDocument. L'opération est réalisée via la méthode updateOneTupleOneTable, qui applique la modification sur l'enregistrement correspondant à l'identifiant de la commande fournie.

```php
/**
 * Modifie l'étape de suivi d'une commande
 * @param string|null $id identifiant de la commande à modifier
 * @param array|null $champs contient l'idSuivi à mettre à jour
 * @return int|null nombre de tuples modifiés ou null si erreur
 */
private function updateSuiviCommande(?string $id, ?array $champs): ?int {

    // Créer un tableau avec uniquement le champ idSuivi pour la mise à jour
    $champsUpdate = ["idSuivi" => $champs["idSuivi"]];

    // Mettre à jour le suivi
    return $this->updateOneTupleOneTable("commandedocument", $id, $champsUpdate);
}
```

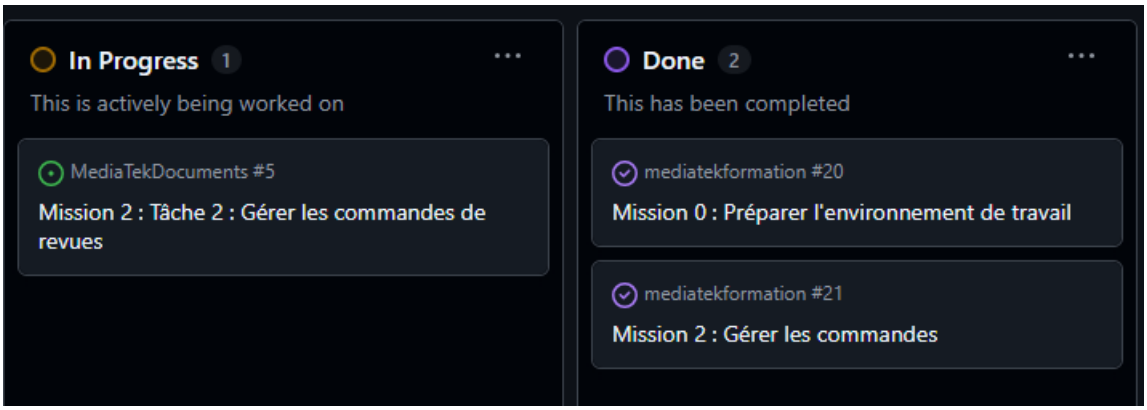
➲ deleteCommande est conçue pour supprimer une commande en effaçant d'abord les enregistrements associés dans la table commandedocument, puis en supprimant la commande correspondante dans la table commande. L'opération est réalisée en deux étapes via la méthode deleteTuplesOneTable, garantissant l'intégrité des données en supprimant d'abord les dépendances avant la commande elle-même. ➲

```php
/**
 * Suppression d'une commande
 * @param array|null $champs
 * @return int|null
 */
private function deleteCommande(?array $champs): ?int {
    $champNecessaire['id'] = $champs['id'];

    $result = $this->deleteTuplesOneTable("commandedocument", $champNecessaire);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->deleteTuplesOneTable("commande", $champNecessaire);
}
```

ONGLET COMMANDE DE DVD :

Pour cette tâche j'ai suivi un processus similaire à celui utilisé pour les commandes de livres.

TACHE 2 : GERER LES COMMANDES DE REVUES



⋙ Créer un onglet pour gérer les commandes de revues / La chartre graphique doit correspondre à l'existant

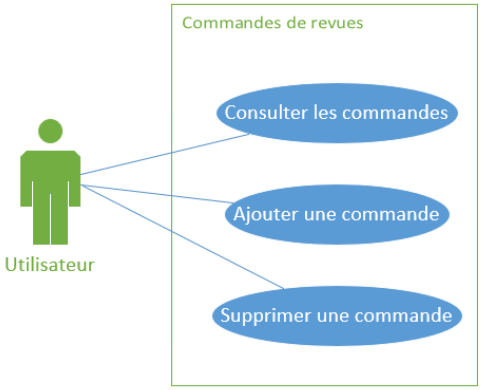
⋙ Permettre la sélection d'une revue par son numéro, afficher les informations de la revue ainsi que la liste des commandes (abonnements), triée par date (ordre inverse de la chronologie). La liste doit comporter les informations suivantes : date de la commande, montant et date de fin d'abonnement

⋙ 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.

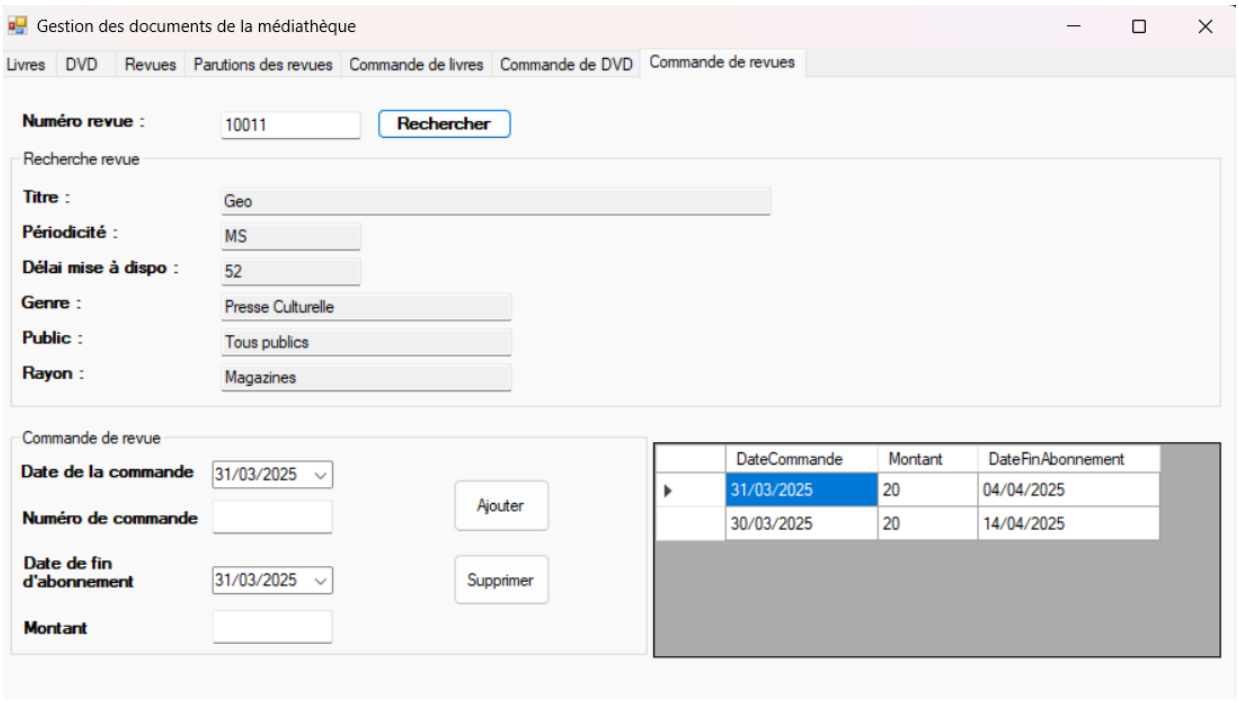
⋙ Permettre de supprimer une commande de revue uniquement si aucun exemplaire n'est rattaché (donc, en vérifiant la date de l'exemplaire, comprise entre la date de la commande et la date de fin d'abonnement). Pour cela, créer et utiliser la méthode 'ParutionDansAbonnement' qui reçoit en paramètre 3 dates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

⋙ Toutes les sécurités seront mises en places pour éviter les 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 fin abonnement) triée sur la date dans l'ordre chronologique



La démarche est similaire à celle des onglets livre et dvd avec une spécificité : lors d'une commande de revue la date de fin d'abonnement doit obligatoirement être renseignée.



Création d’une méthode permettant de valider les saisies :

```
1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
public static bool VerifierChampsCommandeRevue(string idCommande, string montantText, DateTime dateCommande, DateTime dateFinAbonnement)
{
    if (string.IsNullOrWhiteSpace(idCommande))
    {
        MessageBox.Show("Numéro de commande obligatoire", "Information");
        return false;
    }

    if (string.IsNullOrWhiteSpace(montantText))
    {
        MessageBox.Show("Le montant est obligatoire", "Information");
        return false;
    }

    if (!double.TryParse(montantText, out double montant) || montant <= 0)
    {
        MessageBox.Show("Le montant doit être un nombre décimal positif.", "Erreur");
        return false;
    }

    if (dateFinAbonnement < dateCommande)
    {
        MessageBox.Show("La date de fin d'abonnement ne peut pas être antérieure à la date de commande.", "Erreur");
        return false;
    }

    if (dateCommande <= DateTime.Today)
    {
        MessageBox.Show("La date de commande ne peut pas être inférieure à la date d'aujourd'hui");
        return false;
    }

    return true;
}
```

Suppression d’une commande de revue :

Création de la méthode ParutionDansAbonnement qui permet de vérifier si la date de parution d'une revue se trouve entre la date de commande et la date de fin d'abonnement, et de supprimer une commande uniquement si aucun exemplaire n'y est associé.

```
/// <summary>
/// Retourne vrai si la date de parution est entre dateCommande et dateFinAbonnement
/// </summary>
/// <param name="dateCommande">date de la commande</param>
/// <param name="dateFinAbonnement">date de fin d'abonnement</param>
/// <param name="dateParution">date de parution</param>
/// <returns>true si date parution entre date commande et date fin</returns>
6 références | STL, il y a 5 jours | 1 auteur, 2 modifications
public bool ParutionDansAbonnement(DateTime dateCommande, DateTime dateFinAbonnement, DateTime dateParution)
{
    return (DateTime.Compare(dateCommande, dateParution) < 0 && DateTime.Compare(dateParution, dateFinAbonnement) < 0);
}
```

Fenêtre d’alerte permettant d’informer l’utilisateur sur les abonnements arrivant à expiration :

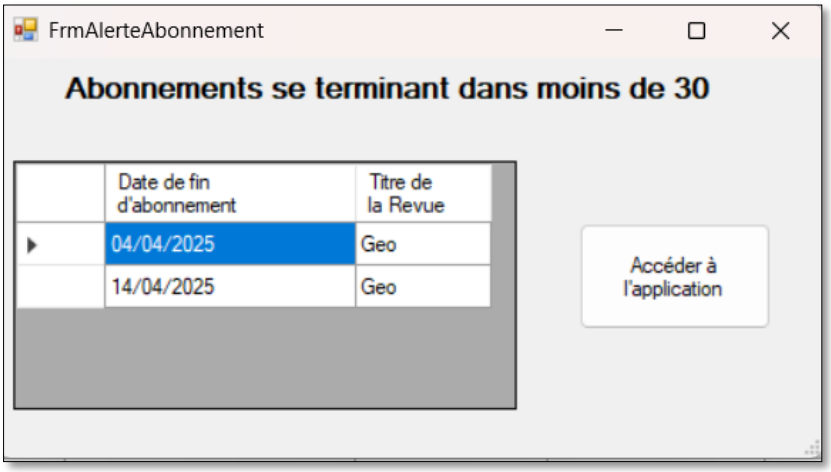
Lors de l’ouverture de la fenêtre principale la méthode AfficherAlerteAbonnementSiNecessaire est appelée :
- la méthode appelle le contrôleur pour récupérer la liste des abonnements expirants via la méthode GetAbonnementExpirants

```
/// <summary>
/// Récupère la liste des abonnements expirants
/// </summary>
/// <returns>liste d'objets abonnement dont la date de fin abonnement est inférieure à 30 jours</returns>
1 référence | STL, Il y a 1 heure | 1 auteur, 3 modifications
public List<Abonnement> GetAbonnementsExpirants()
{
    List<Abonnement> abonnementsExpirants = new List<Abonnement>();
    List<Revue> revues = GetAllRevues();

    foreach (Revue revue in revues)
    {
        List<Abonnement> abonnements = GetAbonnementsRevue(revue.Id);
        var expirants = abonnements.Where(o =>
            o.DateFinAbonnement <= DateTime.Now.AddMonths(1) &&
            o.DateFinAbonnement >= DateTime.Now).ToList();

        abonnementsExpirants.AddRange(expirants);
    }

    return abonnementsExpirants.OrderBy( a=> a.DateFinAbonnement).ToList();
}
```

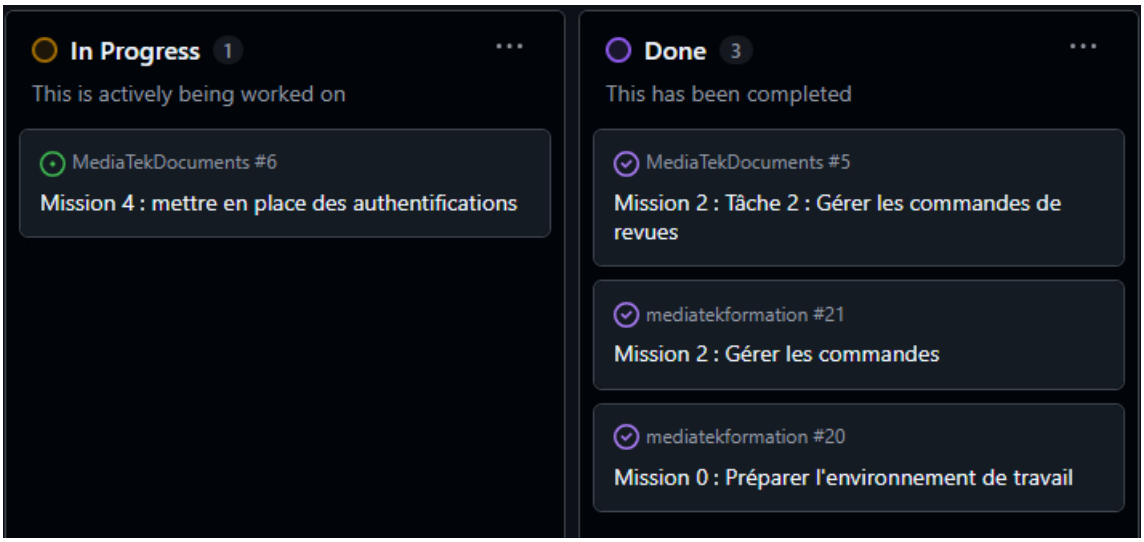
- Création d'une nouvelle liste pour stocker les abonnements expirants
- Parcours de chaque revue afin de récupérer les abonnements correspondants.
- Application de LINQ pour filtrer les abonnements dont la date de fin est inférieure ou égale à 30 jours.
- Tri de la liste par la date de fin d'abonnement

Une fois la liste récupérée, si des abonnements expirants existent, la méthode AfficherAlerteAbonnementSiNecessaire crée une nouvelle instance du formulaire FrmAlerteAbonnement en lui passant la liste des abonnements expirants et des revues associées.

Le formulaire utilise ces données pour afficher la fenêtre d'alerte pour et indiquer le titre des revues et leur date de fin d'abonnement :



MISSION 4 : METTRE EN PLACE DES AUTHENTIFICATIONS

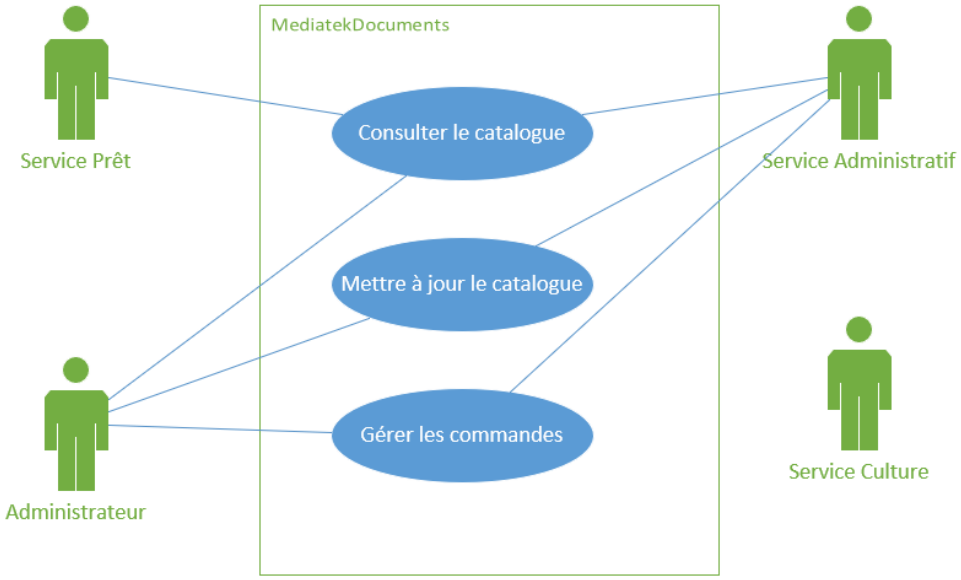


⋙ Dans la base de données, ajouter une table Utilisateur et une table Service, sachant que chaque utilisateur ne fait partie que d'un service. Pour réaliser les tests, remplir les tables d'exemples.

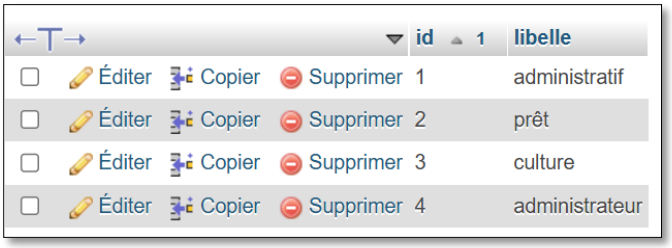
⋙ Ajouter une première fenêtre d'authentification. Faire en sorte que l'application démarre sur cette fenêtre.

⋙ Suivant le type de personne authentifiée, empêcher certains accès en rendant invisibles ou inactifs certains onglets ou objets graphiques.

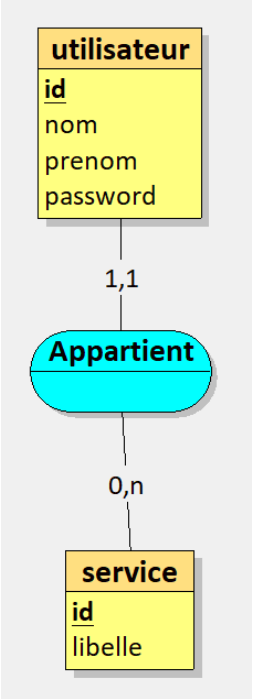
⋙ Dans le cas du service Culture qui n'a accès à rien, afficher un message précisant que les droits ne sont pas suffisants pour accéder à cette application, puis fermer l'application.



1. Ajout des 2 nouvelles tables en base de données : Utilisateur et Service

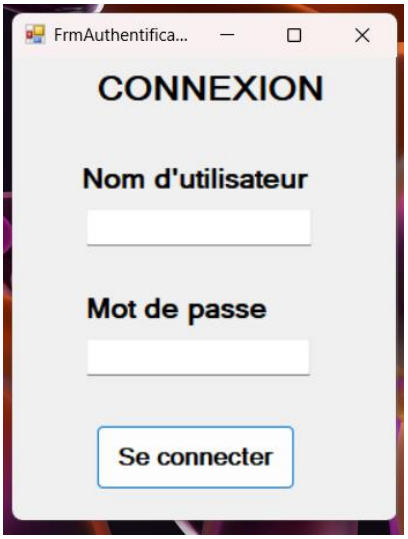


	id	libelle
Éditer Copier Supprimer	1	administratif
Éditer Copier Supprimer	2	prêt
Éditer Copier Supprimer	3	culture
Éditer Copier Supprimer	4	administrateur



2. Lancement de l'application via le formulaire de connexion :

```
namespace MediaTekDocuments
{

    0 références | STL, Il y a 19 heures | 1 auteur, 4 modifications
    static class Program
    {
        /// <summary>
        /// Point d'entrée principal de l'application.
        /// </summary>
        [STAThread]
        0 références | STL, Il y a 19 heures | 1 auteur, 4 modifications
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FrmAuthentification());
        }
    }
}
```



L'utilisateur entre ses informations de connexion : login et mot de passe dans les champs txbNumUtilisateur et txbPassword

- Lorsqu'il clique sur le bouton se connecter, le contrôleur est appelé pour récupérer l'utilisateur

```
/// <summary>
/// Forumulaire d'authentification
/// </summary>
3 références | STL, Il y a 2 heures | 1 auteur, 4 modifications
public partial class FrmAuthentification : Form
{
    private readonly FrmAuthentificationController controller;
    /// <summary>
    /// Constructeur
    /// </summary>
    1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
    public FrmAuthentification()
    {
        InitializeComponent();
        this.controller = new FrmAuthentificationController();
    }
    /// <summary>
    /// clic sur btn Se connecter
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
    private void btnSeConnecter_Click(object sender, EventArgs e)
    {
        // Récupération de l'utilisateur authentifié
        Utilisateur utilisateur = controller.GetAuthentification(txbNomUtilisateur.Text, txbPassword.Text);
        // Vérification si l'utilisateur est bien authentifié
        if (utilisateur != null)
        {
            this.Visible = false; // Masquer la fenêtre d'authentification

            // Ouvrir l'application principale en passant l'utilisateur
            FrmMediatek mediatek = new FrmMediatek(utilisateur);
            mediatek.Show();
        }
        else
        {
            MessageBox.Show("Erreur sur le login ou le mot de passe.", "Authentification échouée", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

3. Création d'un nouveau contrôleur dédié à l'authentification :

```
/// <summary>
/// récupération de l'instance unique d'accès aux données
/// </summary>
1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
public FrmAuthentificationController()
{
    access = Access.GetInstance();
}
/// <summary>
/// Vérifie les informations pour l'authentification d'un utilisateur
/// </summary>
/// <param name="loginUtilisateur">login</param>
/// <param name="password">mot de passe</param>
/// <returns>utilisateur</returns>
1 référence | STL, Il y a 19 heures | 1 auteur, 3 modifications
public Utilisateur GetAuthentification(string loginUtilisateur, string password)
{
    //Récupère l'utilisateur à partir de la méthode GetAuthentification de la classe Access
    Utilisateur utilisateur = access.GetAuthentification(loginUtilisateur);

    // Si aucun utilisateur n'est trouvé, retourne false
    if (utilisateur == null)
    {
        return null;
    }

    // Vérification du mot de passe haché
    string passwordHache = HashPassword(password);
    Console.WriteLine("Mot de passe haché fourni: " + passwordHache);
    Console.WriteLine("Mot de passe dans la base de données: " + utilisateur.Password);

    if (passwordHache != utilisateur.Password)
    {
        return null; // Échec de l'authentification
    }
    return utilisateur;
}
```

```php
/**
 * Récupère les informations d'un utilisateur pour l'authentification
 * @param array|null $champs Tableau contenant les critères de recherche
 * @return array|null Informations de l'utilisateur ou null si non trouvé
 */
private function selectUtilisateur(?array $champs): ?array {
    if (empty($champs)) {
        return null;
    }

    $champNecessaire['login'] = $champs['login'];
    $requete = "select u.login, u.password , u.nom, u.prenom, u.idService, s.libelle  ";
    $requete .= "from utilisateur u  join service s on s.id=u.idService ";
    $requete .= "where u.login =:login  ";

    return $this->conn->queryBDD($requete, $champNecessaire);
}
```

- La méthode GetAuthentification vérifie les informations de connexion. Elle prends en paramètres le login et de l'utilisateur et son mot de passe.
- Le contrôleur appelle la méthode Access communiquant avec l'API
- Si l'utilisateur est trouvé, le mot de passe fourni est haché en utilisant la méthode hashpassword
- Le mot de passe haché est ensuite comparé avec celui stocké dans la base de données.

https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.sha256?view=net-9.0

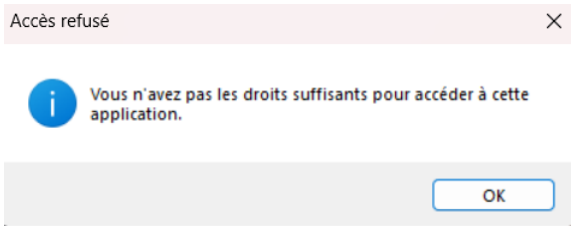
```php
/// <summary>
/// Méthode pour hasher un mot de passe avec SHA-256
/// </summary>
1 référence | STL, il y a 13 jours | 1 auteur, 1 modification
private string HashPassword(string password)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        // Calculer le hash du mot de passe
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(password));

        // Convertir le tableau de bytes en une chaîne hexadécimale
        StringBuilder builder = new StringBuilder();
        foreach (byte t in bytes)
        {
            builder.Append(t.ToString("x2"));
        }

        return builder.ToString();
    }
}
```

- Utilisation de l'algorithme de hachage SHA-256
- Conversion du mot de passe en un tableau d'octets
- Calcul du hachage
- Un StringBuilder est utilisé pour assembler chaque valeur générée en une chaîne représentant le hachage complet du mot de passe
- Une fois tous les octets convertis et ajoutés à la chaîne, celle-ci est retournée et peut être comparée avec le mot de passe haché stocké en base de données

4. Gestion des droits des utilisateurs selon leur service :

```
public partial class FrmMediatek : Form
{

    #region Commun
    private readonly FrmMediatekController controller;
    private readonly BindingSource bdgGenres = new BindingSource();
    private readonly BindingSource bdgPublics = new BindingSource();
    private readonly BindingSource bdgRayons = new BindingSource();


    private readonly Utilisateur utilisateurAuth;


    /// <summary>
    /// Constructeur : création du contrôleur lié à ce formulaire
    /// </summary>
    1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
    internal FrmMediatek(Utilisateur utilisateur)
    {
        this.utilisateurAuth = utilisateur;
        // Vérifier les droits d'accès avant d'initialiser les composants
        if (utilisateurAuth.Service.Libelle == "culture")
        {
            MessageBox.Show("Vous n'avez pas les droits suffisants pour accéder à cette application.",
                            "Accès refusé", MessageBoxButtons.OK, MessageBoxIcon.Information);
            Application.Exit();
            return;
        }
        InitializeComponent();
        this.controller = new FrmMediatekController();

        ConfigurerAcces();


    }
```

- Déclaration d'un objet de type utilisateur : utilisateurAuth qui va contenir toutes les informations sur l'utilisateur connecté (nom, login, mot de passe, service auquel il appartient).
- Le constructeur du formulaire prends en paramètre l'utilisateur connecté qui est ensuite assigné à utilisateurAuth.
- Si l'utilisateur appartient au service Culture un message d'information apparait lui indiquant qu'il n'a pas les droits nécessaires pour accéder à l'application.



- Si l'utilisateur possède les droits nécessaires (service administratif, administrateur, service prêt) l'interface graphique est initialisée et la méthode ConfigurerAcces est appelée.

```
/// <summary>
/// Visibilité des éléments selon l'utilisateur connecté et son service
/// </summary>
1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
private void ConfigurerAcces()
{
    string libelleService = utilisateurAuth.Service.Libelle;

    if (libelleService == "prêt")
    {
        tabOngletsApplication.TabPages.Remove(tabCommandeLivre);
        tabOngletsApplication.TabPages.Remove(tabCommandeDVD);
        tabOngletsApplication.TabPages.Remove(tabCommandeRevue);
        tabOngletsApplication.TabPages.Remove(tabReceptionRevue);
    }
}
```

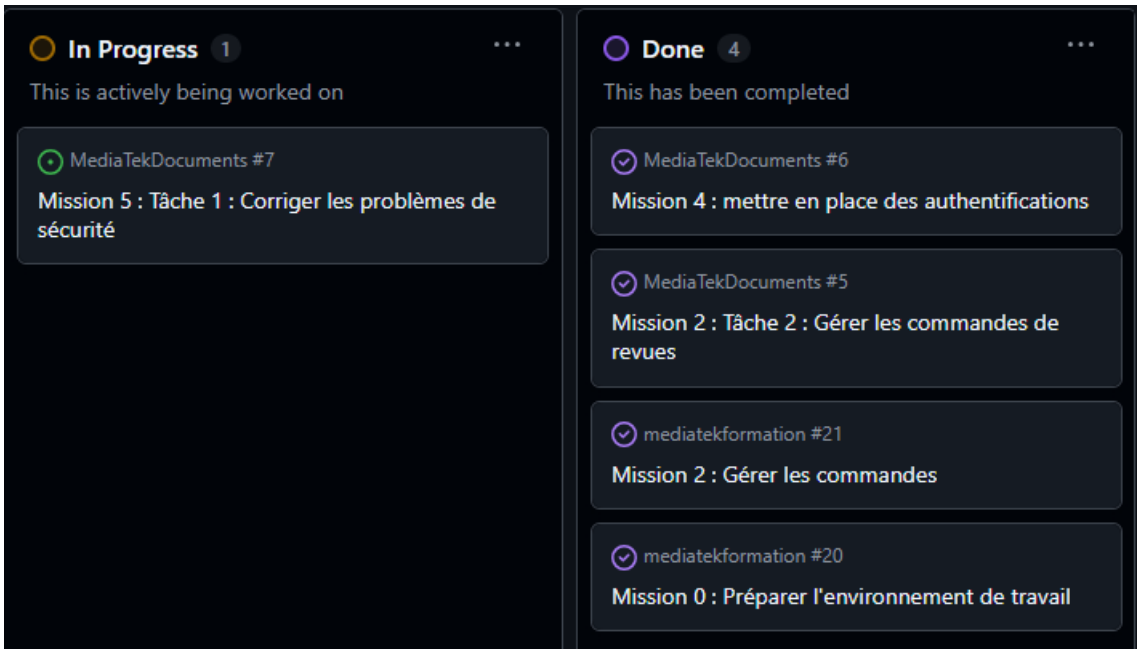
- Cette méthode récupère le libellé du service auquel appartient l'utilisateur, puis applique des restrictions pour les utilisateurs du service "prêt", leur accordant uniquement l'accès aux onglets de consultation des documents.

Afin que la fenêtre d'alerte de fin d'abonnement n'apparaisse que pour les personnes concernées, un test est effectué dans la méthode événementielle FrmMediatek_Shown pour que seuls les administrateurs ou les utilisateurs du service administratif puissent accéder à cette information :

```
/// <summary>
/// Affichage de l'alerte pour les abonnements selon l'utilisateur authentifié
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | STL, il y a 5 jours | 1 auteur, 2 modifications
private void FrmMediatek_Shown(object sender, EventArgs e)
{
    // Vérifier les abonnements si l'utilisateur est autorisé
    if (utilisateurAuth.Service.Libelle == "administratif" || utilisateurAuth.Service.Libelle == "administrateur")
    {
        AfficherAlerteAbonnementSiNecessaire();
    }
}
```

MISSION 5 : ASSURER LA SECURITE, LA QUALITE ET INTEGRER DES LOGS

TACHE 1 : CORRIGER DES PROBLEMES DE SECURITE



✕ Problème n°1 : L'accès à l'API se fait en authentification basique, avec le couple « login :pwd » en dur dans le code de l'application (Constructeur de la classe Access).Le but est de sécuriser cette information.

✓ Correction :

Modification du fichier app.config pour rajouter les informations de connexion (authentification et url de l'api) :

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
    </configSections>
     <connectionStrings>
        <add name="MediaTekDocuments.Properties.Settings.mediatekAuthenticationString"
             connectionString="admin:adminpwd" />
        <add name="MediaTekDocuments.Properties.Settings.mediatekConnectionString"
             connectionString="http://localhost/rest_mediatekdocuments/" />
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-13.0.0.0" newVersion="13.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Memory" publicKeyToken="cc7b13ffcd2ddd51" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.0.1.2" newVersion="4.0.1.2" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

Modification de Access.cs :

Une fois les informations sensibles déplacées dans le fichier app.config , elle sont ensuite récupérées à l'aide de la méthode ConfigurationManager.ConnectionString.

https://learn.microsoft.com/fr-fr/dotnet/api/system.configuration.configurationmanager.connectionstrings?view=windowsdesktop-9.0

```
/// <summary>
/// adresse de l'API
/// </summary>
private static readonly string connectionName = "MediaTekDocuments.Properties.Settings.mediatekConnectionString";
/// <summary>
/// Login api
/// </summary>
private static readonly string authenticationName = "MediaTekDocuments.Properties.Settings.mediatekAuthenticationString";
```

Les informations sont ensuite récupérées dans le constructeur de la classe en utilisant la méthode GetConnectionString. Une fois ces informations extraites, une instance de l'API est initialisée en utilisant l'URL de l'API ainsi que la chaîne d'authentification.

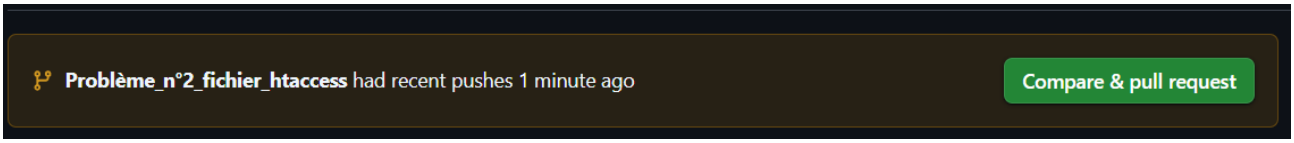
```
/// <summary>
/// Méthode privée pour créer un singleton
/// initialise l'accès à l'API
/// </summary>
1 référence | STL, Il y a 33 minutes | 1 auteur, 3 modifications
private Access()
{

    try
    {
        // Récupérer les informations de connexion depuis App.config
        string authenticationString = GetConnectionString(authenticationName);
        string uriApi = GetConnectionString(connectionName);

        // Initialiser l'instance de l'API
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}
```

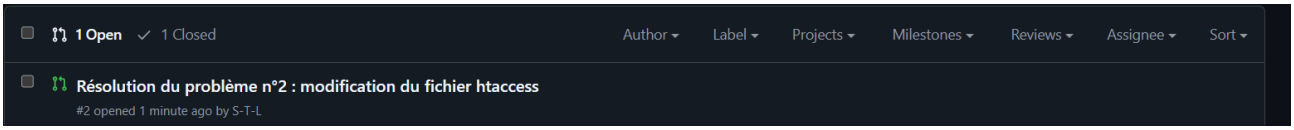
Ajout de la méthode GetConnectionString qui permet de récupérer l'URL de l'API et les identifiants depuis le fichier app.config en utilisant le nom de la chaîne de connexion. Cela sécurise les informations sensibles en les stockant dans le fichier de configuration plutôt que dans le code source.

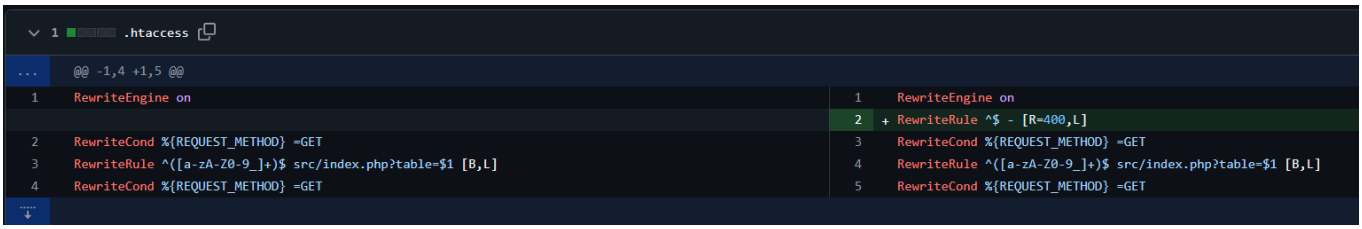
```
/// <summary>
/// Récupération de la chaîne de connexion
/// </summary>
/// <param name="name"></param>
/// <returns></returns>
2 références | STL, Il y a 33 minutes | 1 auteur, 1 modification
static string GetConnectionString(string name)
{
    string returnValue = null;
    ConnectionStringSettings settings = ConfigurationManager.ConnectionStrings[name];
    if (settings != null)
        returnValue = settings.ConnectionString;
    return returnValue;
}
```

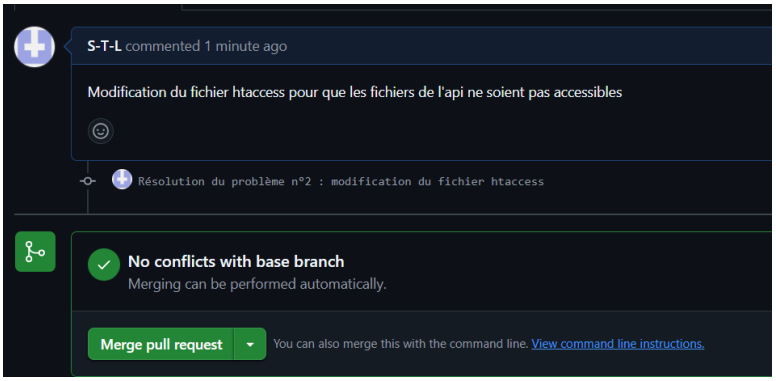
✕ Problème n°2 : Lorsque l'on renseigne l'adresse de l'API dans un navigateur sans mettre de paramètres on obtient la liste des fichiers contenus dans le dossier de l'API. Le but est d'avoir un retour d'erreur.

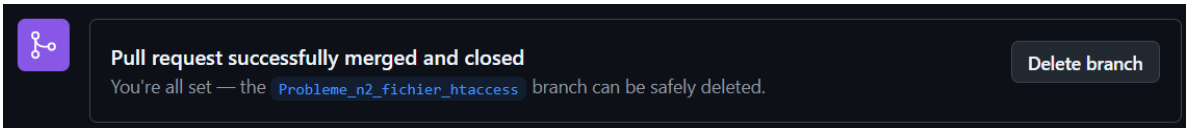
Pull request :



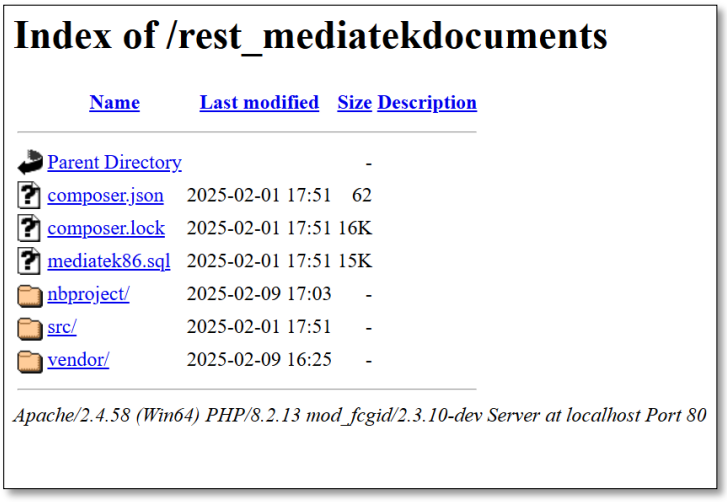








Avant la modification, lorsque l'on entre uniquement l'adresse de l'API sans paramètres, une page comme celle-ci s'affiche :



Index of /rest_mediatekdocuments

Name	Last modified	Size	Description
Parent Directory		-	
composer.json	2025-02-01 17:51	62	
composer.lock	2025-02-01 17:51	16K	
mediatek86.sql	2025-02-01 17:51	15K	
nbproject/	2025-02-09 17:03	-	
src/	2025-02-01 17:51	-	
vendor/	2025-02-09 16:25	-	

Apache/2.4.58 (Win64) PHP/8.2.13 mod_fcgid/2.3.10-dev Server at localhost Port 80

✓ Objectif : Dans le cas d'une arrivée vide on souhaite qu'une erreur 400 soit retournée.

Modification du fichier htaccess dans l'api :

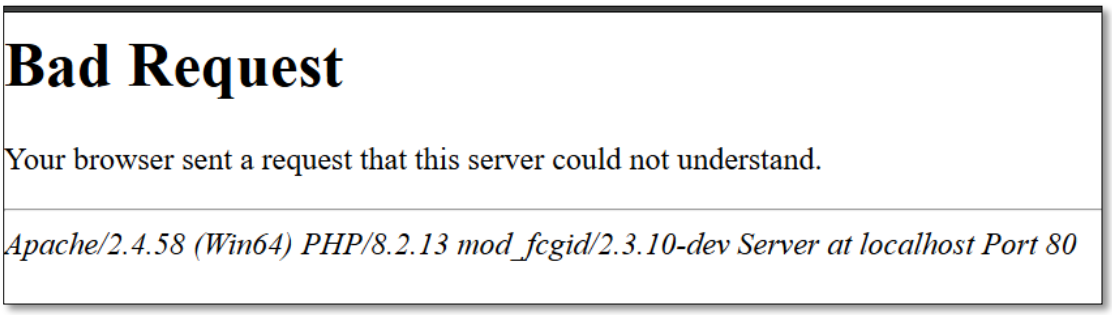
```
RewriteEngine on
RewriteRule ^$ - [R=400,L]
RewriteCond %{REQUEST_METHOD} =GET
RewriteRule ^([a-zA-Z0-9_]+)$ src/index.php?table=$1 [B,L]
RewriteCond %{REQUEST_METHOD} =GET
RewriteRule ^([a-zA-Z0-9_]+)/({.*})$ src/index.php?table=$1&champs=$2 [B,L]
RewriteCond %{REQUEST_METHOD} =POST
RewriteRule ^([a-zA-Z0-9_]+)$ src/index.php?table=$1 [B,L]
RewriteCond %{REQUEST_METHOD} =PUT
RewriteRule ^([a-zA-Z0-9_]+)$ src/index.php?table=$1 [B,L]
RewriteCond %{REQUEST_METHOD} =PUT
RewriteRule ^([a-zA-Z0-9_]+)/([a-zA-Z0-9]+)$ src/index.php?table=$1&id=$2 [B,L]
RewriteCond %{REQUEST_METHOD} =DELETE
RewriteRule ^([a-zA-Z0-9_]+)$ src/index.php?table=$1 [B,L]
RewriteCond %{REQUEST_METHOD} =DELETE
RewriteRule ^([a-zA-Z0-9_]+)/({.*})$ src/index.php?table=$1&champs=$2 [B,L]
```

Pour trouver cette syntaxe je me suis aidée de la documentation d'Apache sur les drapeaux de réécriture :

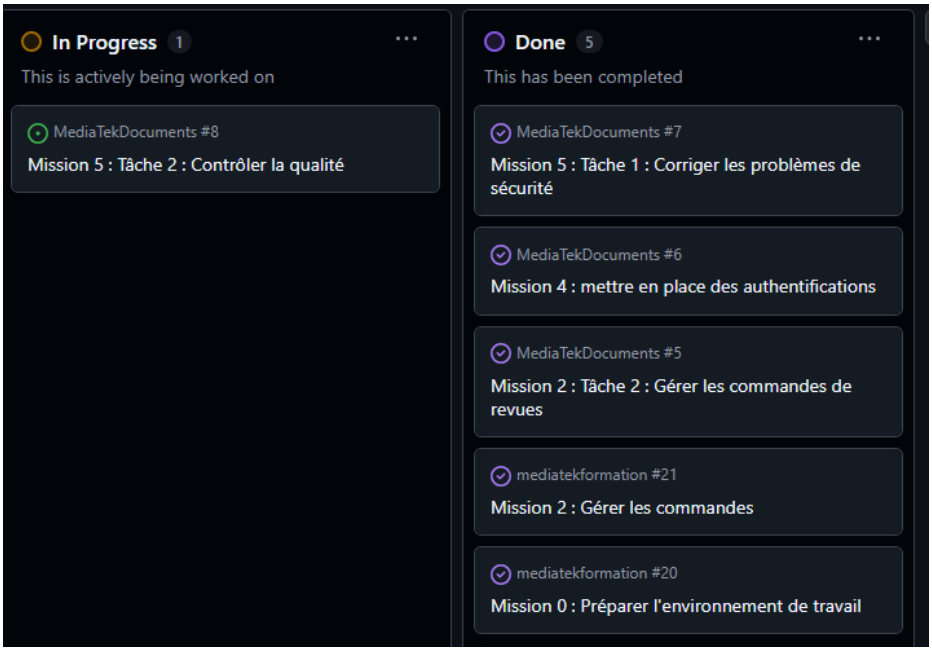
https://httpd.apache.org/docs/current/rewrite/flags.html

et de https://httpd.apache.org/docs/current/rewrite/intro.html#regex pour les règles d'écriture.

Après modification :

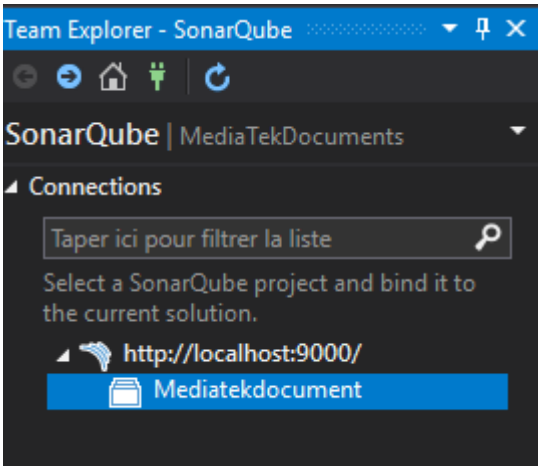


TACHE 2 : CONTROLER LA QUALITE

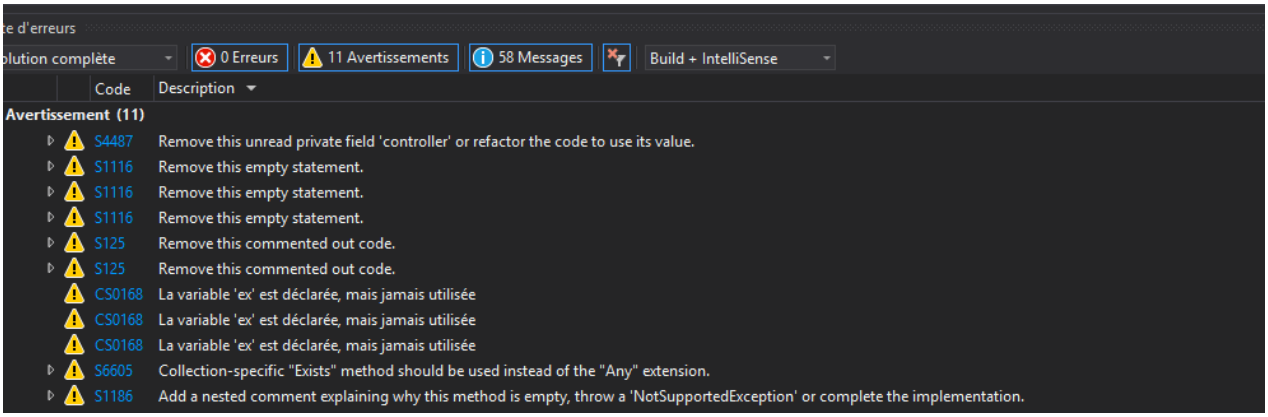


- Utilisation de SonarQube :

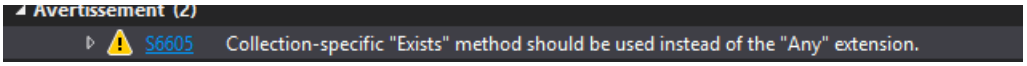
Liaison du projet :



Liste des avertissements :



Exemple pour cette erreur :



Avant modification Any était utilisé :

```
/// <summary>
/// vérifie si l'identifiant renseigné existe déja
/// </summary>
/// <param name="id">identifiant de la commande</param>
/// <returns>true si la commande existe, sinon false</returns>
3 références | STL, il y a 1 jour | 1 auteur, 2 modifications
private bool CommandeExistante(string id)
{
    List<Commande> commandesExistantes = controller.GetAllCommandes();
    if (commandesExistantes.Any(c => c.Id == id))
    {
        MessageBox.Show("Une commande avec ce numéro existe déjà", "Erreur");
        //si commande existe
        return true;
    }
    //si elle n'existe pas
    return false;
}
```

SonarQube me recommande d'utiliser la méthode **Exists**, spécifique aux collections, pour améliorer l'efficacité du code plutôt que **Any**. Je me renseigne sur son fonctionnement et adapte le code : https://learn.microsoft.com/fr-fr/dotnet/api/system.collections.generic.list-1.exists?view=net-9.0

```
/// <summary>
/// vérifie si l'identifiant renseigné existe déja
/// </summary>
/// <param name="id">identifiant de la commande</param>
/// <returns>true si la commande existe, sinon false</returns>
3 références | STL, il y a 1 jour | 1 auteur, 2 modifications
private bool CommandeExistante(string id)
{
    List<Commande> commandesExistantes = controller.GetAllCommandes();
    if (commandesExistantes.Exists(c => c.Id == id))
    {
        MessageBox.Show("Une commande avec ce numéro existe déjà", "Erreur");
        //si commande existe
        return true;
    }
    //si elle n'existe pas
    return false;
}
```

Avant modification :

```
readonly Abonnement abonnement = new Abonnement("1", new DateTime(2025, 3, 1), 10, new DateTime(2025, 4, 1), null);
readonly DateTime dateCommande = new DateTime(2025, 3, 1);
readonly DateTime dateFinAbonnement = new DateTime(2025, 4, 1);


[TestMethod()]

0 références | STL, il y a 3 jours | 1 auteur, 1 modification
public void ParutionDansAbonnementTest()
{
    // Cas 1: La date de parution est exactement la date de commande

    Assert.AreEqual(false, abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 4, 1)), "La date de parutio

    // Cas 2: La date de parution est exactement la date de fin d'abonnement

    Assert.AreEqual(false, abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 5, 1)), "La date de parutio

    // Cas 3: La date de parution est entre la date de commande et la date de fin d'abonnement

    Assert.AreEqual(true, abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 3, 15)), "La date de parutio

    // Cas 4: La date de parution est avant la date de commande

    Assert.AreEqual(false, abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 2, 28)), "La date de paruti

    // Cas 5: La date de parution est après la date de fin d'abonnement

    Assert.AreEqual(false, abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 5, 2)), "La date de parutio
}
```

https://rules.sonarsource.com/csharp/RSPEC-2701/

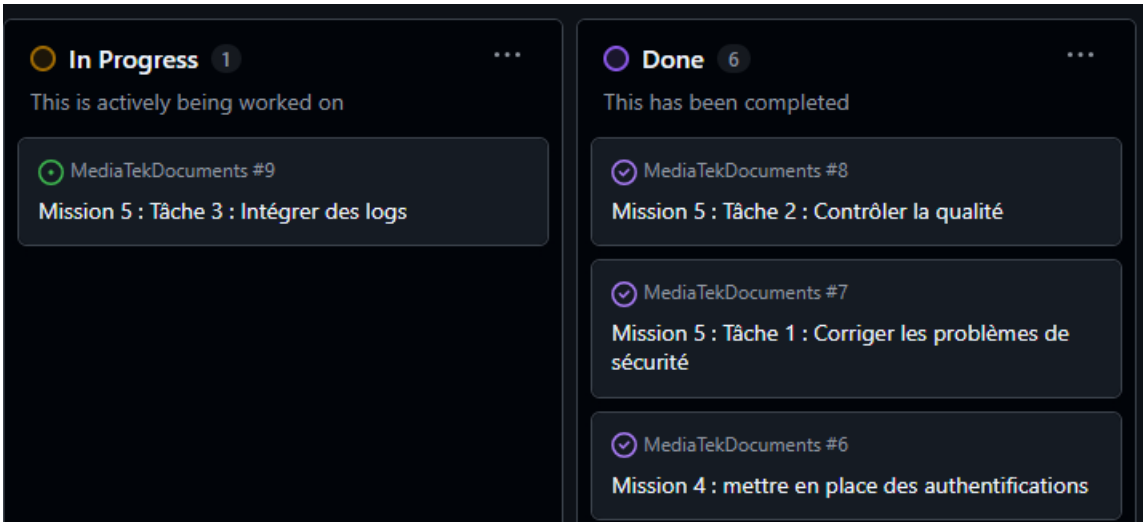
https://rules.sonarsource.com/csharp/RSPEC-6562/

Après :

```
readonly Abonnement abonnement = new Abonnement("1", new DateTime(2025, 3, 1, 0, 0, 0, DateTimeKind.Local), 10, new DateTime(2025, 4, 1, 0, 0, 0, DateTimeKind.Local), null);
readonly DateTime dateCommande = new DateTime(2025, 3, 1, 0, 0, 0, DateTimeKind.Local);
readonly DateTime dateFinAbonnement = new DateTime(2025, 4, 1, 0, 0, 0, DateTimeKind.Local);

[TestMethod()]
| 0 références | STL, il y a 3 jours | 1 auteur, 1 modification
public void ParutionDansAbonnementTest()
{
    // Cas 1: La date de parution est exactement la date de commande
    Assert.IsFalse(abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 4, 1, 0, 0, 0, DateTimeKind.Local)), "La date de parution ne doit pas

    // Cas 2: La date de parution est exactement la date de fin d'abonnement
    Assert.IsFalse(abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 5, 1, 0, 0, 0, DateTimeKind.Local)), "La date de parution ne doit pas

    // Cas 3: La date de parution est entre la date de commande et la date de fin d'abonnement
    Assert.IsTrue(abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 3, 15, 0, 0, 0, DateTimeKind.Local)), "La date de parution devrait être

    // Cas 4: La date de parution est avant la date de commande
    Assert.IsFalse(abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 2, 28, 0, 0, 0, DateTimeKind.Local)), "La date de parution devrait êtr

    // Cas 5: La date de parution est après la date de fin d'abonnement
    Assert.IsFalse(abonnement.ParutionDansAbonnement(dateCommande, dateFinAbonnement, new DateTime(2025, 5, 2, 0, 0, 0, DateTimeKind.Local)), "La date de parution devrait être
}
```

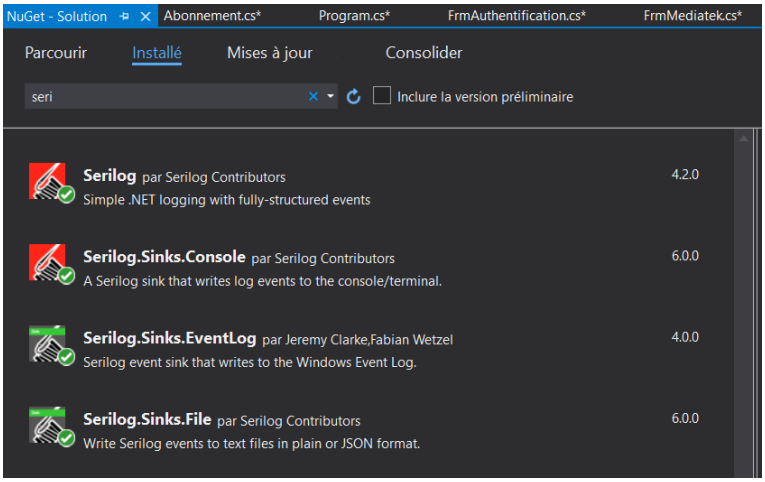
Ici l'utilisation de Assert.IsTrue et Assert.IsFalse rends le test plus explicite concernant les dates.

TACHE 3 : INTEGRER DES LOGS



»» 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).

1. Installation des outils nécessaires :



2. Configuration de Serilog dans le constructeur d'Access :

```
/// <summary>
/// Méthode privée pour créer un singleton
/// initialise l'accès à l'API
/// </summary>
1 référence | STL, il y a 5 jours | 1 auteur, 4 modifications
private Access()
{
    try
    {
        Log.Logger = new LoggerConfiguration()
            .MinimumLevel.Verbose()
            .WriteTo.Console()
            .WriteTo.File("logs/log.txt")
            .CreateLogger();
        // Récupérer les informations de connexion depuis App.config
        string authenticationString = GetConnectionString(authenticationName);
        string uriApi = GetConnectionString(connectionName);

        // Initialiser l'instance de l'API
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Log.Fatal("Access catch erreur={0}", e.Message);
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}
```

Le niveau minimal est configuré sur Verbose afin d'inclure tous les types de logs dans les dispositifs de sortie. Dans ce cas, les logs sont envoyés à la console, et ils sont également enregistrés dans un fichier dédié aux logs, nommé **log.txt**.

Exemple d'ajout de log dans la méthode TraitementRecup :

```
{
    JObject retour = api.RecupDistant(methode, message, parametres);
    // extraction du code retourné
    // Vérifier si retour est NULL
    if (retour == null)
    {
        Log.Error("L'API n'a pas retourné de réponse valide.");
        Console.WriteLine("[ERREUR] L'API n'a pas retourné de réponse valide.");
        return liste;
    }
    // Debug du retour brut de l'API
    Console.WriteLine("[DEBUG] Retour API : " + retour.ToString());
    String code = (String)retour["code"];
    if (code.Equals("200"))
    {
        // dans le cas du GET (select), récupération de la liste d'objets
        if (methode.Equals(GET))
        {
            String resultString = JsonConvert.SerializeObject(retour["result"]);
            // construction de la liste d'objets à partir du retour de l'api
            liste = JsonConvert.DeserializeObject<List<T>>(resultString, new CustomBooleanJsonConverter());
        }
    }
    else
    {
        Log.Error("Code erreur API = {0}, Message = {1}", code, (String)retour["message"]);
        Console.WriteLine("code erreur = " + code + " message = " + (String)retour["message"]);
        Console.WriteLine("Requête envoyée : " + message + " | " + parametres);

    }
}
catch (Exception e)
{
    Log.Fatal("Erreur lors de l'accès à l'API : {0}", e.Message);
    Console.WriteLine("Erreur lors de l'accès à l'API : " + e.Message);
    Environment.Exit(0);
}
```

Retour dans le fichier des logs :

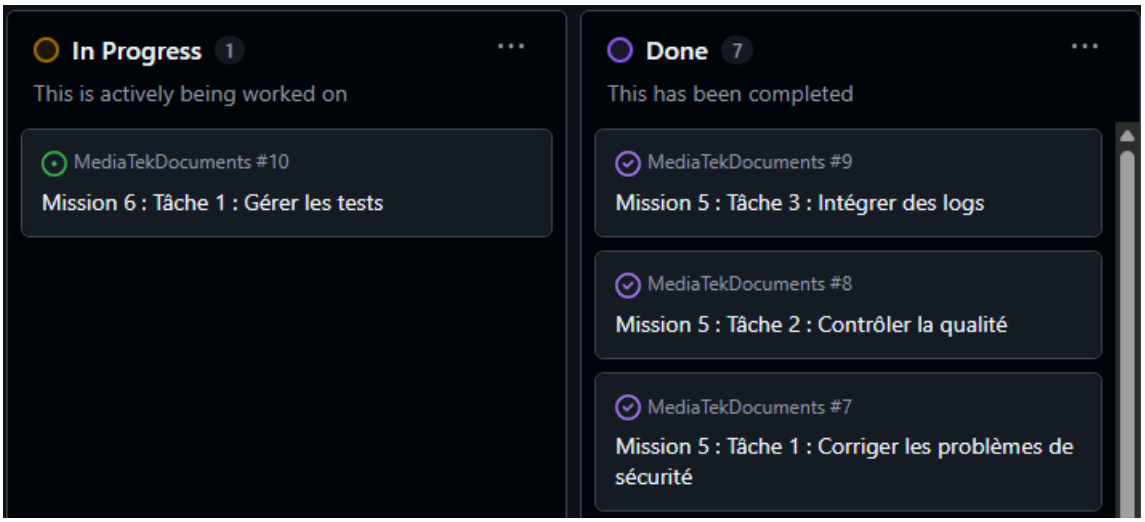
log.tx | Liens c | eclipse | Nouveau Do | Nouve | eclipse.ini

Fichier Modifier Affichage

```
2025-03-25 17:27:23.784 +01:00 [ERR] code erreur = 400 message = requete invalide
2025-03-25 17:27:27.026 +01:00 [ERR] code erreur = 400 message = requete invalide
2025-03-26 08:58:36.213 +01:00 [ERR] Code erreur API = 400, Message = requete invalide
2025-03-26 08:58:41.606 +01:00 [ERR] Code erreur API = 400, Message = requete invalide
```

MISSION 6 : TESTER ET DOCUMENTER

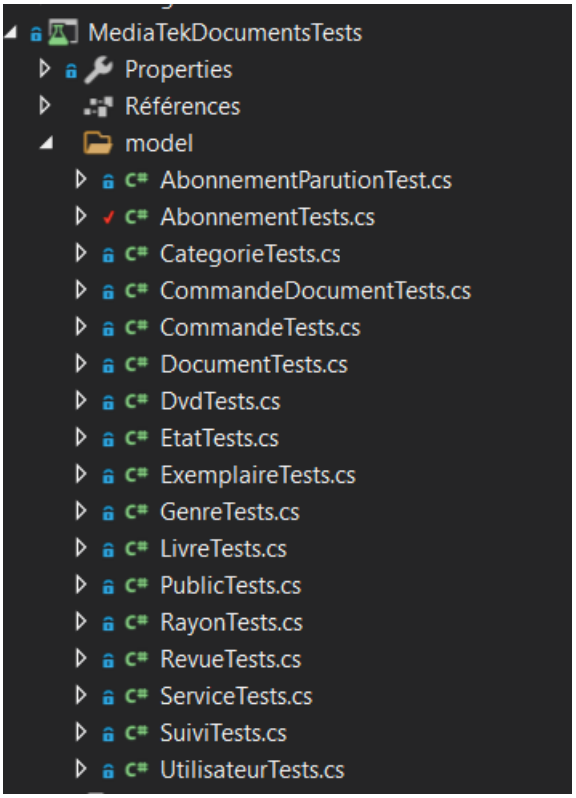
TACHE 1 : GERER LES TESTS

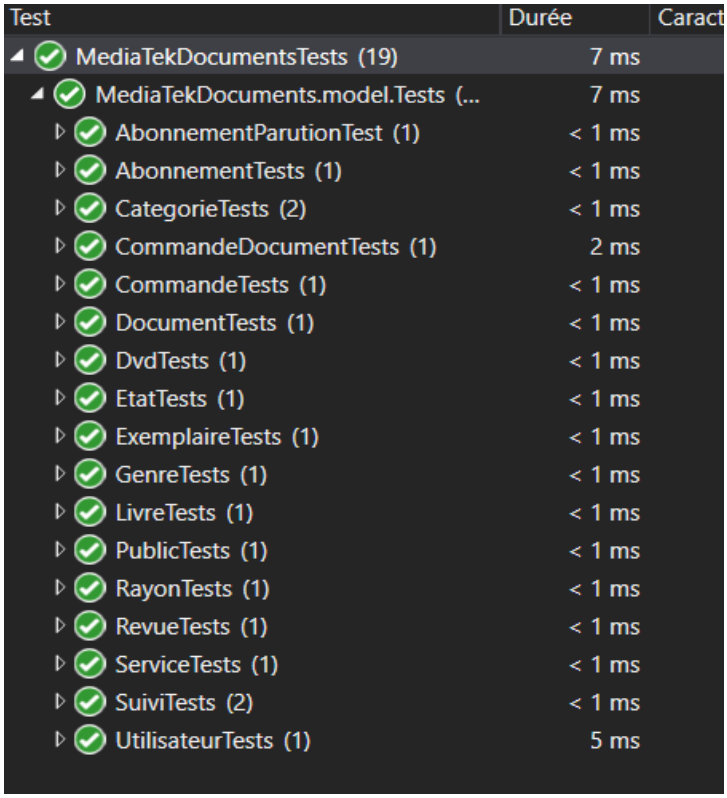


- Écrire les tests unitaires sur les classes du package Model.Le plan des tests unitaires et fonctionnels avec Postman est disponible en annexe de ce document.
- Construire une collection de tests dans Postman pour contrôler les fonctionnalités de l'API d'accès à la BDD.

Le plan des tests unitaires et fonctionnels est disponible en annexe de ce bilan.

Tests unitaires sur les classes du package Model :

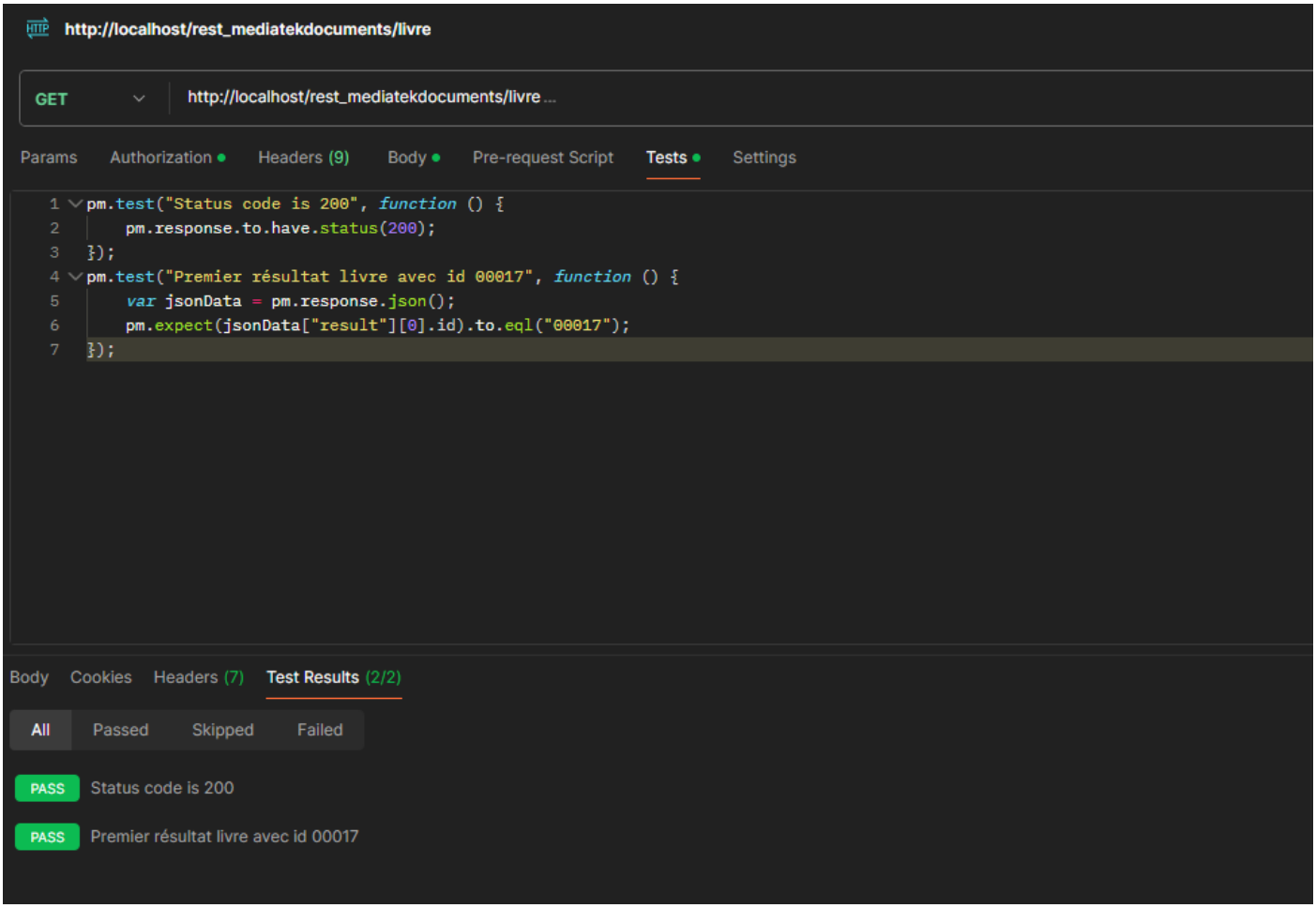


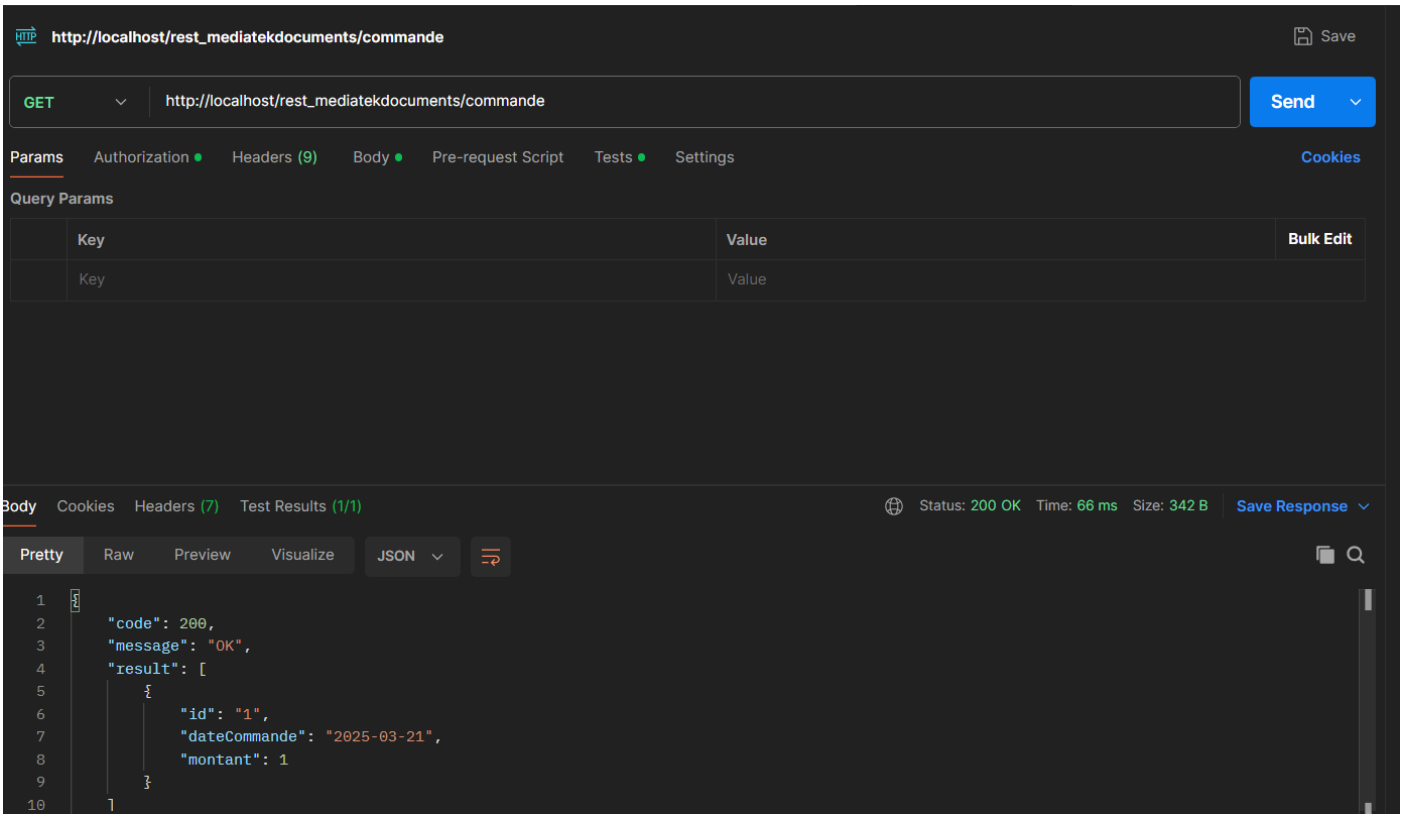


Dans le cas de commandeDocument :

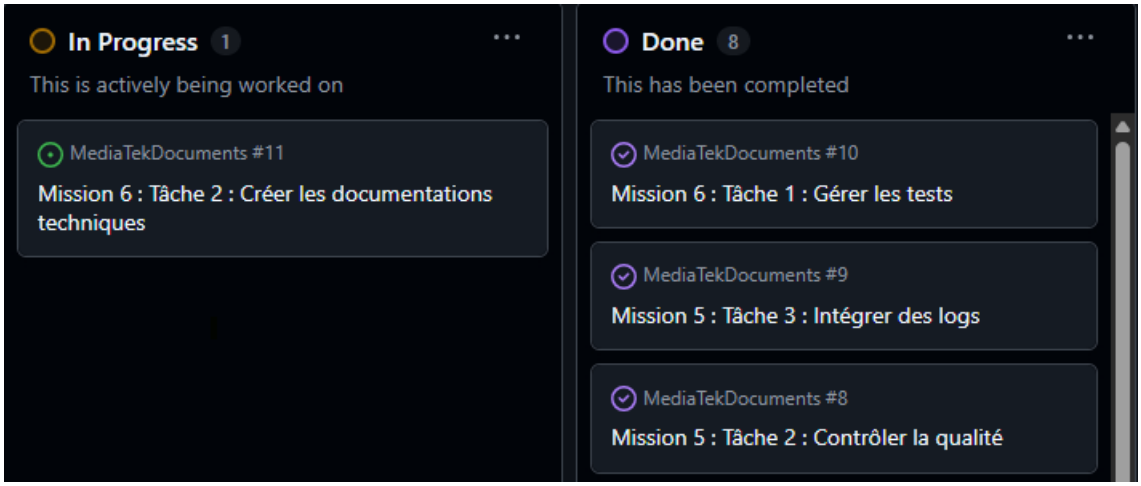
```
namespace MediaTekDocuments.model.Tests
{
    [TestClass()]
    0 références | STL, Il y a 1 heure | 1 auteur, 1 modification
    public class CommandeDocumentTests
    {
        private const string id = "12";
        private static readonly DateTime dateCommande = DateTime.Now;
        private const double montant = 15;
        private const int nbExemplaire = 4;
        private const string idLivreDvd = "00010";
        private const int idSuivi = 1;
        private const string libelle = "en cours";
        private static readonly CommandeDocument commandeDocument = new CommandeDocument(id, dateCommande, montant, nbExemplaire, idLivreDvd, idSuivi, libelle);

        [TestMethod()]
        | 0 références | STL, Il y a 1 heure | 1 auteur, 1 modification
        public void CommandeDocumentTest()
        {
            Assert.AreEqual(id, commandeDocument.Id, "devrait réussir : id valorisé");
            Assert.AreEqual(dateCommande, commandeDocument.DateCommande, "devrait réussir : date de commande valorisée");
            Assert.AreEqual(montant, commandeDocument.Montant, "devrait réussir : montant valorisé");
            Assert.AreEqual(nbExemplaire, commandeDocument.NbExemplaire, "devrait réussir : nombre d'exemplaires valorisé");
            Assert.AreEqual(idLivreDvd, commandeDocument.IdLivreDvd, "devrait réussir : idLivreDvd valorisé");
            Assert.AreEqual(idSuivi, commandeDocument.IdSuivi, "devrait réussir : idSuivi valorisé");
            Assert.AreEqual(libelle, commandeDocument.LibelleSuivi, "devrait réussir : libellé valorisé");
        }
    }
}
```

Test fonctionnel avec Postman :





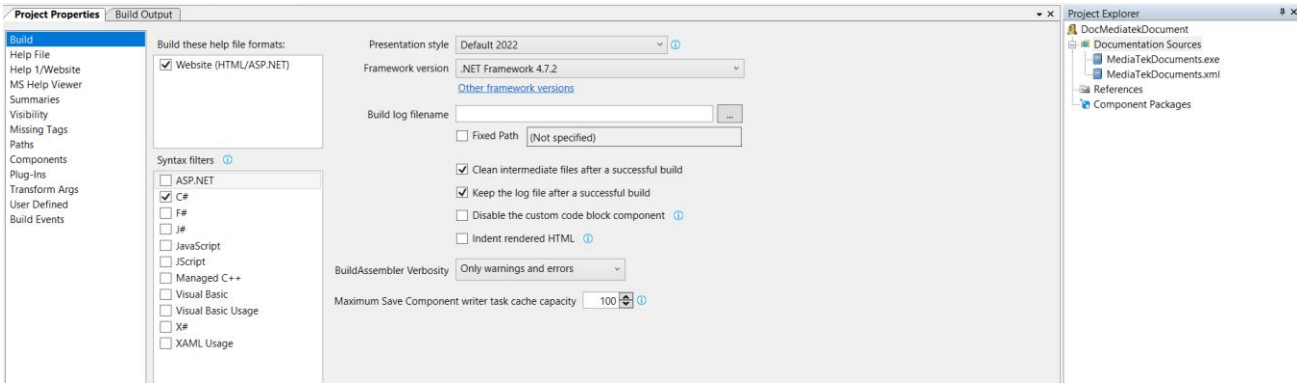
TACHE 2 : CREER LES DOCUMENTATIONS TECHNIQUES



- Contrôler, dans chaque application, que les commentaires normalisés sont bien tous ajoutés et corrects.
- Générer la documentation technique de l'application C#
- Générer la documentation technique de l'API REST
- Transférer les documentations dans les dépôts.

Documentation technique de l'application :

Génération à l'aide de l'outil SandCastle :




```
    Last step completed in 00:00:00,0463
------------------------------
Generating full-text index for the website...

    Last step completed in 00:00:00,2125
------------------------------
Copying website files to output folder...

Copied 270 files for the website content
    Last step completed in 00:00:00,1259
------------------------------
Removing intermediate files...
    Last step completed in 00:00:02,7243
------------------------------

Build completed successfully at 26/03/2025 08:40:52.  Total time: 00:00:06,6267
```

mediatekDocument

IN THIS ARTICLE

Namespaces

MediaTekDocuments.dal	Access
MediaTekDocuments.model	model
MediaTekDocuments.view	view

Documentation de l'API :

doc_rest_mediatekdocuments

Search (Press "/" to focus)

Documentation

Table of Contents

Packages

P Application

Classes

C AccessBDD

Classe qui sollicite ConnexionBDD pour l'accès à la BDD MySQL Elle contient les méthodes appelées par Controle et les méthodes abstraites que MyAccessBDD doit redéfinir pour construire les requêtes

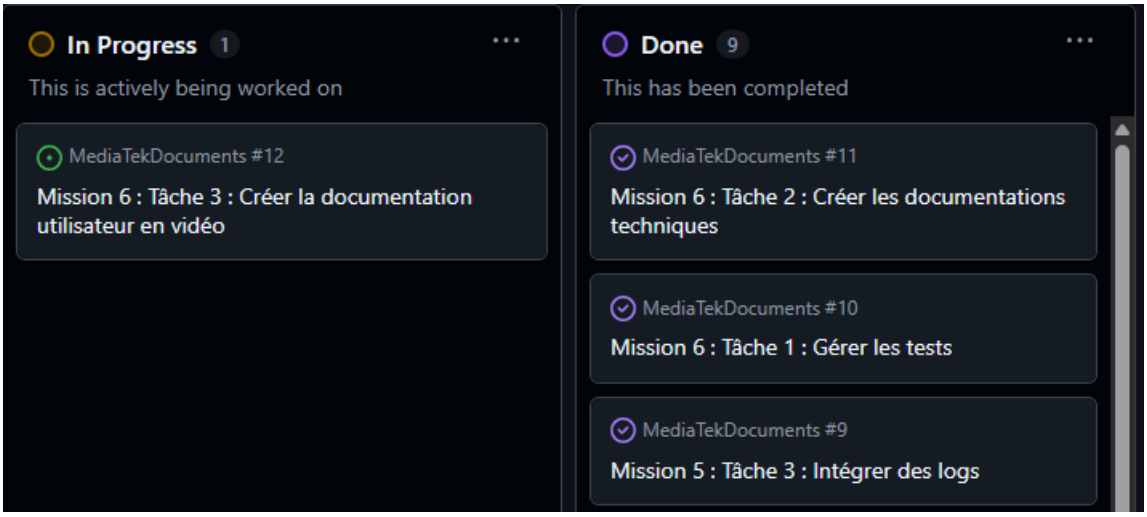
C Connexion

Classe de connexion à la BDD MySQL (singleton) et d'exécution des requêtes en retournant : - pour les requêtes LID : contenu du curseur au format tableau associatif - pour les requêtes LMD : nbre d'enregistrements impactés Dans tous les cs, 'null' est renvoyé si la requête échpie.

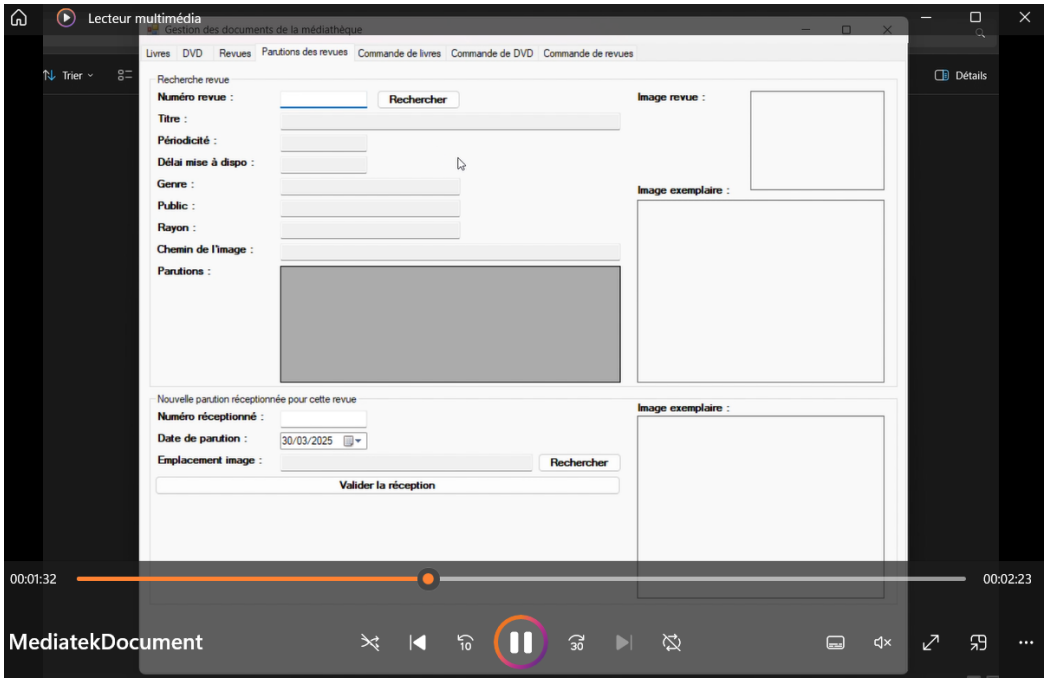
C Controle

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

TACHE 3 : CREER LA DOCUMENTATION UTILISATEUR

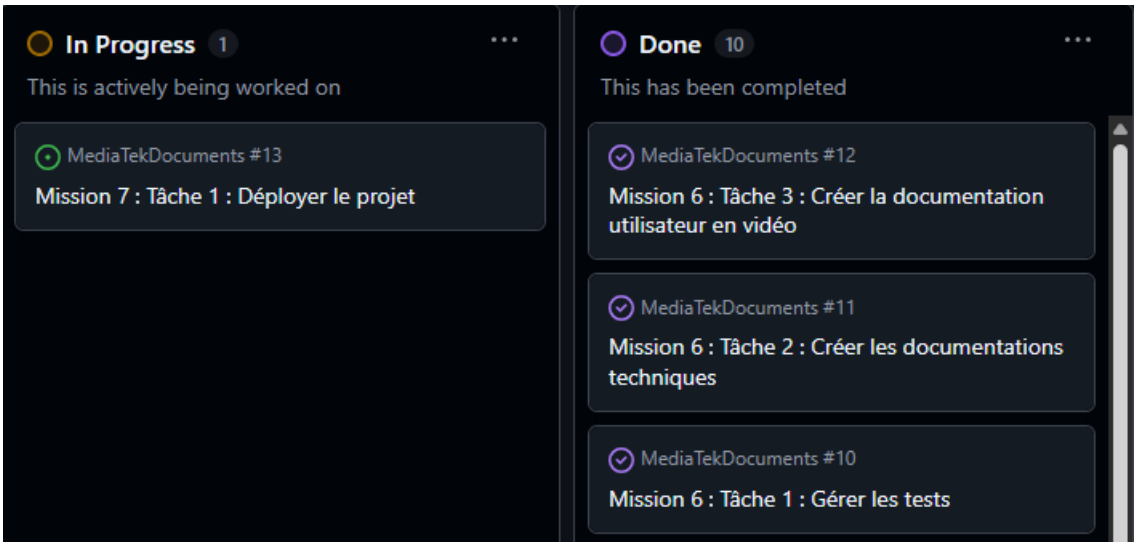


⋙ Créer une vidéo de 10mn maximum qui présente l'ensemble des fonctionnalités de l'application C#.



MISSION 7 : DEPLOYER ET GERER LES SAUVEGARDES DE DONNEES

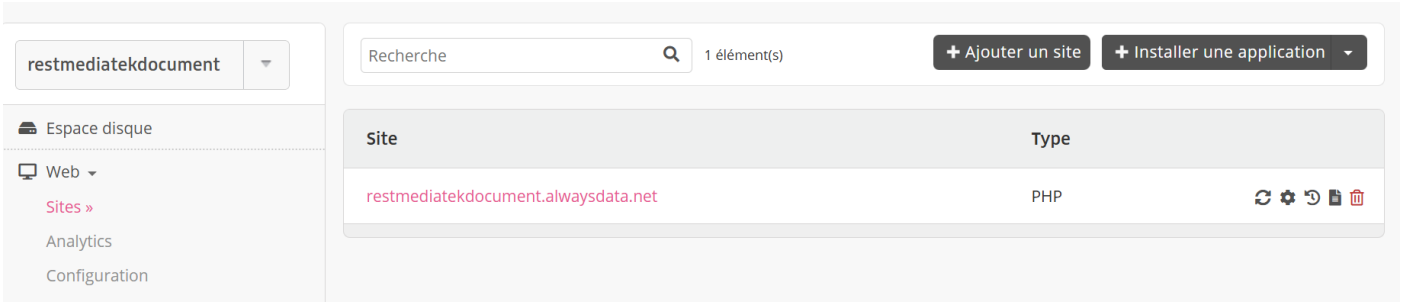
TACHE 1 : DEPLOYER LE PROJET



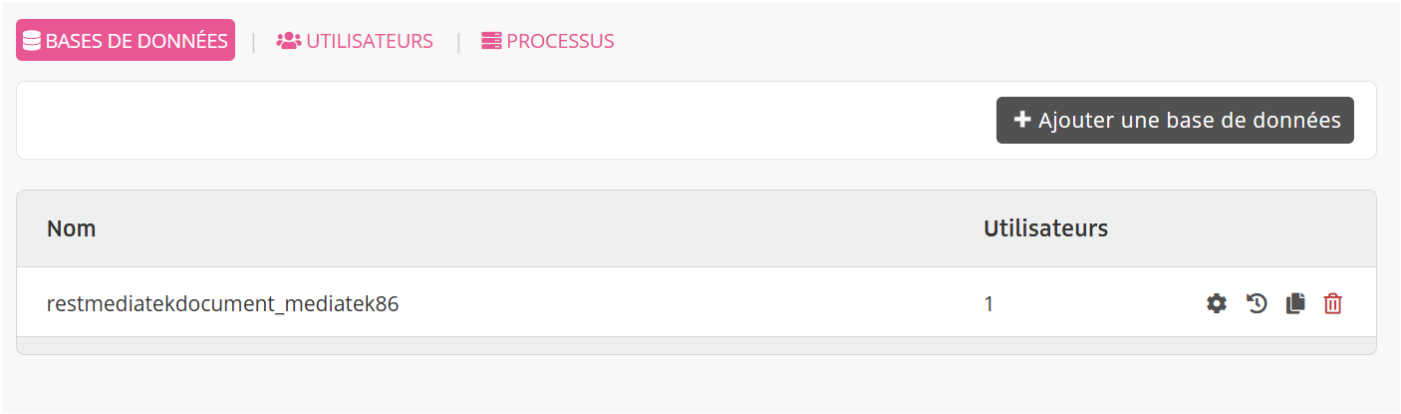
METTRE EN LIGNE L'API :

- Renforcer la sécurité de l'API : changer les username/password de l'accès à l'API (les informations de connexions à la BDD, mises dans le fichier '.env', devront être modifiées en tenant compte des informations prises chez l'hébergeur).
- Déployer l'API et la BDD.
- Tester l'API avec Postman

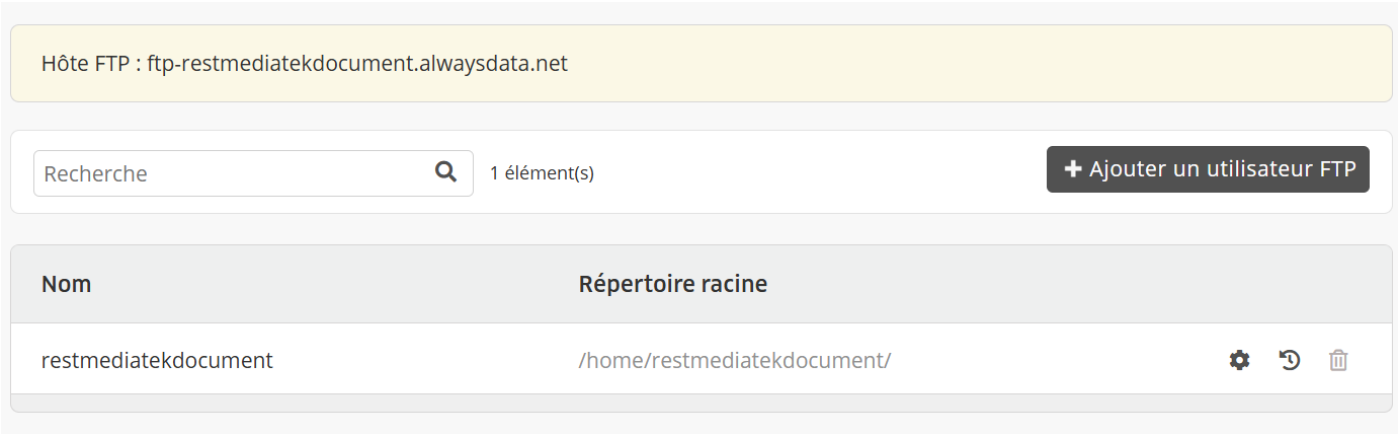
Après modifications du fichier .env et des données de connexions à la base de données, hébergement de l'API chez alwaysdata :



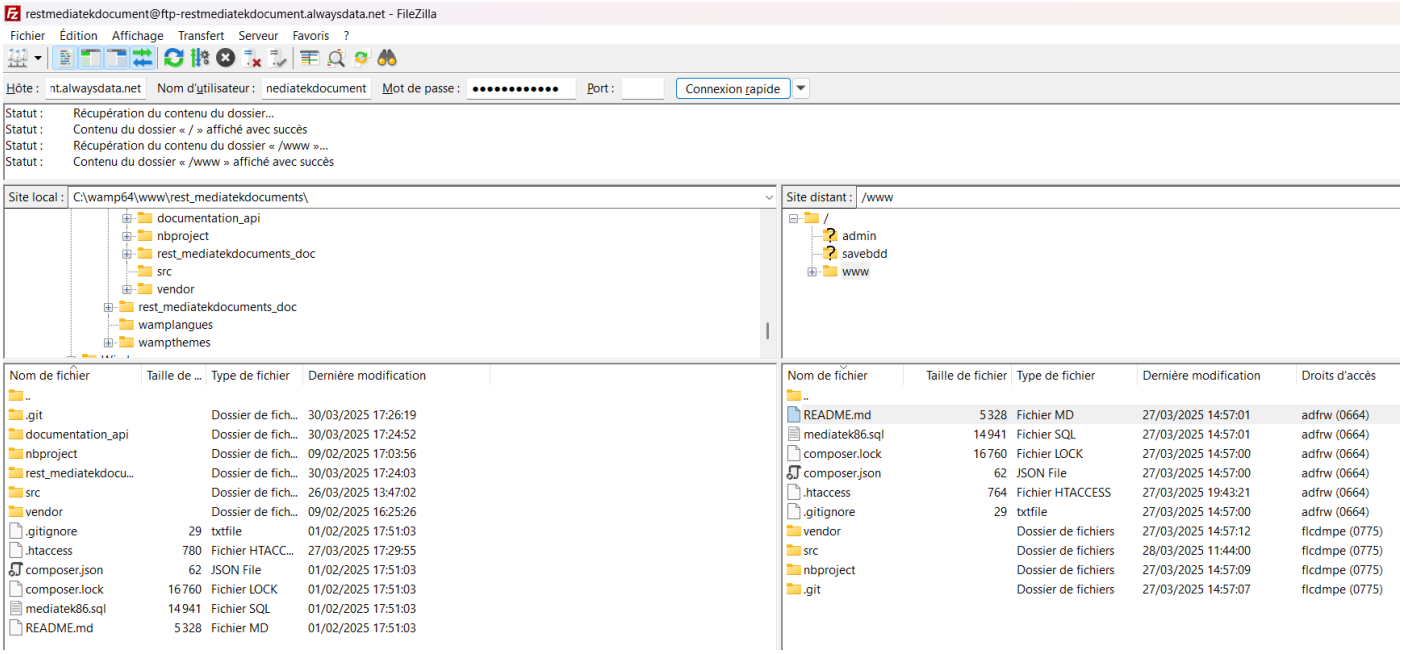
Création de la base de données et de son utilisateur en important la base locale :



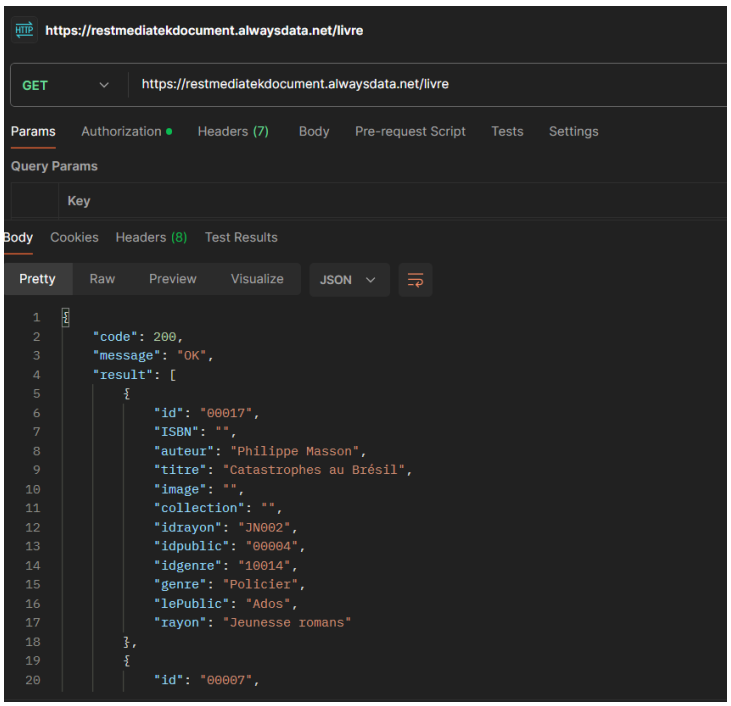
Création d’un accès FTP afin de pouvoir transférer l’api locale vers l’hébergeur :



Transfert des fichiers avec fileZilla :



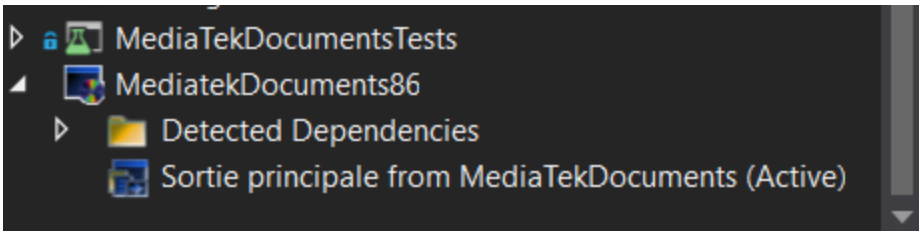
Test de l’API sur Postman :



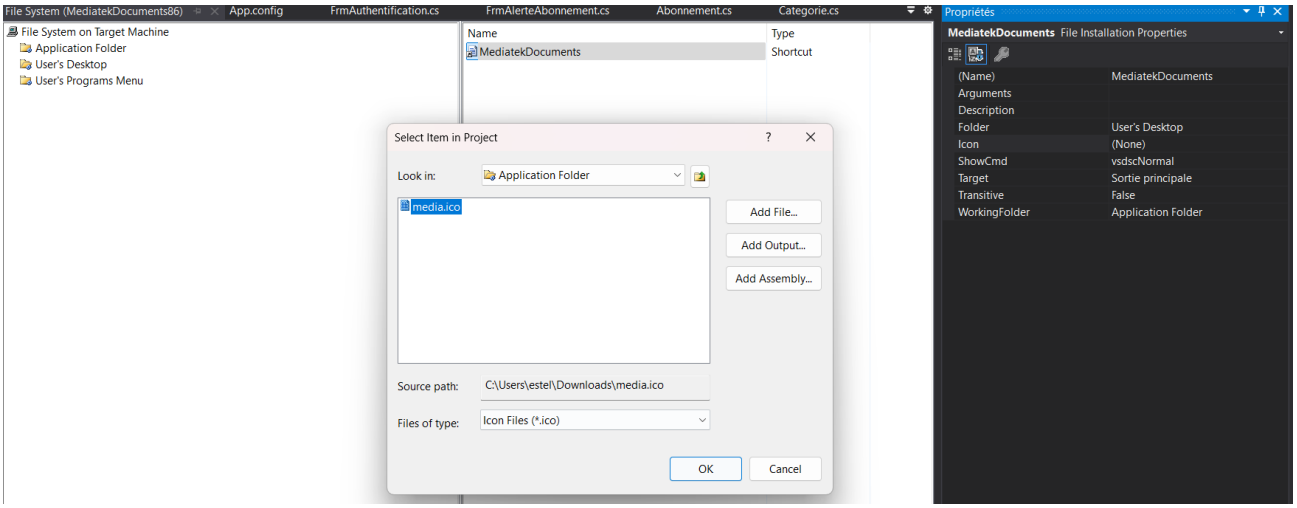
CREER UN INSTALLATEUR POUR L'APPLICATION C# :

- Modifier le code pour l'accès à l'API distante
- Créer l'installeur, tester son installation et l'utilisation de l'application installée.
- L'envoyer vers le dépôt.

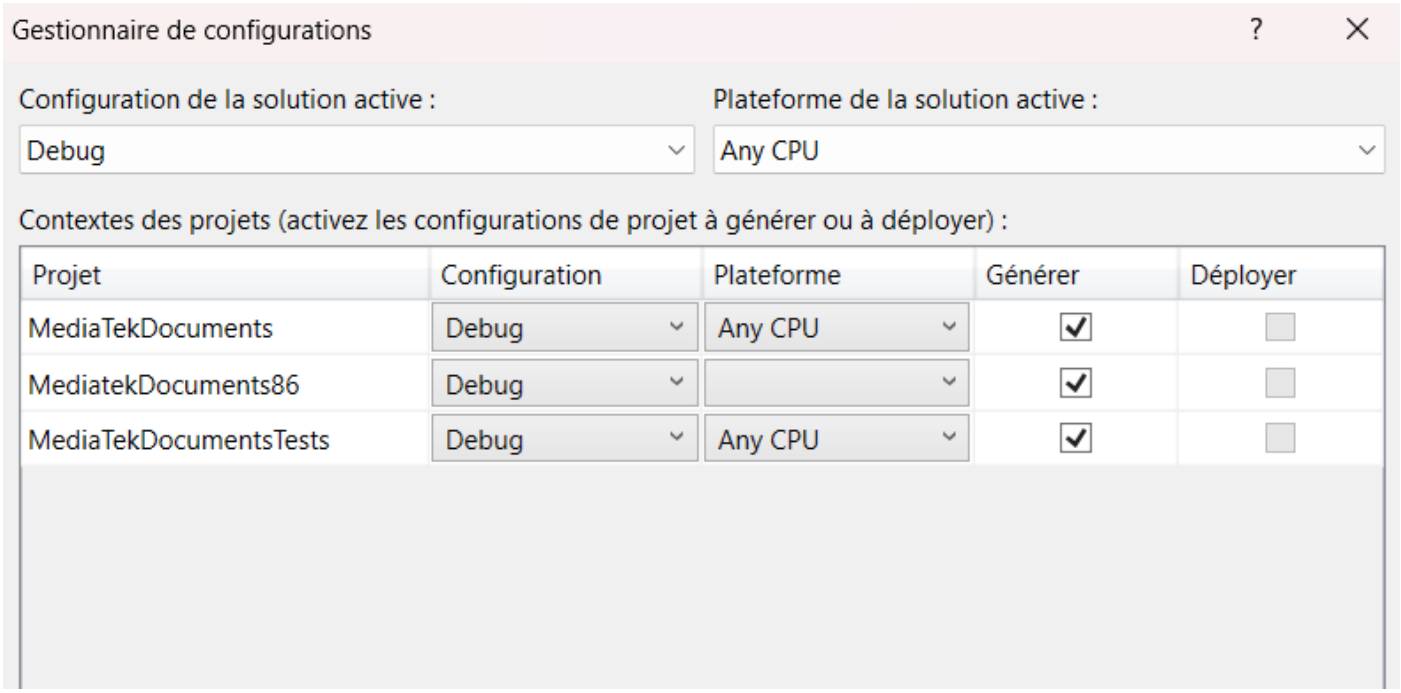
1. Ajout d'un nouveau projet Setup « MediatekDocuments86 » :



2. Ajout d'une icône pour le projet :



3. Génération du setup Debug / Release :



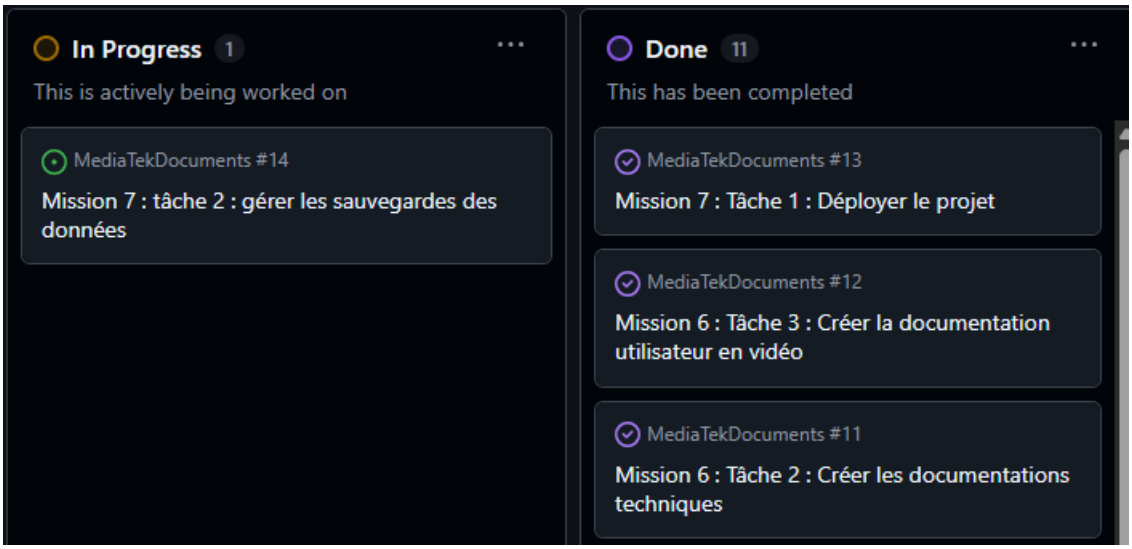
4. Génération des 2 fichiers d'installation : setup.exe et MediatekDocument86.msi

Nom	Modifié le	Type	Taille
MediatekDocuments86	31/03/2025 10:33	Package Windows Installer	3 180 Ko
setup	31/03/2025 10:33	Application	557 Ko

5. Envoi des fichiers d'installation vers le dépôt Github :

```
C:\Users\estel\Desktop\BTS SIO\2annee\ATELIERS PRO\ATELIER 2 C#\MediaTekDocuments>git add "Fichiers d'installation"

C:\Users\estel\Desktop\BTS SIO\2annee\ATELIERS PRO\ATELIER 2 C#\MediaTekDocuments>git commit -m "Ajout du dossier contenant les fichiers nécessaires à l'installation de l'application"
[master 7da039e] Ajout du dossier contenant les fichiers nécessaires à l'installation de l'application
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Fichiers d'installation/MediatekDocuments86.msi
 create mode 100644 Fichiers d'installation/setup.exe

C:\Users\estel\Desktop\BTS SIO\2annee\ATELIERS PRO\ATELIER 2 C#\MediaTekDocuments>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.93 MiB | 209.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/S-T-L/MediaTekDocuments.git
   b5426a1..7da039e  master -> master

C:\Users\estel\Desktop\BTS SIO\2annee\ATELIERS PRO\ATELIER 2 C#\MediaTekDocuments>
```

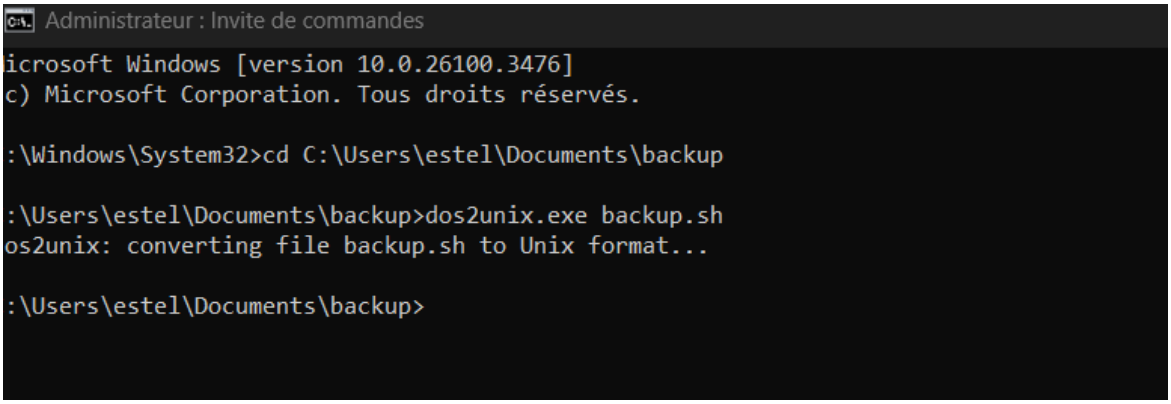
TACHE 2 : GERER LES SAUVEGARDES DES DONNEES



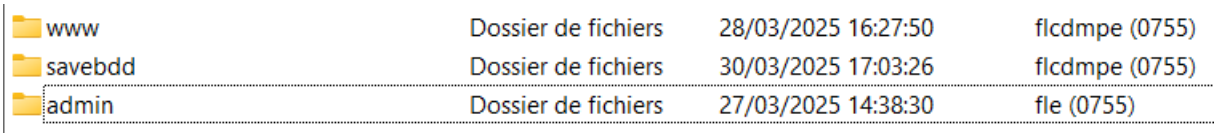
⋙ Une sauvegarde journalière automatisée doit être programmée pour la base de données.

1. Création d'un fichier de script nommé backup.sh puis conversion au format Linux à l'aide de dos2unix

Nom	Modifié le	Type	Taille
backup	28/03/2025 16:59	Shell Script	1 Ko
dos2unix	22/01/2024 21:36	Application	102 Ko



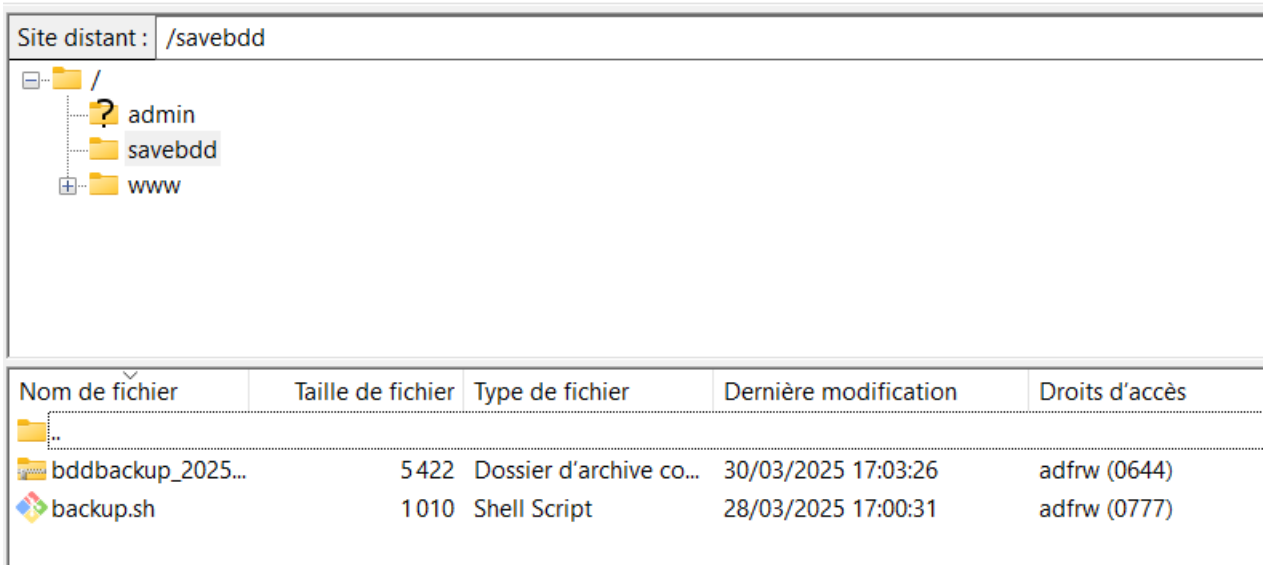
2. Transfert de ce fichier vers alwaysdata dans un dossier « savebdd » en racine du projet



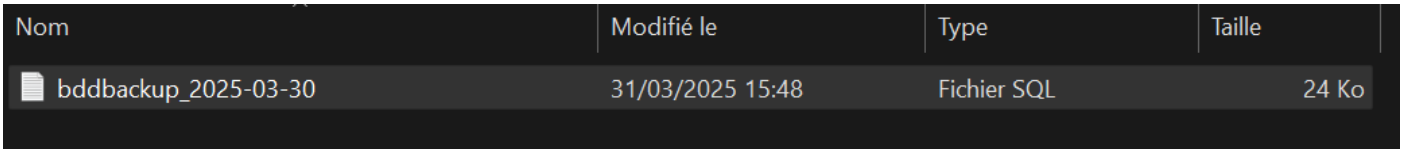
3. Création d'une tâche automatisée chez l'hébergeur :

4. Choix de la périodicité :

5. Vérification de la tache automatisée : un dossier zippé portant le nom de backup avec la date du jour est bien créé :



6. Si je télécharge ce fichier je récupère bien toutes les données :



Bilan

Après avoir finalisé l'atelier 1 sur Symfony, j'ai rencontré quelques difficultés à me replonger dans la syntaxe du C#, ce qui m'a demandé un certain temps d'adaptation, notamment pour comprendre le code existant.

Pour m'aider, je me suis appuyé sur le référentiel du langage ainsi que sur les tutoriels disponibles sur Microsoft Learn. J'ai parfois du effectuer plusieurs tests avant d'obtenir le résultat souhaité, mais plus j'avançais dans le développement, plus j'avais envie d'expérimenter. Cela m'a permis d'apprendre de nouvelles choses même si je pense que certaines parties du code pourraient encore être optimisées.

J'ai rencontré quelques difficultés qui m'ont fait perdre du temps au moment du déploiement de l'API à cause d'une différence d'encodage sur une de mes tables de la base de données reliées à la table utilisateur, ce qui bloquait totalement l'authentification.

Malgré un temps de doute et une petite déception de ne pas avoir pu réaliser les missions optionnelles en temps voulu (travaillant seule), et en dépit des difficultés rencontrées sur cet atelier, j'ai pris plaisir à coder et à résoudre les erreurs qui se sont présentées à moi.

Contexte : MediaTek86

Application : MediatekDocuments (application de bureau C#) exploitant rest_mediatekdocuments (API REST en PHP)

ANNEXE - PLAN DE TESTS

TESTS UNITAIRES SUR LES CLASSES DU PACKAGE MODEL

But du test	Action de contrôle	Résultat attendu	Bilan
Contrôler la méthode ToString() de la classe Categorie pour voir si elle retourne le libelle.	Test unitaire lancé après avoir créé un objet de type Catégorie avec libelle contenant : "Horreur"	"Horreur"	OK
Vérifier l'instanciation de la classe Abonnement	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Catégorie	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe CommandeDocument	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Commande	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Dvd	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Etat	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Exemplaire	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Genre	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe livre	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Public	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Rayon	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Revue	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK

Vérifier l'instanciation de la classe Service	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Vérifier l'instanciation de la classe Suivi	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK
Contrôler la méthode ToString de la classe Suivi	Test unitaire lancé après avoir créé un objet de type Suivi avec le libellé « délai modifié »	« Délai modifié »	OK
Vérifier l'instanciation de la classe Utilisateur	Test unitaire après création d'un nouvel objet en valorisant ses propriétés	Propriétés valorisées	OK

TESTS FONCTIONNELS DANS POSTMAN (FONCTIONNALITES DE L'API)

But du test	Action de contrôle	Résultat attendu	Bilan
Test sur les commandes	GET http://localhost/rest_mediatekdocuments/commande	{ "code": 200, "message": "OK", "result": [{ "id": "1", "dateCommande": "2025-03-21", "montant": 1 }, { "id": "2", "dateCommande": "2025-03-24", "montant": 2 }, { "id": "3", "dateCommande": "2025-03-21", "montant": 3 }, { "id": "5",	OK

		"dateCommande": "2025-03-25", "montant": 120 }] }	

Test de l'ajout d'une commande de livre	POST http://localhost/rest_mediatekdocuments/commandedocument et champs:{"Id": "4","DateCommande": "2025-01-10","Montant": "120.00","NbExemplaire": "3","IdLivreDvd": "00017","IdSuivi": "1"}	{
		"code": 200,
		"message": "OK",
		"result": 1
		}
			OK

Test de l'ajout d'une nouvelle commande à l'abonnement d'une revue	POST http://localhost/rest_mediatekdocuments/abonnement et champs:{"Id": "5","DateCommande": "2025-03-25","Montant": "120.00","DateFinAbonnement": "2025-04-25","IdRevue": "10001"}	{
		"code": 200,
		"message": "OK",
		"result": 1
		}
			OK

Test de modification du suivi d'une commande	PUT http://localhost/rest_mediatekdocuments/commandedocument/4 et champs:{"idSuivi": 2}	{
		"code": 200,
		"message": "OK",
		"result": 1
		}
			OK

Test de suppression d'une commande	DELETE http://localhost/rest_mediatekdocuments/commandedocument et champs:{"id": "4"}	{
		"code": 200,
		"message": "OK",
		"result": 1
		}
			OK

Test de suppression d'un abonnement à une revue	DELETE http://localhost/rest_mediatekdocuments/abonnement et champs:{"id":"3"}	{
		"code": 200,
		"message": "OK",
		"result": 1
		}
			OK