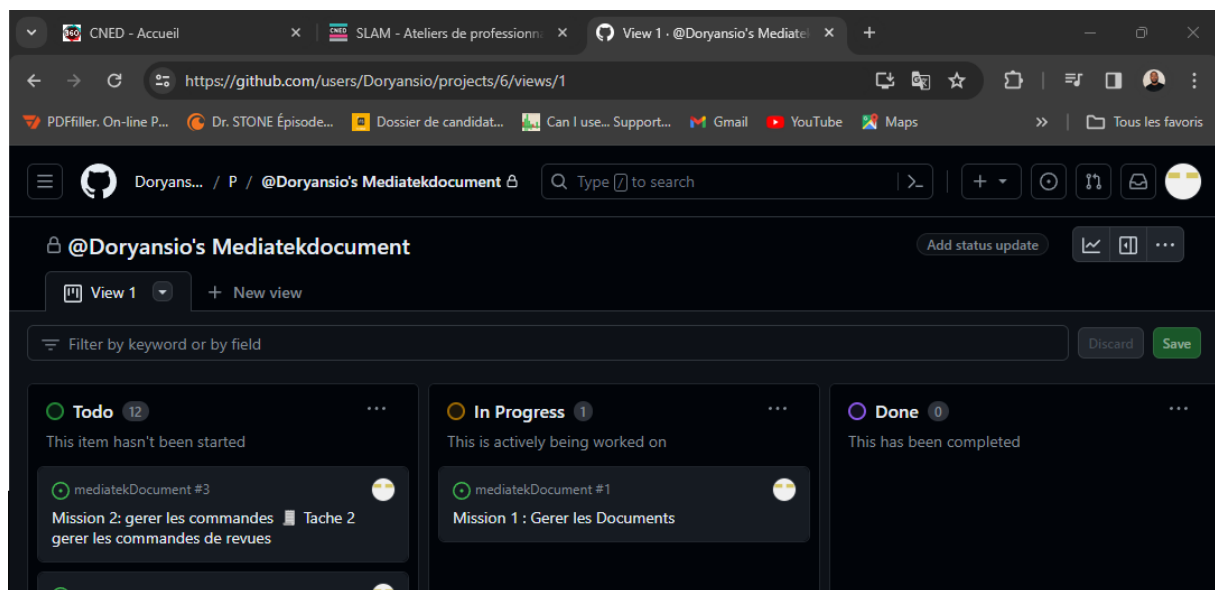


MediatekDocument

MediatekDocument.....	1
Mission 1 (facultative) : gérer les documents	2
Mission 2 gerer les commandes.....	12
Mission 4 : mettre en place des authentifications.....	19
Mission 5 : Assurer la sécurité, la qualité et intégrer des logs	26
Intégration des logs	28
Mission 6 : gérer les tests	29
Création de la documentation technique	31
Déployer et gérer les sauvegardes de données	33
Conclusion	34
Problèmes rencontrés.....	34

Mission 1 (facultative) : gérer les documents



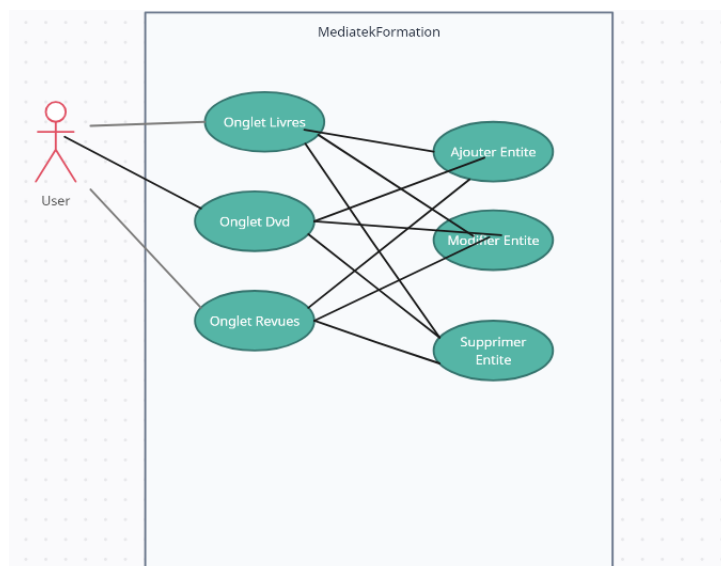
Dans les onglets actuels (Livres, Dvd, Revues), ajouter les fonctionnalités (boutons) qui permettent d'ajouter, de modifier ou de supprimer un document. Un document ne peut être supprimé que s'il n'a pas d'exemplaire rattaché, ni de commandes. La modification d'un document ne peut pas porter sur son 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.

Temps estimé : 8

Temps de réalisation : 9h

Voici le cas diagramme des cas d'utilisation de cette mission



Cette mission consiste a compléter l'interface visuelle des onglets livres, dvd et revues et de permettre l'ajout la modification ou la suppression d'un document. Ces modifications dans l'application seront aussi effectives dans la dans la base de données a travers une l'Api « rest_mediatekdocument »

Gestion des documents de la médiathèque

Livres DVD Revues Parutions des 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

Ou sélectionner le rayon : X

Id	Titre	Auteur	Collection	Genre	Public	Rayon
00007	Dans les coulisses du musée	test	test	Roman	Tous publics	Littérature étrangère
00006	La marque jaunâtre	Edgar P. Jacobs	Blake et Mortimer	Bande dessinée	Tous publics	BD Adultes
00026	La planète des singes	Pierre Boulle	Julliard	Science Fiction	Tous publics	Littérature française
00012	La souris bleue	Kate Atkinson		Roman	Tous publics	Littérature française
00016	Le butin du requin	Julian Press		Policier	Ados	Jeunesse romans
00018	Le Routard - Maroc		Guide du Routard	Voyages	Tous publics	Voyages
00023	Le secret du janissaire	Ayrolles - Maabou	De cape et de crocs	Bande dessinée	Adultes	BD Adultes
00010	Le vestibule des causes perdues	Manon Moreau		Roman	Adultes	Littérature étrangère

Informations détaillées

Numéro de document : 00007 Code ISBN :

Titre : Dans les coulisses du musée

Auteur(s) : test

Collection : test

Genre : Roman

Public : Tous publics

Rayon : Littérature étrangère

Chemin de l'image :

Image :

Ajouter Modifier Supprimer

Annuler Valider

Voici la maquette de la page de livres avec les boutons ajouter (la page des dvd et des revues est similaire)

Sur l'onglet livres nous avons donc ajouter 5 boutons : Ajouter, Modifier, Supprimer, Annuler et Valider. De plus nous avons aussi changer les textbox de Genre public et rayon pour les remplacer par des Combo box.

Nous allons voir les propriétés de chacun de ses boutons et Combo box, les classes qu'elles impactent et les méthodes ajoutées. Certaines méthodes seront similaires du coup nous montreront les différences si elles en ont.

boutons ajouter

les boutons « ajouter » des onglet livres, dvd, et revues ont des noms par lesquelles nous pourront les appeler et un évènement lorsque l'on cliquera dessus. Ici les boutons d'ajout se nommeront BtnAjouterLivre BtnAjouterDvd et BtnAjouterRevues. Cependant c'est par les evenements qui seront appeler dans le fichier frmMediatek.cs BtnAjouterLivre_Click BtnAjouterDvd_Click et BtnAjouterRevues_Click

```

/// <summary>
/// enclanche la procédure d'ajout de livre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | Doryan, Il y a 11 heures | 1 auteur, 1 modification
private void BtnAjouterLivres_Click(object sender, EventArgs e)
{
    LivreEnCoursDeModif(true);
    txbLivresNumero.ReadOnly = false;
    VideLivresInfos();
}

```

Cette methode appelle une autre methode booleene « LivreEncoursModif » avec en parametre true. Voici la methode :

```
/// <summary>
/// applique des droits sur l'interface en fonction de la situation
/// </summary>
/// <param name="modif"></param>
5 références | Doryan, Il y a 11 heures | 1 auteur, 1 modification
private void LivreEnCoursDeModif(bool modif)
{
    BtnAjouterLivres.Enabled = !modif;
    BtnSupprimerLivres.Enabled = !modif;
    BtnModifierLivres.Enabled = !modif;
    BtnAnnulerChoix.Enabled = modif;
    BtnValiderChoix.Enabled = modif;
    txbLivresTitre.ReadOnly = !modif;
    txbLivresAuteur.ReadOnly = !modif;
    cbxLivresPublicInfos.Enabled = modif;
    txbLivresIsbn.ReadOnly = !modif;
    txbLivresCollection.ReadOnly = !modif;
    cbxLivresGenreInfos.Enabled = modif;
    cbxLivresRayonInfos.Enabled = modif;
    txbLivresImage.ReadOnly = !modif;
    txbLivresNumero.ReadOnly = true;
    dgvLivresListe.Enabled = !modif;
}
```

Cette méthode permet de rendre accessible ou non certains boutons dans l'interface, en parametre nous avons une valeur booleene « modif ». nous voyons déjà que les boutons d'ajout, de modification et de suppression valent le contraire de modif : ils ne sont donc pas accessible contrairement aux boutons annuler et valider.

Cette methode rend accessible les champs dans le groupbox « informations détaillé » des livres excepté le champ txbLivresNumero qui est en ReadOnly. Car cette methode sera aussi utile pour la modification d'un livre qui ne porte pas sur l'id.

Mais dans la methode d'ajouter nous avons retirer le ReadOnly du txbLivresNumero pour permettre de créer une nouvelle entité donc avec un nouvel id. de plus nous faisons appel a la methode videLivresInfos qui reinitialise tout les champs.

Pour les dvd et les revues les méthodes d'ajout sont similaire pour ne pas dire identiques sauf dans le noms des méthodes evidement.

Boutons modifier

Cette methode est simple : elle fait juste appel a la methode LivreEncoursDeModif mentionné au dessus mais avec la valeur boolene true. Pareil pour les autres entité.

```

368
369     /// <summary>
370     /// enclanche la procédure de modification de livre
371     /// </summary>
372     /// <param name="sender"></param>
373     /// <param name="e"></param>
374     1 référence | Doryan, Il y a 11 heures | 1 auteur, 1 modification
375     private void BtnModifierLivres_Click(object sender, EventArgs e)
376     {
377         LivreEnCoursDeModif(true);
378     }
379     /// <summary>

```

Boutons Supprimer

Cette methode requiert un peu plus d'explications

```

/// <summary>
/// enclanche la procédure de suppression de livre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
référence | Doryan, Il y a 11 heures | 1 auteur, 1 modification
private void BtnSupprimerLivres_Click(object sender, EventArgs e)
{
    Livre leLivre = (Livre)bdgLivresListe.List[bdgLivresListe.Position];
    if (MessageBox.Show($"Etes vous sur de vouloir supprimer {leLivre.Titre} de {leLivre.Auteur} ?",
        "Validation suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        // fonction a modifier pour prendre en charge le fait que l'on ne pourra pas supprimer un l
        if (controller.SupprimerLivre(leLivre))
        {
            lesLivres = controller.GetAllLivres();
            RemplirLivresListeComplete();
        }
        else
        {
            MessageBox.Show("Erreur");
        }
    }
}

```

Tout d'abord nous allons valoriser un variable locale de type Livre(classe métier) avec la liste des livres que l'ont a encapsulé dans une binding source et la positions de l'elements lelivre que l'on souhaite supprimer. L'application nous renvoie donc une messageBox pour confirmer que l'on souhaite supprimer le livre. Si nous confirmons la message box. Un appel a la methode supprimerLivre du controller est faite avec comme parametre leLivre (nous aborderons cette methode dans la partie « controller ». si cette methode est appelé et opérationnelle. La liste des livre est actualisé. Dans le cas contraire nous avons une message d'erreur. La forme de cette methode est commune aux autres entité.

Boutons annuler

A l'image de la methode de modification cette methode appelle la methode LivreEncoursDeModification avec comme arguments false

```

    /// <summary>
    /// annule les modification ou ajout en cours
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    1 référence | Doryan, Il y a 12 heures | 1 auteur, 1 modification
    private void btnAnnulerChoixLivres_Click(object sender, EventArgs e)
    {
        LivreEnCoursDeModif(false);
    }

```

Bouton de validation

Cette methode requiert aussi plusieurs explication au vu des nombreux test qu'elle contient

```

    /// <summary>
    /// valide dans la bdd les changements en cours ( ajout / modification)
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    1 référence | Doryan, Il y a 12 heures | 1 auteur, 1 modification
    private void btnValiderChoixLivres_Click(object sender, EventArgs e)
    {
        bool checkValid;
        if (MessageBox.Show("Etes vous sur ?", "oui ?", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            string id = txblivresNumero.Text; ;
            int? a = null;
            try
            {
                a = int.Parse(txblivresNumero.Text);
            }
            catch
            {
                MessageBox.Show("Le Numéro de document doit etre un entier");
            }
            Genre unGenre = (Genre)cbxlivresGenreInfos.SelectedItem;
            Public unPublic = (Public)cbxlivresPublicInfos.SelectedItem;
            Rayon unRayon = (Rayon)cbxlivresRayonInfos.SelectedItem;
            if (unGenre == null)
                MessageBox.Show("Genre invalide");
            if (unPublic == null)
                MessageBox.Show("Public invalide");
            if (unRayon == null)
                MessageBox.Show("Rayon invalide");
            string titre = txblivresTitre.Text;
            string image = txblivresImage.Text;
            string isbn = txblivresIsbn.Text;
            string auteur = txblivresAuteur.Text;
            string collection = txblivresCollection.Text;
            string idGenre = (unGenre == null) ? null : unGenre.Id;
            string genre = (unGenre == null) ? null : unGenre.Libelle;
            string idPublic = (unPublic == null) ? null : unPublic.Id;
            string lePublic = (unPublic == null) ? null : unPublic.Libelle;
            string idRayon = (unRayon == null) ? null : unRayon.Id;
            string rayon = (unRayon == null) ? null : unRayon.Libelle;
            if (a != null && titre != "" && auteur != "" && genre != null && unPublic != null)
            {
                Livre livre = new Livre(id, titre, image, isbn, auteur, collection, idGenre, genre, idPublic, lePublic, idRayon, rayon);
                if (txblivresNumero.ReadOnly) // si on est en modification
                    checkValid = controller.ModifierLivre(livre);
                else // si on est en creation
                    checkValid = controller.CreerLivre(livre);
                if (checkValid)
                {
                    LivreEnCoursDeModif(false);
                    lesLivres = controller.GetAllLivres();
                    RemplirLivresListeComplete();
                }
                else
                {
                    if (txblivresNumero.ReadOnly)
                        MessageBox.Show("numéro de publication déjà existant", "Erreur");
                    else
                        MessageBox.Show("Erreur");
                }
            }
        }
    }

```

Tout d'abord nous creons une valeur booleen pour nos test dans la methode. Lors du click sur le boutons valider nous avons une message box qui nous demande de confirmer. Avant de créer ou modifier l'entité la methode va vérifier plusieurs choses : tout d'abord nous allons nous assurer que le numero du livre soit bien un entier. Nous allons donc déclarer une variable int

« a » avec une valeur null avec « int ? a = null » qui est l'écriture simplifiée de nullable <int> a dans un try...catch; puis attribuer a la variable a la conversion de txtLivresNumero.text qui était un chaîne de caractère. Si nous entrons autre chose qu'un entier nous aurons un message d'erreur.

Ensuite il faut contrôler que les champs genre, public et rayon sont remplis sans quoi nous aurons un message d'erreur. Puis on fait correspondre les paramètres de l'objet livre avec les champs informatifs. Puis nous contrôlons que certaines valeurs ne sont pas null. Si tel est le cas nous créons un nouvel objet de type livres et contrôlons si l'id est en lecture seule ou non puis nous appelons les méthodes du contrôleur CréerLivre ou ModifierLivre selon le statut de l'id. cependant si pendant la création d'un livre l'id est en lecture seule nous avons un message d'erreur nous informons que le numéro de publication est déjà existant. Et si les méthodes du contrôleur sont incorrectes nous aurons un message d'erreur.

Pour les dvd et les revues la méthode est similaire mais étant donné que les objets dvd et revue possèdent des attributs de type int nous allons devoir les tester à la manière de l'id

```
881
882 /// <summary>
883 /// Valide les modifications en cours dans la BDD (ajouter / supprimer)
884 /// </summary>
885 /// <param name="sender"></param>
886 /// <param name="e"></param>
887 //référence | Dorian, il y a 12 heures | 1 auteur, 1 modification
888 private void btnValiderChoixDvd_Click(object sender, EventArgs e)
889 {
890     bool Valide = true;
891     if (MessageBox.Show("Êtes vous sûr ?", "oui ?", MessageBoxButtons.YesNo) == DialogResult.Yes)
892     {
893         string id = txtDvdNumero.Text;
894         int? a = null; // version simplifiée de nullable <int> a
895         try
896         {
897             a = int.Parse(txtDvdNumero.Text);
898         }
899         catch
900         {
901             MessageBox.Show("Le Numéro de document doit être un entier");
902         }
903         Genre unGenre = (Genre)cbxDvdGenreInfos.SelectedItem;
904         Public unPublic = (Public)cbxDvdPublicInfos.SelectedItem;
905         Rayon unRayon = (Rayon)cbxDvdRayonInfos.SelectedItem;
906         if (unGenre == null)
907             MessageBox.Show("Genre invalide");
908         if (unPublic == null)
909             MessageBox.Show("Public invalide");
910         if (unRayon == null)
911             MessageBox.Show("Rayon invalide");
912         string titre = txtDvdTitre.Text;
913         string image = txtDvdImage.Text;
914         int duree = (txtDvdDuree.Text == "") ? 0 : int.Parse(txtDvdDuree.Text);
915         string realisateur = txtDvdRealisateur.Text;
916         string synopsis = txtDvdSynopsis.Text;
917         string idGenre = (unGenre == null) ? null : unGenre.Id;
918         string genre = (unGenre == null) ? null : unGenre.Libelle;
919         string idPublic = (unPublic == null) ? null : unPublic.Id;
920         string lePublic = (unPublic == null) ? null : unPublic.Libelle;
921         string idRayon = (unRayon == null) ? null : unRayon.Id;
922         string rayon = (unRayon == null) ? null : unRayon.Libelle;
923         if (a != null && titre != "" && realisateur != "" && genre != null && unPublic != null)
924         {
925             Dvd dvd = new Dvd(id, titre, image, duree, realisateur, synopsis, idGenre, genre, idPublic, lePublic, idRayon, rayon);
926             if (txtDvdNumero.ReadOnly) // si on est en modification
927                 Valide = controller.ModifierDvd(dvd);
928             else // si on est en création
929                 Valide = controller.CreerDvd(dvd);
930             if (Valide)
931             {
932                 DvdEnCoursDeModif(false);
933                 lesDvd = controller.GetAllDvd();
934                 RemplirDvdListeComplete();
935             }
936             else
937             {
938                 if (txtDvdNumero.ReadOnly)
939                     MessageBox.Show("numéro de publication déjà existant", "Erreur");
940                 else
941                     MessageBox.Show("Erreur");
942             }
943         }
944     }
945 }
946
947
```

Les contrôleurs

Ces méthodes sont très utiles à la vue car les boutons les appellent pour ajouter, modifier ou supprimer une entité. Ces méthodes s'appuient-elles même sur des classes dans les fichiers access.


```

/// </summary>
/// <returns>Liste d'objets Livre</returns>
4références | Doryan, il y a 31 jours | 1 auteur, 1 modification
public List<Livre> GetAllLivres()
{
    return access.GetAllLivres();
}

/// <summary>
/// creer un livre dans la BDD
/// </summary>
/// <param name="livre"></param>
/// <returns>true si opération valide</returns>
1 référence | Doryan, il y a 16 jours | 1 auteur, 2 modifications
public bool CreerLivre(Livre livre)
{
    return access.CreerEntite("livre", JsonConvert.SerializeObject(livre));
}

/// <summary>
/// modifie un livre dans la bddd
/// </summary>
/// <param name="livre"></param>
/// <returns>true si opération valide</returns>
1 référence | Doryan, il y a 16 jours | 1 auteur, 3 modifications
public bool ModifierLivre(Livre livre)
{
    return access.ModifierEntite("livre", livre.Id, JsonConvert.SerializeObject(livre));
}

/// <summary>
/// supprime un livre dans la bdd
/// </summary>
/// <param name="livre"></param>
/// <returns>true si opération valide</returns>
1 référence | Doryan, il y a 16 jours | 1 auteur, 3 modifications
public bool SupprimerLivre(Livre livre)
{
    return access.SupprimerEntite("livre", JsonConvert.SerializeObject(livre));
}
#endregion

```

Classe Access.cs

Tout d'abord nous devons rajouter deux constantes pour gérer les méthode http PUT et DELETE

```

/// <summary>
/// méthode HTTP pour select
/// </summary>
private const string GET = "GET";
/// <summary>
/// méthode HTTP pour insert
/// </summary>
private const string POST = "POST";
/// <summary>
/// méthode HTTP pour update
/// </summary>
private const string PUT = "PUT";
/// <summary>
/// Méthode HTTP pour delete
/// </summary>
private const string DELETE = "DELETE";

```

Les méthodes du Controller concernant les dvd et les revues sont similaires, elle permettent de convertir en format Json l'objet donné (« livre », dvd, ou revues)

Pour créer, modifier ou supprimer que ce soit un livre un dvd ou une revue les méthodes font appel aux méthodes CréerEntité, ModifierEntite ou SupprimerEntite ci-dessous

(ici nous avons pris la methode ModifierEntité)

```
/// <summary>
/// Modifie une entite dans la BDD, return true si l'opération, c'est correctement déroulé
/// </summary>
/// <param name="type"></param>
/// <param name="id"></param>
/// <param name="jsonEntite"></param>
/// <returns></returns>
Références | Doryan, Ilya 1 jour | 1 auteur, 4 modifications
public bool ModifierEntite(string type, string id, String jsonEntite)
{
    try
    {
        // récupération soit d'une liste vide (requête ok) soit de null (erreur)
        List<Object> liste = TraitementRecup<Object>(PUT, type + "/" + id + "/" + jsonEntite);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Log.Error(ex, "Access.UpdateEntite catch type erreur={0}, table={1}, champs={2}", ex, type, jsonEntite);
    }
    return false;
}
```

Cette methode attend un type (un livre un dvd, une revue) et une Entité Json). Nous recuperons une liste vide si la methode fonctionne et retourne null si ce n'est pas le cas. contrairement a aux methode de creation et de suppression. La methode de modification prend aussi comme parametre l'id de l'entite en question car une modification ne doit pas porter sur son id.

Les méthodes essayent par le biais d'un try catch de créer une liste object ou une liste vide a transmettre a l'api grâce a la methode TraitementRecup<T> qui prend en parametre une methode (get post put delete) et un message qui est les informations que les méthodes CréerEntite, ModifierEntite et supprimerEntite.

Ces Ajouts se font sur des entités composées un livre est aussi un documents mais également un livre_dvd. De même pour les dvd. Il nous faut donc gérer ces entités composées dans l'api pour que lors de l'ajout d'une entité les attributs des classes mère soient aussi ajoutés.

```
/**
 * Ajout de l'entité composée livre dans la bdd
 *
 * @param [type] $champs nom et valeur de chaque champs de la ligne
 * @return true si l'ajout a fonctionné
 */
public function insertLivre($champs)
{
    $champsDocument = [ "id" => $champs["Id"], "titre" => $champs["Titre"],
        "image" => $champs["Image"], "idRayon" => $champs["IdRayon"],
        "idPublic" => $champs["IdPublic"], "idGenre" => $champs["IdGenre"]];
    $champsLivreDvd = [ "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", $champsLivreDvd);
    if ($result == null || $result == false) {
        return null;
    }
    return $this->insertOne("livre", $champsLivre);
}
```

Cette methode provient de la classe AccessBdd.php de l'api et fait en sorte que lors d'un ajout d'un livre les champs de l'entité document et de l'entité livreDvd soient ajouté aussi. Il en va de même pour les dvd que ce soit dans l'ajout, la modification ou la suppression.

Lors de la creation d'une entité pour éviter qu'un id se repete nous creons une methode qui incremente l'id a partir de l'id max de la table concerné

```

    /**
     * Retourne la plus grande id de la table livre
     *
     * @return lignes de la requete
     */
    public function selectMaxLivre() {
        $req = "Select MAX(id) AS id FROM livre";
        return $this->conn->query($req);
    }

    /**
     * Retourne la plus grande id de la table dvd
     *
     * @return lignes de la requete
     */
    public function selectMaxDvd() {
        $req = "Select MAX(id) AS id FROM dvd";
        return $this->conn->query($req);
    }

    /**
     * Retourne la plus grande id de la table revue
     *
     * @return lignes de la requete
     */
    public function selectMaxRevue() {
        $req = "Select MAX(id) AS id FROM revue";
        return $this->conn->query($req);
    }

```

```

    }

    /// <summary>
    /// Augmente un index de type string
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>

    private string plusUnIdString(string id)
    {
        int taille = id.Length;
        int idnum = int.Parse(id) + 1;
        id = idnum.ToString();
        if (id.Length > taille)
            MessageBox.Show("Taille du registre arrivé a saturation");
        while (id.Length != taille)
        {
            id = "0" + id;
        }
        return id;
    }
}

```

Mission 2 gerer les commandes.

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).

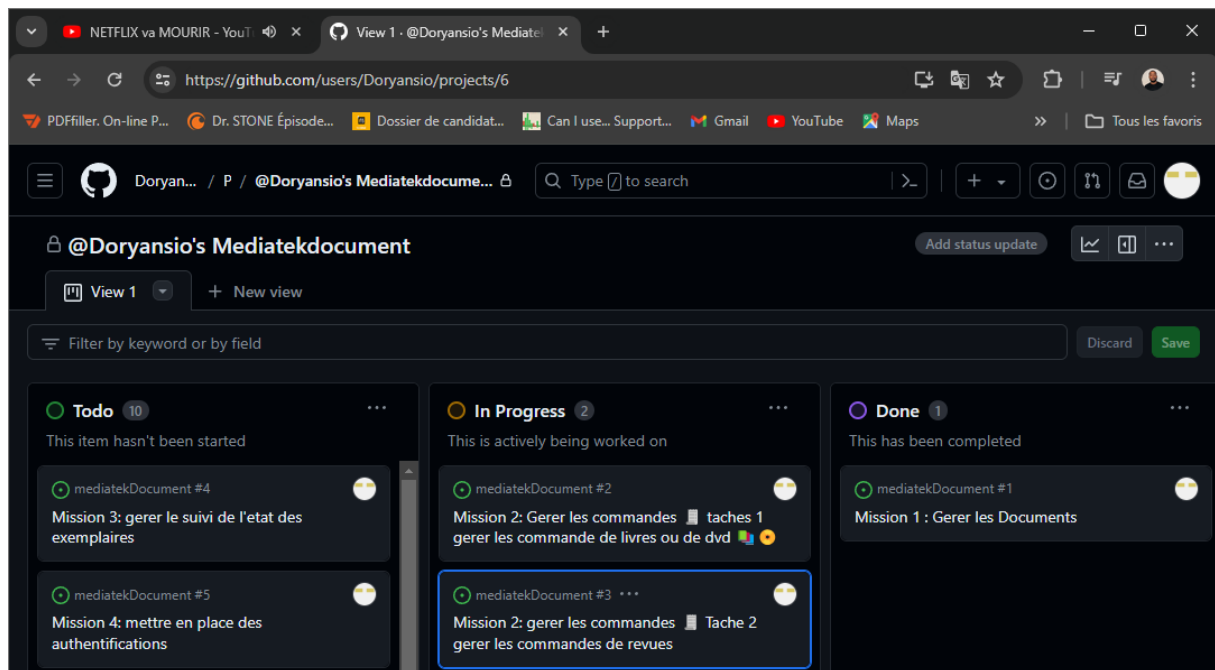
Permettre de supprimer une commande uniquement si elle n'est pas encore livrée. Faire un trigger qui réalise aussi la suppression dans la classe fille.

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.

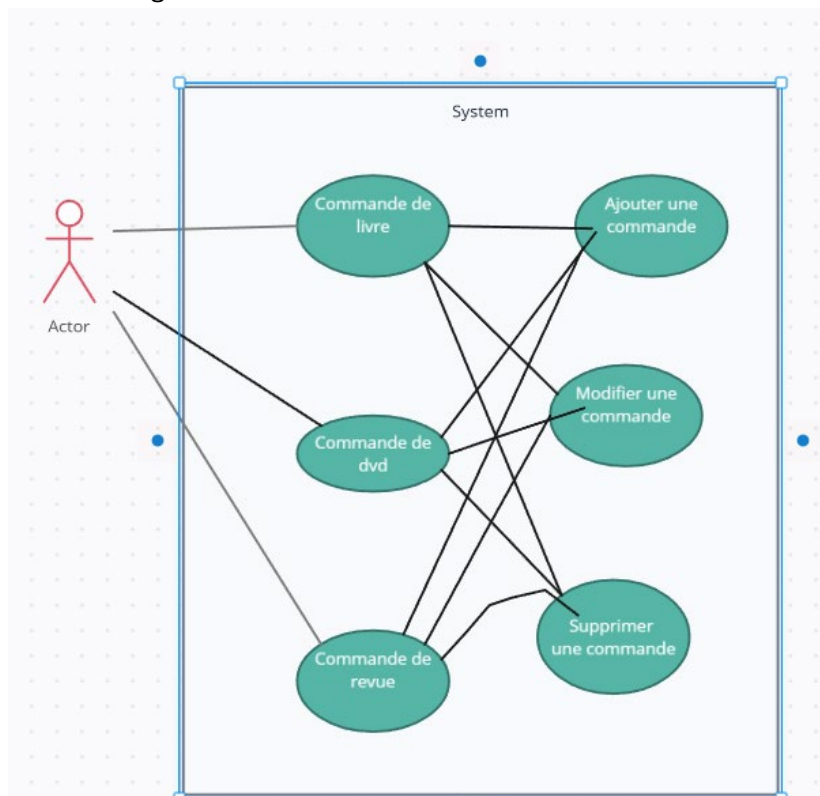
Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Commande.



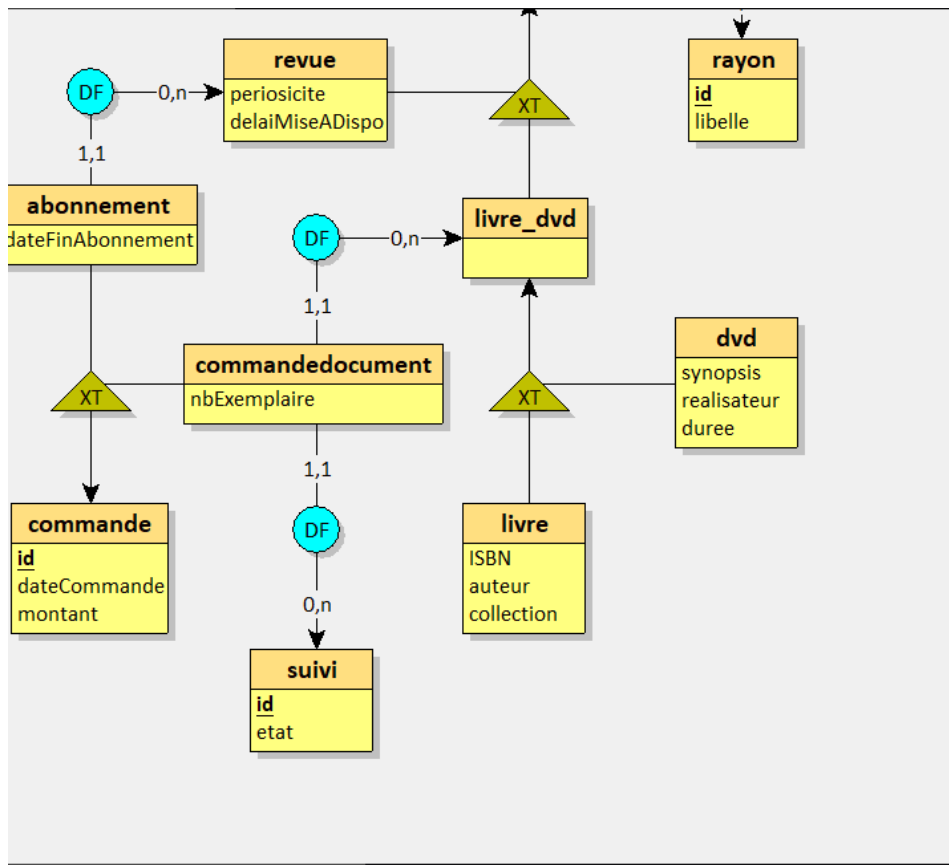
Temps estimé 12h

Temps de réalisations 10h

Voici le diagramme de cas d'utilisation de la mission



Et le MCD de la base de donnée incluant l'ajout de la table suivi



Création de la classe suivie qui permettra de connaître l'avancée d'une commande d'une entité.

Structure de table

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	varchar(5)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
2	nbExemplaire	int(11)			Oui	NULL			Modifier Supprimer Plus
3	idLivreDvd	varchar(10)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
4	idSuivi	int(4)			Non	Aucun(e)			Modifier Supprimer Plus

Affichage des lignes 0 - 3 (total de 4, traitement en 0,0016 seconde(s).)

```
SELECT * FROM `suivi`
```

Options supplémentaires

	id	etat
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	en cours
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	relancée
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	livrée
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	4	réglée

La table suivi est est lié a la table commande document par le biais de la clé étrangère idSuivi en reference a suivi.

Il nous faut également créer une classe metier relative a cette table dans l'application c#

```

MediaTekDocuments
MediaTekDocuments.model.Suivi

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace MediaTekDocuments.model
8 {
9     17 références | Doryan, Il y a 1 jour | 1 auteur, 3 modifications
10    public class Suivi
11    {
12        10 références | 1/1 ayant réussi | Doryan, il y a 30 jours | 1 auteur, 1 modification
13        public int Id { get; }
14        5 références | 1/1 ayant réussi | Doryan, il y a 30 jours | 1 auteur, 1 modification
15        public string Etat { get; }
16
17        1 référence | Doryan, il y a 30 jours | 1 auteur, 1 modification
18        public Suivi(int id, string etat)
19        {
20            this.Id = id;
21            this.Etat = etat;
22        }
23        /// <summary>
24        /// Permet de retourner le libellé en chaine de caractere
25        /// </summary>
26        /// <returns></returns>
27        0 références | Doryan, Il y a 1 jour | 1 auteur, 2 modifications
28        public override string ToString()
29        {
30            return this.Etat;
31        }
32    }
33 }

```

Pour que nous puissions accéder au libellé correspondant a l'état nous devons faire une surcharge de la methode toString afin de retourner l'état sous forme de chaine de caractères. Dans la classe access.cs nous utilisons une méthode get pour avoir tout les suivis de la base de données dans une liste d'énumérable nommée « les suivis »

```

1 référence | Doryan, Il y a 3 jours | 1 auteur, 1 modification
public List<Suivi> GetAllSuivis()
{
    IEnumerable<Suivi> LesSuivis = TraitementRecup<Suivi>(GET, "suivi");
    return new List<Suivi>(LesSuivis);
}

```

Voici la page de commande de livre qui est similaire a celle des dvd et des revues

Gestion des documents de la médiathèque

Livres DVD Revues Parutions des revues Commande de Livres Commande de Dvd Abonnements

Recherche Livres

Saisir une partie ou le titre d'un document : Ou Sélectionner le Genre : X

Saisir un numéro de document : Recherche Ou Sélectionner le Public : X

Ou Sélectionner le Rayon : X

Numero du document :

Code ISBN :

Titre :

Auteur(e) :

Collection :

Genre :

Public :

Rayon :

Chemin de l'image :

Commandes concernant le livre sélectionné

Numéro de la commande :

Date de la commande : mercredi 17 avril 2024

Montant :

Nombre d'emplaires :

Numéro du livre :

Etat de la commande :

Ajouter Modifier Supprimer

Valider Annuler

Cet onglet se divise en deux parties, celle du haut avec une datagrid view permettant de voir la liste des livres avec à sa gauche les informations de l'entité sélectionnée dans le datagrid. Lors d'une sélection d'une entité, si il y a des commandes concernant l'entité, elle s'affiche dans le data grid du dessous. Sur la sélection d'une commande, les informations relatives à cette dernière s'affichent dans les informations de commandes. Les dgv peuvent permettre de trier les entités.

Voici les méthodes pour trier les entités dans le dgv de la liste des livres


```

/// <summary>
/// Tri sur les colonnes
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | Doryan, il y a 21 jours | 1 auteur, 1 modification
private void DgvLesLivresListe_ColumnHeaderMouseClick(object sender, DataGridViewCellEventArgs e)
{
    VideComLivresZones();
    string titreColonne = DgvLesLivresListe.Columns[e.ColumnIndex].HeaderText;
    List<Livres> sortedList = new List<Livres>();
    switch (titreColonne)
    {
        case "Id":
            sortedList = lesComLivres.OrderBy(o => o.Id).ToList();
            break;
        case "Titre":
            sortedList = lesComLivres.OrderBy(o => o.Titre).ToList();
            break;
        case "Collection":
            sortedList = lesComLivres.OrderBy(o => o.Collection).ToList();
            break;
        case "Auteur":
            sortedList = lesComLivres.OrderBy(o => o.Auteur).ToList();
            break;
        case "Genre":
            sortedList = lesComLivres.OrderBy(o => o.Genre).ToList();
            break;
        case "Public":
            sortedList = lesComLivres.OrderBy(o => o.Public).ToList();
            break;
        case "Rayon":
            sortedList = lesComLivres.OrderBy(o => o.Rayon).ToList();
            break;
    }
    RemplirComLivresListe(sortedList);
}

```

Et la methode pour trier le dgv des commandes

```

/// <summary>
/// tri sur les colonnes
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence | Doryan, il y a 21 jours | 1 auteur, 2 modifications
private void dgvLivresCommande_ColumnHeaderMouseClick(object sender, DataGridViewCellEventArgs e)
{
    if (lesCommandes.Count > 0 && dgvLivresCommande != null)
    {
        VideLivresComInfos();
        string titreColonne = dgvLivresCommande.Columns[e.ColumnIndex].HeaderText;
        List<CommandeDocument> sortedList = new List<CommandeDocument>();
        switch (titreColonne)
        {
            case "IdLivres":
                sortedList = lesCommandes.OrderBy(o => o.Id).ToList();
                break;
            case "DateCommande":
                sortedList = lesCommandes.OrderBy(o => o.DateCommande).ToList();
                break;
            case "NbExemplaire":
                sortedList = lesCommandes.OrderBy(o => o.NbExemplaire).ToList();
                break;
            case "Etat":
                sortedList = lesCommandes.OrderBy(o => o.IdSuivi).ToList();
                break;
            case "Montant":
                sortedList = lesCommandes.OrderBy(o => o.Montant).ToList();
                break;
        }
        RemplirLivresComListeCommandes(sortedList);
    }
}
#endregion

```

Les ajouts, les modifications, et les suppressions fonctionnent de la même façon que pour la mission 1, certains boutons deviennent inaccessibles par l'utilisateur pour le pousser à remplir les champs nécessaires. Et avant la validation d'un ajout et/ou d'une modification certaines

vérifications sont faite. L'utilisateur ne peut pas valider une suppression ou une modification si il n'y a pas de commande sélectionné. Une commande qui a un IdSuivi superieur a 2 ne peut pas être supprimé

```

2293 private void btnLivresComModifier_Click(object sender, EventArgs e)
2294 {
2295     if (dgvLivresComande.CurrentRow != null && txtCommandeNumeroComande.Text != "")
2296     {
2297         List<Suivi> lesSuivi = controller.GetAllSuivis().FindAll(o => o.Id >= ((Suivi)cbxCommandeLivreEtat.SelectedItem).Id).ToList();
2298         if (lesSuivi.Count > 2)
2299             lesSuivi = lesSuivi.FindAll(o => o.Id < 4).ToList();
2300         CommandeLivreEtdComModif(true);
2301         RemplirComboSuivi(lesSuivi, bdxComLivreEtat, cbxCommandeLivreEtat);
2302         cbxCommandeLivreEtat.SelectedIndex = 0;
2303     }
2304     else
2305     {
2306         MessageBox.Show("Aucune commande sélectionné");
2307     }
2308 }
2309
2310 /// <summary>
2311 /// supprime une commande de livre
2312 /// </summary>
2313 /// <param name="sender"></param>
2314 /// <param name="e"></param>
2315 1 référence | Dorian, il y a 21 jours | 1 auteur, 3 modifications
private void btnLivresComSupprimer_Click(object sender, EventArgs e)
2316 {
2317     CommandeDocument commandeDocument = (CommandeDocument)bdgListeCommandeLivre[bdgListeCommandeLivre.Position];
2318     if (dgvLivresComande.CurrentRow != null && txtCommandeNumeroComande.Text != "")
2319     {
2320         if (commandeDocument.IdSuivi > 2)
2321             MessageBox.Show("Une commande livrée ou réglée ne peut être supprimée");
2322         else if (MessageBox.Show("Êtes vous sûr de vouloir supprimer la commande n°" + commandeDocument.Id +
2323             " concernant " + lesComLivres.Find(o => o.Id == commandeDocument.IdLivreDvd).Titre + " ?",
2324             "Validation suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
2325         {
2326             if (controller.SupprimerLivreDvdCom(commandeDocument))
2327             {
2328                 try
2329                 {
2330                     Livre livre = (Livre)bdgListeLivresListe.List[bdgListeLivresListe.Position];
2331                     AfficheLivresComandeInfos(livre);
2332                     txtCommandeNumeroComande.Text = livre.Id;
2333                 }
2334                 catch
2335                 {
2336                     VideLivresComZones();
2337                 }
2338             }
2339             else
2340             {
2341                 MessageBox.Show("Erreur");
2342             }
2343         }
2344     }
2345 }

```

De plus la validation d'une commande ne peut pas s'effectuer si certaines conditions ne sont pas remplies comme le montre la capture ci-dessous

```

2378 /// <summary>
2379 /// valide la modification ou l'ajout en cours
2380 /// </summary>
2381 /// <param name="sender"></param>
2382 /// <param name="e"></param>
2383 1 référence | Dorian, il y a 7 jours | 1 auteur, 4 modifications
private void btnLivresComValider_Click(object sender, EventArgs e)
2384 {
2385     if (MessageBox.Show("Êtes vous sûr ?", "oui ?", MessageBoxButtons.YesNo) == DialogResult.Yes)
2386     {
2387         string id = txtCommandeNumeroComande.Text;
2388         bool checkValid = false;
2389         DateTime dateComande = DtpComandeLivre.Value;
2390         float montant = -1;
2391         int nbExemplaire = -1;
2392         try
2393         {
2394             montant = float.Parse(txtCommandeLivreMontant.Text);
2395         }
2396         catch
2397         {
2398             MessageBox.Show("Le montant doit être un nombre à virgule");
2399         }
2400         try
2401         {
2402             nbExemplaire = int.Parse(txtCommandeExemplaireLivre.Text);
2403         }
2404         catch
2405         {
2406             MessageBox.Show("Le nombre d'exemplaire doit être un nombre à entier");
2407         }
2408         string idLivreDvd = txtCommandeNumeroLivre.Text;
2409         int idSuivi = 0;
2410         string etat = "";
2411         Suivi suivi = (Suivi)cbxCommandeLivreEtat.SelectedItem;
2412         if (suivi != null)
2413         {
2414             idSuivi = suivi.Id;
2415             etat = suivi.Etat;
2416         }
2417         else
2418             MessageBox.Show("Veuillez sélectionner un état");
2419         if (montant <= -1 && nbExemplaire != -1 && etat != "")
2420         {
2421             CommandeDocument commandeLivre = new CommandeDocument(id, dateComande, montant, nbExemplaire, idLivreDvd, idSuivi, etat);
2422             if (!ajouterBool)
2423                 checkValid = controller.ModifierLivreDvdCom(commandeLivre);
2424             else
2425                 checkValid = controller.CreerLivreDvdCom(commandeLivre);
2426             if (checkValid)
2427             {
2428                 // ...
2429             }
2430         }
2431     }
2432 }

```

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.

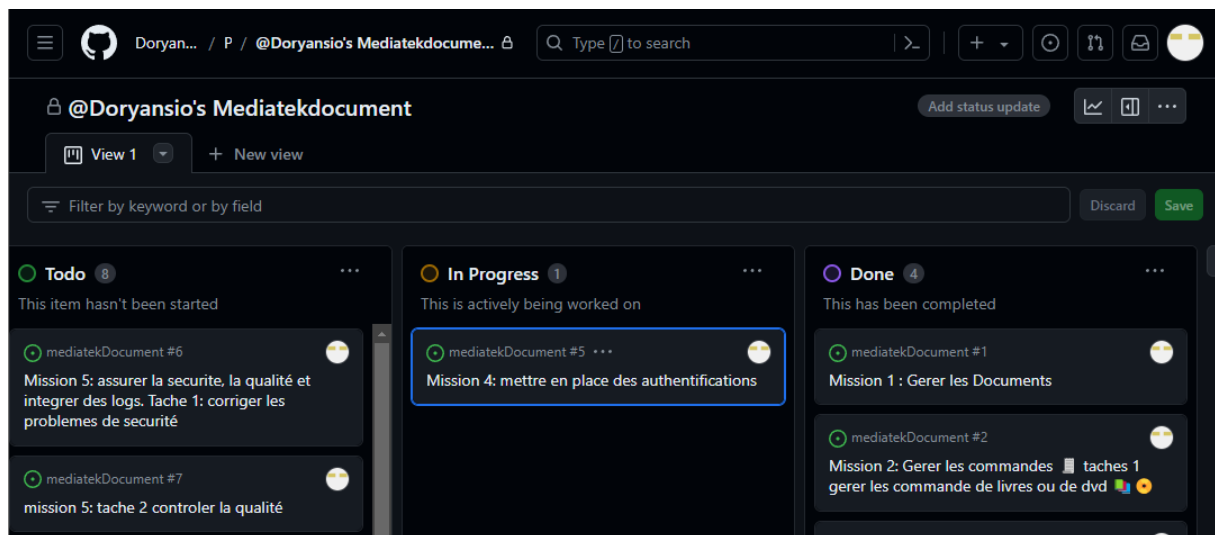
Suivant le type de personne authentifiée, empêcher certains accès en rendant invisibles ou inactifs certains onglets ou objets graphiques.

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.

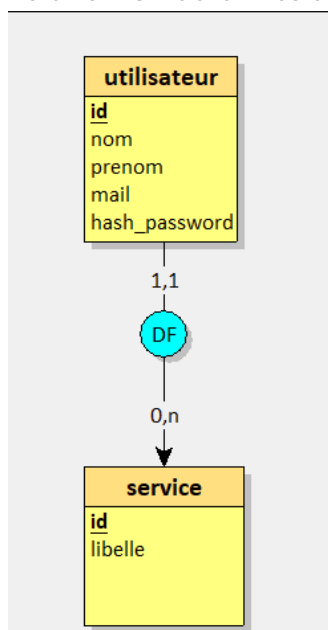
Faire en sorte que l'alerte de fin d'abonnement n'apparaisse que pour les personnes concernées (qui gèrent les commandes).

Temps de réalisation estimé : 4 heures

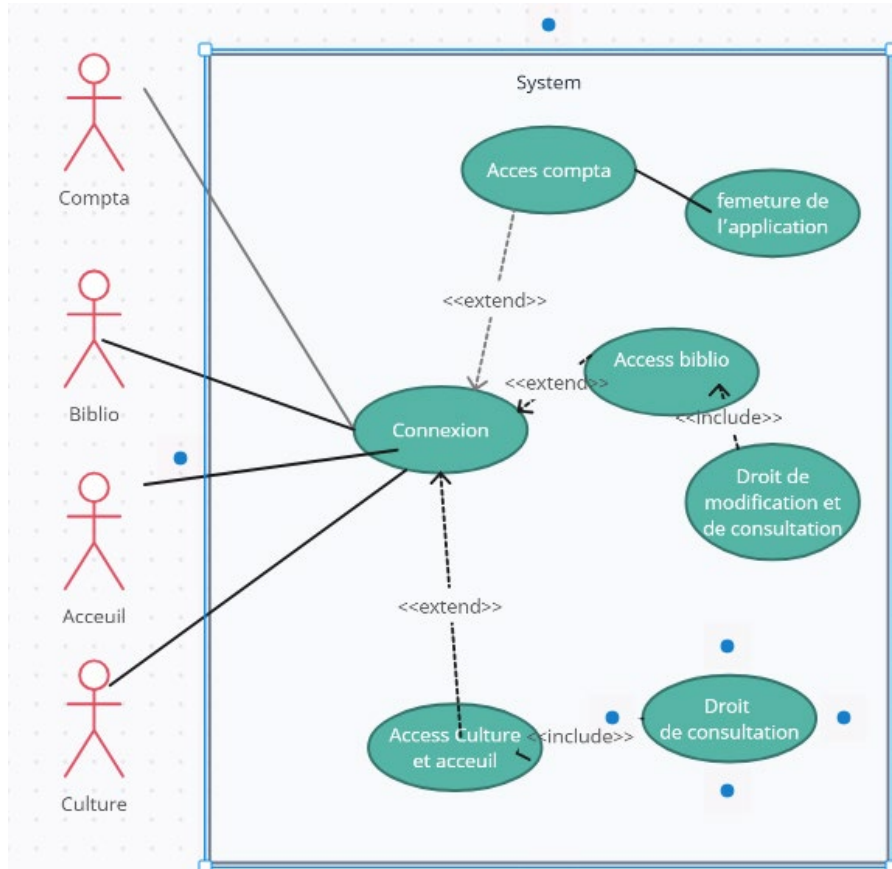
Temps de réalisation réel 3 heures



Voici le MCD de la mission concernée



Et le diagramme de cas d'utilisation :



Dans le cadre de cette mission nous devons mettre à jour la base de donnée pour y ajouter la table « service » et utilisateur en sachant qu'un utilisateur ne peut être rattaché qu'à un seul service

`SELECT * FROM `service``

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Options supplémentaires

	id	libelle
<input type="checkbox"/> Éditer Copier Supprimer	0001	compta
<input type="checkbox"/> Éditer Copier Supprimer	0002	biblio
<input type="checkbox"/> Éditer Copier Supprimer	0003	culture
<input type="checkbox"/> Éditer Copier Supprimer	0004	accueil

↑ ☐ Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

✓ Affichage des lignes 0 - 4 (total de 5, traitement en 0,0015 seconde(s).)

`SELECT * FROM `utilisateur``

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Options supplémentaires

	id	nom	prenom	mail	password	idservice
<input type="checkbox"/> Éditer Copier Supprimer	0001	test	test	test@mail.com	71ca866e741cf1502ff37be74eaa714f44699fae060f1cbe6...	0002
<input type="checkbox"/> Éditer Copier Supprimer	0002	test2	test2	test2@mail.com	71ca866e741cf1502ff37be74eaa714f44699fae060f1cbe6...	0001
<input type="checkbox"/> Éditer Copier Supprimer	0003	test3	test3	test3@mail.com	71ca866e741cf1502ff37be74eaa714f44699fae060f1cbe6...	0003
<input type="checkbox"/> Éditer Copier Supprimer	0004	test4	test4	test4@mail.com	71ca866e741cf1502ff37be74eaa714f44699fae060f1cbe6...	0004
<input type="checkbox"/> Éditer Copier Supprimer	0005	Doryan	Doryan	dodo@mail.com	ecd71870d1963316a97e3ac3408c9835ad8cf0f3c1bc703527...	0001

↑ ☐ Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

Nous devons aussi créer ces table dans l'application C#

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace MediaTekDocuments.model
8  {
9      3 références | Doryan, il y a 13 jours | 1 auteur, 1 modification
10     public class Service
11     {
12         2 références | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
13         public string Id { get; }
14         3 références | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
15         public string Libelle { get; }
16
17         1 référence | Doryan, il y a 13 jours | 1 auteur, 1 modification
18         public Service(string id, string libelle)
19         {
20             this.Id = id;
21             this.Libelle = libelle;
22         }
23
24         1 référence | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
25         public override string ToString()
26         {
27             return this.Libelle;
28         }
29     }
30 }

```

```

7  namespace MediaTekDocuments.model
8  {
9      14 références | Doryan, il y a 12 jours | 1 auteur, 2 modifications
10     public class Utilisateur
11     {
12
13         2 références | Doryan, il y a 12 jours | 1 auteur, 1 modification
14         public Utilisateur(string id, string nom, string prenom, string mail, string idService, string service)
15         {
16             this.Id = id;
17             this.Nom = nom;
18             this.Prenom = prenom;
19             this.Mail = mail;
20             this.IdService = idService;
21             this.Service = service;
22         }
23
24         2 références | 1/1 ayant réussi | Doryan, il y a 12 jours | 1 auteur, 2 modifications
25         public string Id { get; set; }
26         4 références | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
27         public string Nom { get; set; }
28         2 références | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
29         public string Mail { get; set; }
30
31         2 références | 1/1 ayant réussi | Doryan, il y a 12 jours | 1 auteur, 1 modification
32         public string Prenom { get; set; }
33         2 références | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
34         public string IdService { get; set; }
35         5 références | 1/1 ayant réussi | Doryan, il y a 13 jours | 1 auteur, 1 modification
36         public string Service { get; set; }
37     }
38 }

```

Encore un fois nous devons surcharger la methode toString afin de pouvoir récupérer le libellé du service concerné

Pour permettre la connection nous devons récupérer les utilisateur dans la classe access.cs

```

    ///<summary>
    ///Methode pour avoir un utilisateur
    ///<param name="mail"></param>
    ///<param name="hash"></param>
    /// </summary>
    1 référence | Doryan, il y a 5 jours | 2 auteurs, 4 modifications
    public Utilisateur GetLogin(string mail, string hash)
    {
        Dictionary<string, string> login = new Dictionary<string, string>
        {
            { "mail", mail },
            { "password", hash }
        };
        string mailHash = JsonConvert.SerializeObject(login);
        List<Utilisateur> utilisateurs = TraitementRecup<Utilisateur>(GET, "utilisateur/" + mailHash);
        Console.WriteLine(utilisateurs.Count);
        if (utilisateurs.Count > 0)
        {
            return utilisateurs[0];
        }
        Log.Error("Access.GetLogin catch user fail connection : {Mail}", mail);
        return null;
    }

```

Le login est convertit en un dictionnaire de cles/valeur au format Json afin de pouvoir envoyer cette information a l'API pour permettre ou non la connexion si les informations correspondent avec celle de la base de données (soit le mail et le mot de passe sur lequel nous avons appliquer un salt pour qu'il ne soit pas accessible en clair dans la base de données.

La methode salage se trouve dans la classe FrmLoginController que nous avons créer

```

MediaTekDocuments
MediaTekDocuments.controller.FrmLoginController
GetLogin(string mail, string password)
35
36
37
38
39
    /// <summary>
    /// Lance la vue principale
    /// </summary>
    1 référence | Doryan, il y a 7 jours | 1 auteur, 2 modifications
    private void Init()
    {
        FrmMediatek mediatek = new FrmMediatek(utilisateur);
        mediatek.Show();
    }
45
    1 référence | Doryan, il y a 21 heures | 1 auteur, 5 modifications
    public bool GetLogin(string mail, string password)
    {
        password = "Mediatek" + password;
        string hash = "";
        using (SHA256 sha256Hash = SHA256.Create())
        {
            hash = GetHash(sha256Hash, password);
        }
        utilisateur = access.GetLogin(mail, hash);
        if (utilisateur != null)
        {
            Init();
            return true;
        }
        return false;
    }
63
    1 référence | Doryan, il y a 13 jours | 1 auteur, 1 modification
    private static string GetHash(HashAlgorithm hashAlgorithm, string input)
    {
        byte[] data = hashAlgorithm.ComputeHash(Encoding.UTF8.GetBytes(input));
        var sBuilder = new StringBuilder();
        for (int i = 0; i < data.Length; i++)
        {
            sBuilder.Append(data[i].ToString("x2"));
        }
        return sBuilder.ToString();
    }
77
78

```

Nous devons aussi faire en sorte qu'a l'execution du programme nous démarrions sur la page d'authentification pour que l'application ne se lance qu'après qu'un utilisateur ait renseigné ses

informations. Donc dans la vue on crée une classe FrmLogin.cs contenant les méthodes d'initialisation et de vérification de la saisie des informations de l'utilisateur

```
1 using System;
2 using MediatekDocuments.controller;
3 using System.Windows.Forms;
4
5 namespace MediatekDocuments.view
6 {
7     3 références | Doryan, il y a 13 jours | 1 auteur, 1 modification
8     public partial class FrmLogin : Form
9     {
10         private FrmLoginController controller;
11         1 référence | Doryan, il y a 13 jours | 1 auteur, 1 modification
12         public FrmLogin()
13         {
14             InitializeComponent();
15             Init();
16         }
17
18         /// <summary>
19         /// Initialisations :
20         /// Création du contrôleur
21         /// </summary>
22         1 référence | Doryan, il y a 13 jours | 1 auteur, 1 modification
23         private void Init()
24         {
25             txtLogin.Text = "";
26             txtPwd.Text = "";
27             controller = new FrmLoginController();
28         }
29
30         /// <summary>
31         /// Demande de connexion
32         /// </summary>
33         /// <param name="sender"></param>
34         /// <param name="e"></param>
35         1 référence | Doryan, il y a 13 jours | 1 auteur, 1 modification
36         private void BtnConnec_Click(object sender, EventArgs e)
37         {
38             if (controller.GetLogin(txtLogin.Text, txtPwd.Text))
39                 this.Visible = false;
40             else
41             {
42                 MessageBox.Show("Mauvais mot de passe ou login utilisateur");
43                 txtLogin.Text = "";
44                 txtPwd.Text = "";
45             }
46         }
47     }
48 }
```

Puis nous allons appeler cette classe dans la vue de FrmMediatek afin que l'application se lance directement sur la fenêtre d'authentification

```
18 namespace MediatekDocuments.view
19 {
20     /// <summary>
21     /// Classe d'affichage
22     /// </summary>
23     7 références | Doryan, il y a 4 jours | 2 auteurs, 13 modifications
24     public partial class FrmMediatek : Form
25     {
26         #region Commun
27         private readonly FrmMediatekController controller;
28         private readonly BindingSource bdgGenres = new BindingSource();
29         private readonly BindingSource bdgPublics = new BindingSource();
30         private readonly BindingSource bdgRayons = new BindingSource();
31         private readonly Utilisateur utilisateur;
32         private bool ajouterBool = false;
33         private bool premierLoad = false;
34
35         /// <summary>
36         /// Constructeur : création du contrôleur lié à ce formulaire
37         /// </summary>
38         2 références | Doryan, il y a 4 jours | 2 auteurs, 3 modifications
39         public FrmMediatek(Utilisateur utilisateur)
40         {
41             InitializeComponent();
42             this.controller = new FrmMediatekController();
43             this.utilisateur = utilisateur;
44             VerifDroitAccueil(utilisateur);
45         }
46     }
47 }
```

Cette méthode fait appel à la méthode VerifDroitAccueil qui permet de prendre en compte le service de l'utilisateur afin de lui accorder différents droits sur l'application


```

72     }
73
74     /// <summary>
75     /// Retourne vrai ou faux si le service de l'utilisateur
76     /// est autorisé
77     /// </summary>
78     /// <param name="utilisateur"></param>
79     /// <returns></returns>
80     1 référence | PC, il y a 11 jours | 1 auteur, 1 modification
81     public bool VerifDroitAccueil(Utilisateur utilisateur)
82     {
83         Console.WriteLine(utilisateur.Nom);
84         List<string> services = new List<string> { "compta", "biblio", "accueil" };
85         if (services.Contains(utilisateur.Service))
86             return true;
87         return false;
88     }
89
90     /// <summary>
91     /// Retourne vrai ou faux si le service de l'utilisateur
92     /// est autorisé
93     /// </summary>
94     /// <param name="utilisateur"></param>
95     /// <returns></returns>
96     3 références | PC, il y a 11 jours | 1 auteur, 1 modification
97     public bool VerifDroitModif(Utilisateur utilisateur)
98     {
99         Console.WriteLine(utilisateur.Nom);
100         List<string> services = new List<string> { "biblio", "accueil" };
101         if (services.Contains(utilisateur.Service))
102             return true;
103         return false;
104     }
105
106     /// <summary>
107     /// Retourne vrai ou faux si le service de l'utilisateur
108     /// est autorisé
109     /// </summary>
110     /// <param name="utilisateur"></param>
111     /// <returns></returns>
112     4 références | PC, il y a 11 jours | 1 auteur, 1 modification
113     public bool VerifCommande(Utilisateur utilisateur)
114     {
115         List<string> services = new List<string> { "biblio" };
116         if (services.Contains(utilisateur.Service))
117             return true;
118         return false;
119     }

```

les méthodes accordant les droit sont des booleens qui renvoient vrai si le service de l'utilisateur est autorisé et faux si il ne l'est pas dans le cas du service compta qui n'as access a rien l'application se ferme a la connexion grâce a cette methode

```

117     }
118
119     /// <summary>
120     /// Verifie les droit d'un utilisateur
121     /// </summary>
122     /// <param name="lutilisateur"></param>
123     1 référence | PC, il y a 11 jours | 1 auteur, 1 modification
124     private void VerifDroitAccueil(Utilisateur lutilisateur)
125     {
126         if (!controller.VerifDroitAccueil(lutilisateur))
127         {
128             MessageBox.Show("Droit insuffisant");
129             Application.Exit();
130         }
131     }

```

L'accès a l'onglet de commande que ce soit des livres des dvd et des abonnements n'est accessible que par certains services donc a l'ouverture d'un onglet concernant les commandes on fait appel a la méthode booléenne de contrôle correspondant

```

/// <summary>
/// Ouverture de l'onglet Commande de Livres :
/// appel des méthodes pour remplir le datagrid des livres et des combos (genre, rayon, public)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void TabCommandeLivres_Enter(object sender, EventArgs e)
{
    if (!controller.VerifCommande(utilisateur))
    {
        MessageBox.Show("Droits insuffisant pour acceder a cette fonctionnaité");
        tabControl.SelectedIndex = 0;
    }
    else
    {
        lesComLivres = controller.GetAllLivres();
        RemplirComboCategorie(controller.GetAllGenres(), bdgGenres, cbxComLivresGenres);
        RemplirComboCategorie(controller.GetAllPublics(), bdgPublics, cbxComLivresPublics);
        RemplirComboCategorie(controller.GetAllRayons(), bdgRayons, cbxComLivresRayons);
        RemplirComboCategorie(controller.GetAllGenres(), bdgComLivreGenresInfo, cbxComLivresGenreInfos);
        RemplirComboCategorie(controller.GetAllPublics(), bdgComLivrePublicInfo, cbxComLivresPublicInfos);
        RemplirComboCategorie(controller.GetAllRayons(), bdgComLivreRayonInfo, cbxComLivresRayonInfos);
        RemplirComboSuivi(controller.GetAllSuivis(), bdgComLivreEtat, cbxCommandeLivreEtat);
        CommandeLivreEnCoursDeModif(false);
        RemplirComLivresListeComplete();
    }
}

```

Il en va de même pour les alertes de fin d'abonnement qui s'affiche que si l'utilisateur connecté est concerné par cette alerte.

```

89 /// <summary>
90 /// Ouvre une MessageBox au lancement de FrmMediatek.cs
91 /// si des abonnements sont proches de se terminer
92 /// </summary>
93 1 référence | Doryan, il ya 7 jours | 2 auteurs, 2 modifications
private void AfficherAlerteAbo()
{
    if (controller.VerifCommande(utilisateur))
    {
        bool interrupteur = false;
        List<Revue> revues = controller.GetAllRevues();
        string alerteRevues = "Revues dont l'abonnement se termine dans moins de 30 jours : \n";
        foreach (Revue revue in revues)
        {
            List<Abonnement> abonnements = controller.GetAbonnements(revue.Id);
            abonnements = abonnements.FindAll(o => (o.DateFinAbonnement <= DateTime.Now.AddMonths(1))
            && (o.DateFinAbonnement >= DateTime.Now));
            if (abonnements.Count > 0)
            {
                alerteRevues = string.Concat(alerteRevues, " - " + revue.Titre + "\n");
                interrupteur = true;
            }
        }
    }
}

```

Mission 5 : Assurer la sécurité, la qualité et intégrer des logs

mémoriser le couple "login:pwd" dans App.config au lieu de le laisser en dur directement dans Access.php.

Modifier le fichier htaccess pour prendre en compte une route vide et en tenir compte dans le fichier mediatekdocuments.php

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.

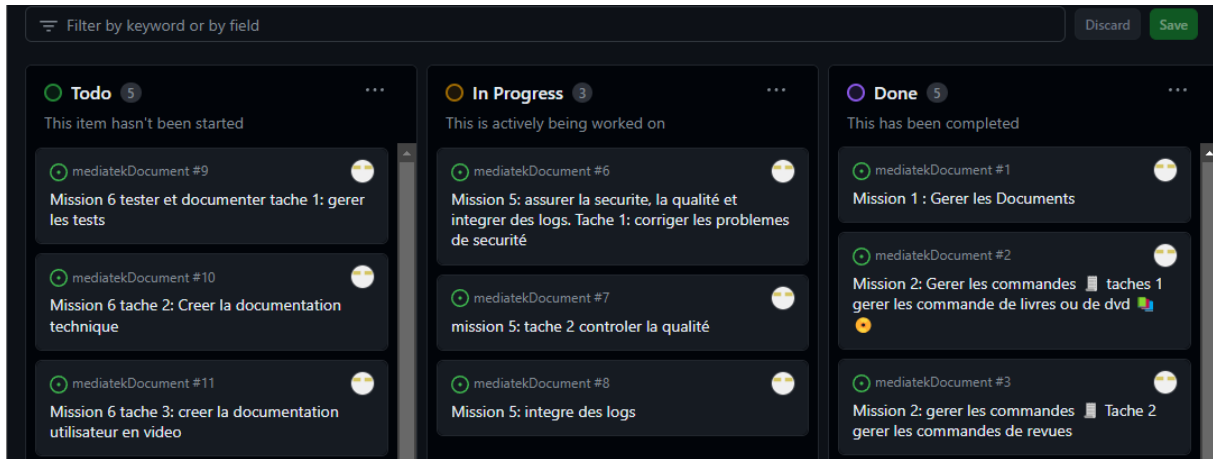
Corriger les problèmes relevés par Sonarlint dans le code ajouté (excepté les problèmes qui ne doivent pas être corrigés, comme les noms des méthodes événementielles qui commencent par une minuscule).

Contrôler les messages dans SonarQube.

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).

Temps estimé 8 heures

Temps de réalisations 8h



Par soucis de sécurité il est préférable de ne pas mettre les informations de connections dans la classe access.cs mais nous pouvons utiliser une un chaine de caractere en xml dans le fichier app.config de l'application

```
<connectionStrings>
<add name="MediatekDocuments.Properties.Settings.mediatekAuthenticationString"
connectionString="admin:adminpwd" />
</connectionStrings>
```

De cette manière nous pouvons directement appeler la connections string dans le fichier Access.cs

```
///<summary>
///Mot de passe et Login de L'api
///</summary>
private static readonly string authenticationName = "MediatekDocuments.Properties.Settings.mediatekAuthenticationString";
///</summary>
```

Pour l'api rest_mediatekdocument lorsque nous testons l'url sans preciser le champs auquel nous souhaitons accéder nous pouvons avoir acces a la structure de l'api ce qui n'est pas sécurisé non plus.

Pour remerdier a ce probleme nouspouvons modifier les règles de rewriting pour que lorsqu'un utilisateur tape http://localhost/rest_mediatekdocument il soit rediriger vers une page d'erreur 404.

```
1 RewriteEngine on
2 RewriteRule ^$ mediatekdocuments.php?error=404
```

```

1 RewriteEngine on
2 RewriteRule ^$ mediatekdocuments.php?error=404

6 // Contrôle de l'authentification
7 if(!isset($_SERVER['PHP_AUTH_USER']) || (isset($_SERVER['PHP_AUTH_USER']) &&
8     !($_SERVER['PHP_AUTH_USER']=='admin' && ($_SERVER['PHP_AUTH_PW']=='adminpwd')))){
9     $controle->unauthorized();
10
11 }else{
12     if(isset($_GET['error']) && $_GET['error'] == 404){
13         echo json_encode(array("message" => "404 Not Found "));
14         exit();
15     }
16

```

De cette manière nous spécifions que vouloir accéder a ce chemin particulier renvoie une erreur.

Intégration des logs

Les logs se place dans la classe Access.Cs de l'application en c# un utilisant le packages Serilog. Voici le code de configuration des logs

The screenshot shows a Trello board titled "@Doryansio's Mediatekdocument". The board is organized into three columns: "Todo" (2 items), "In Progress" (3 items), and "Done" (8 items). Each item is a card representing a task or mission.

Column	Item	Description
Todo (2)	mediatekDocument #12	Mission 7 deployer et gerer les sauvegardes de données tache 1: mettre en ligne l'api
	mediatekDocument #13	Mission 7 tache 2 gerer les sauvegardes des données
In Progress (3)	mediatekDocument #11	Mission 6 tache 3: creer la documentation utilisateur en video
	mediatekDocument #10	Mission 6 tache 2: Creer la documentation technique
	mediatekDocument #9	Mission 6 tester et documenter tache 1: gerer les tests
Done (8)	mediatekDocument #4	Mission 3: gerer le suivi de l'etat des exemplaires
	mediatekDocument #5	Mission 4: mettre en place des authentifications
	mediatekDocument #6	Mission 5: assurer la securite, la qualité et integrer des logs. Tache 1: corriger les problemes de sécurité

```

/// <summary>
/// Méthode privée pour créer un singleton
/// initialise l'accès à l'API
/// </summary>
1 référence | Doryan, il y a 5 jours | 2 auteurs, 4 modifications
private Access()
{
    try
    {
        Log.Logger = new LoggerConfiguration()
            .MinimumLevel.Verbose()
            .WriteTo.Console()
            .WriteTo.File("logs/log.txt")
            .CreateLogger();

        String authenticationString = GetAuthenticationString(authenticationName);
        String uriApi = GetAuthenticationString(uriApiName);
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Log.Fatal("Access catch error = {0}", e.Message);
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}

```

Et un exemple d'ajout d'un log dans un methode déjà écrite.

```

///<summary>
///Methode pour avoir un utilisateur
///<param name="mail"></param>
///<param name="hash"></param>
/// </summary>
1 référence | Doryan, il y a 5 jours | 2 auteurs, 4 modifications
public Utilisateur GetLogin(string mail, string hash)
{
    Dictionary<string, string> login = new Dictionary<string, string>
    {
        { "mail", mail },
        { "password", hash }
    };
    string mailHash = JsonConvert.SerializeObject(login);
    List<Utilisateur> utilisateurs = TraitementRecup<Utilisateur>(GET, "utilisateur/" + mailHash);
    Console.WriteLine(utilisateurs.Count);
    if (utilisateurs.Count > 0)
    {
        return utilisateurs[0];
    }
    Log.Error("Access.GetLogin catch user fail connection : {Mail}", mail);
    return null;
}

```

Mission 6 : gérer les tests

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

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.

Voici un exemple de test unitaire, concernant la classe abonnement

```

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using MediaTekDocuments.model;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace MediaTekDocuments.model.Tests
10 {
11     [TestClass]
12     public class AbonnementTests
13     {
14         private const string id = "0001";
15         private static readonly DateTime dateCommande = DateTime.Now;
16         private const float montant = 25.3F;
17         private static readonly DateTime dateFinAbonnement = DateTime.Now.AddMonths(2);
18         private const string idRevue = "0002";
19         private static readonly Abonnement abonnement = new Abonnement(id, dateCommande, montant, dateFinAbonnement, idRevue);
20
21         [TestMethod]
22         public void AbonnementTest()
23         {
24             Assert.AreEqual(id, abonnement.Id, "devrait reussir");
25             Assert.AreEqual(dateCommande, abonnement.DateCommande, "devrait reussir");
26             Assert.AreEqual(montant, abonnement.Montant, "devrait reussir");
27             Assert.AreEqual(dateFinAbonnement, abonnement.DateFinAbonnement, "devrait reussir");
28             Assert.AreEqual(idRevue, abonnement.IdRevue, "devrait reussir");
29         }
30     }
31 }
32

```

Et le test fonctionnel sous SpecFlow

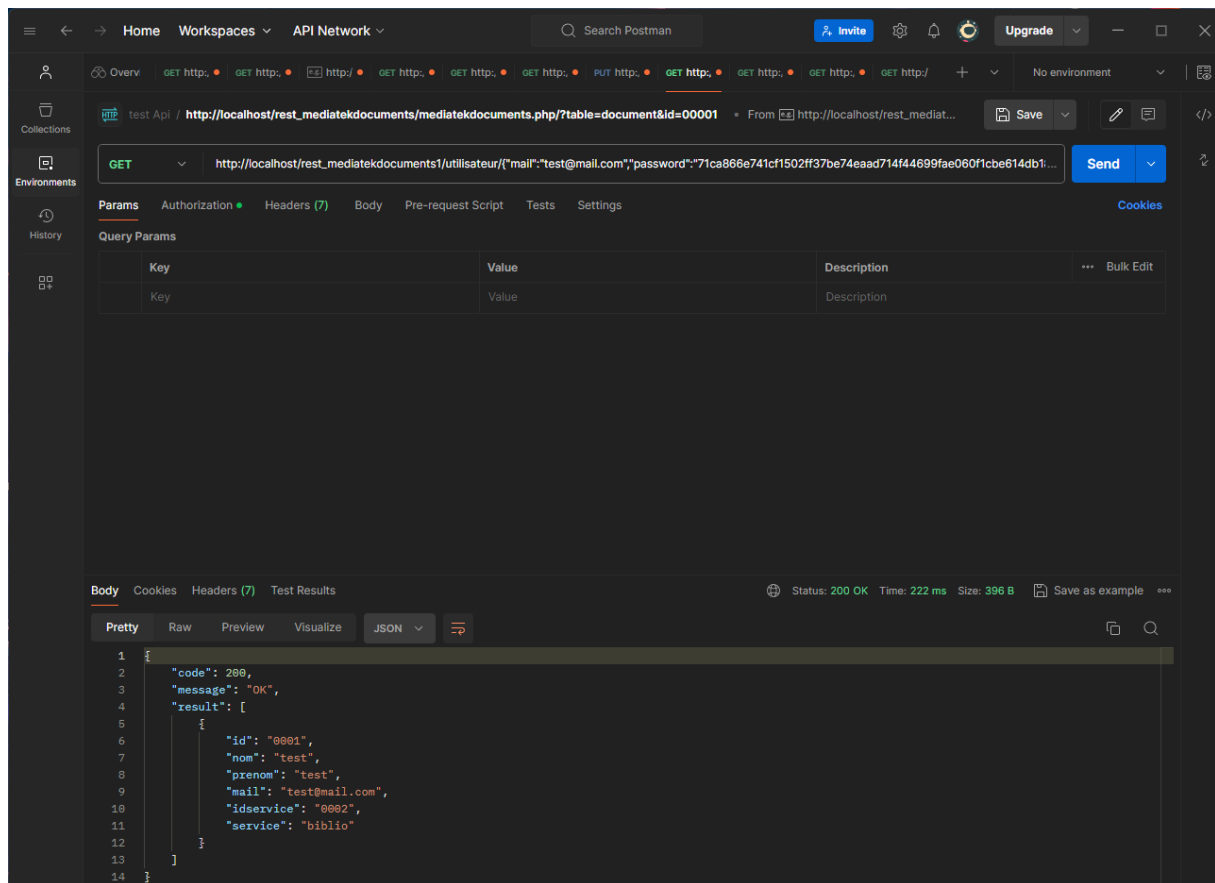
```

1  Feature: TestLivres
2
3
4
5  Scenario: Test Recherche par ID
6      Given Je saisis la valeur "00003" dans le champ de recherche de l'ID
7      When je clique sur le bouton de recherche
8      Then le datagridview affiche le livre possédant l'id "00003"

```

Le resultat des test :
(Le test fonctionnel ne fonctionne pas)

Et voici un test postman vérifiant que l'api fait bien ce qu'on lui demande.



Création de la documentation technique

Voici un exemple de commentaire normalisé qui sera utile pour la documentation technique dans l'application c# et dans l'api Rest

```

/// <summary>
/// Permet de gérer les demandes de requêtes post update delete concernant
/// une commande de livre ou dvd
/// </summary>
/// <param name="id"></param>
/// <param name="nbExemplaire"></param>
/// <param name="idLivreDvd"></param>
/// <param name="idSuivi"></param>
/// <param name="verbose"></param>
/// <returns></returns>
3 références | PC, il y a 11 jours | 1 auteur, 1 modification

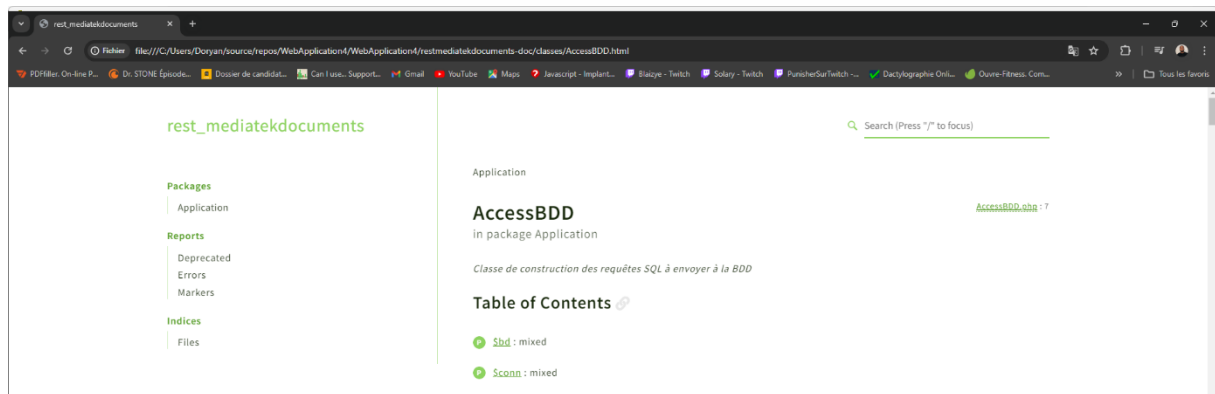
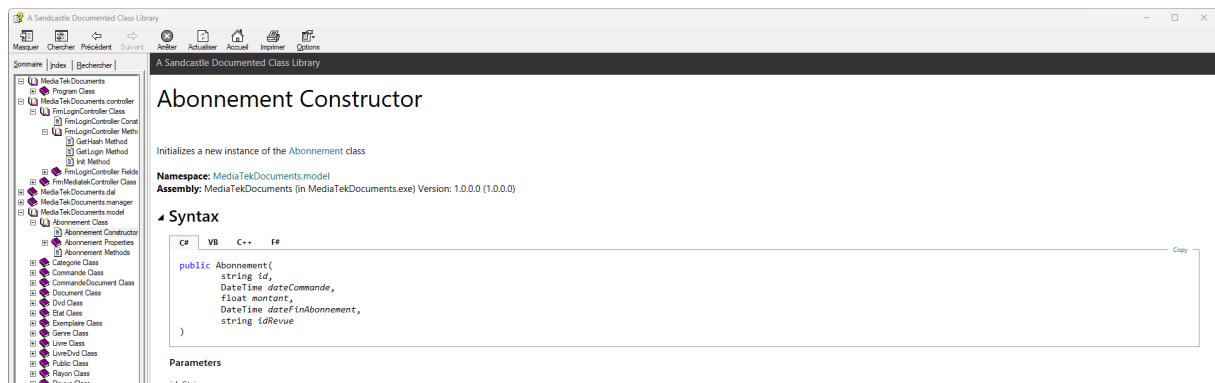
```

```

/**
 * récupération de toutes les lignes d'une table
 * @param string $table nom de la table
 * @return lignes de la requete
 */

```

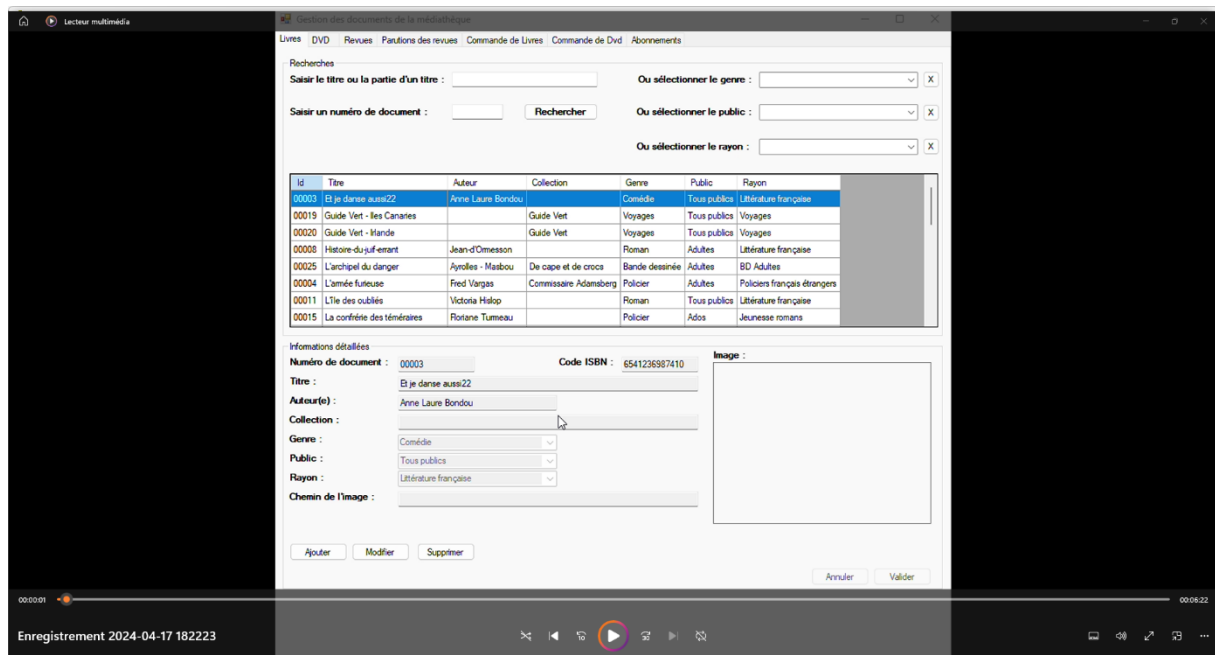
Et voici un extrait des documentation technique.



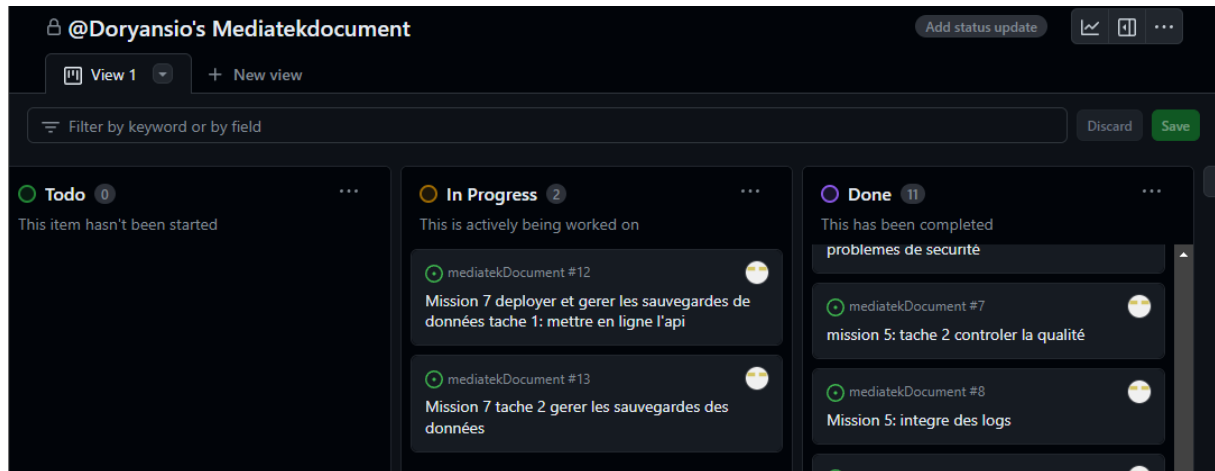
Documentation utilisateur en video

Outils utilisé : capture d'écran

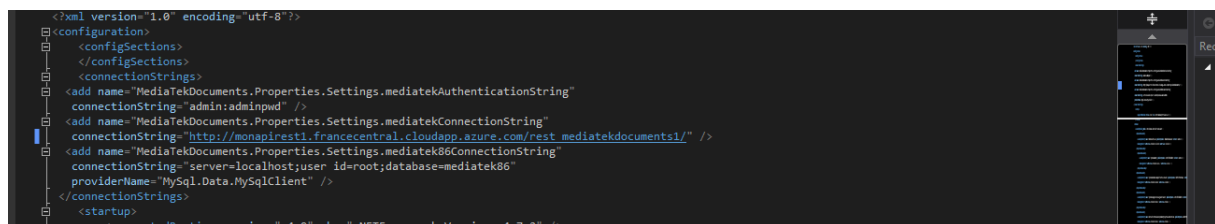
Temps de la video 6min24



Déployer et gérer les sauvegardes de données



Une fois l'Api mise en ligne il faut renseigner l'url dans le fichier app config de l'application en C#



Ensuite il faut modifier les règles de rewriting dans l'api en ligne







Afin de mettre l'api en ligne nous avons eu recours a une machine virtuelle azure ou nous avons installer wampServer afin de mettre la base de données sur php admin

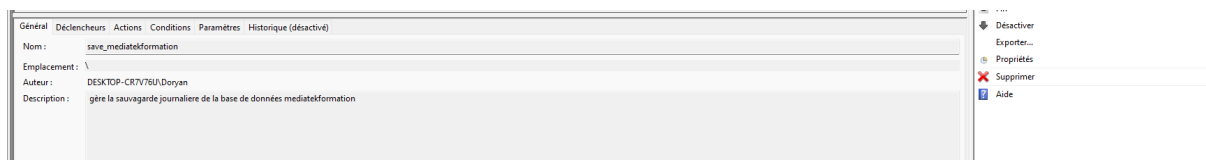
Pour sauvegarder la base de données de manière journaliere nous creons un script .bat que nous pouvons exécuter manuellement ou bien grâce au planificateur de tache de windows.

Voici le script et le fichier de sauvegarde.

```
set D=%date%
set DA=%D: /- %-
mysqldump -u root -p --databases mediatek --single-transaction > "C:\savebdd\bddbbackup_%DA%.sql"
```

Nom	Modifié le	Type	Taille
 bddbbackup_16-04-2024	16/04/2024 15:54	Fichier SQL	31 Ko
 bddbbackup_17-04-2024	17/04/2024 15:57	Fichier SQL	31 Ko
 Nouveau Document texte	16/04/2024 15:32	Document texte	0 Ko
 save	16/04/2024 15:54	Fichier de comma...	1 Ko

Dans le planificateur de tache :



Pour restaurer la base de données en cas de suppression ou de perte les fichiers de sauvegardes seront utile pour restaurer la base de données a la dernières sauvegardes. Il suffira juste d'importer ce fichier qui est un script sql dans phpMyAdmin.

Conclusion

A travers cette application nous avons pu mettre en pratique plusieurs compétences, comme la gestion des données, la conceptions d'applications et la conceptions d'une base de données et l'exploitations de donn   a l'aide d'un langage de requ  tes.

Recueillir, analyser et mettre    jour les informations sur une version d'une solution applicative

Probl  mes rencontr  s

L'applications comporte quelques bugs, notamment l'ajout d'un livre qui renvoie une erreur d'acc  s a l'api mais l'ajout est int  gr   a la base de donn     la r  ouverture de l'application.

L'ajout et la modification de commande ne s'actualisent pas dans la base de donn  es.