

Compte rendu d'activités

Réalisé lors de l'atelier professionnel n°3 au CNED

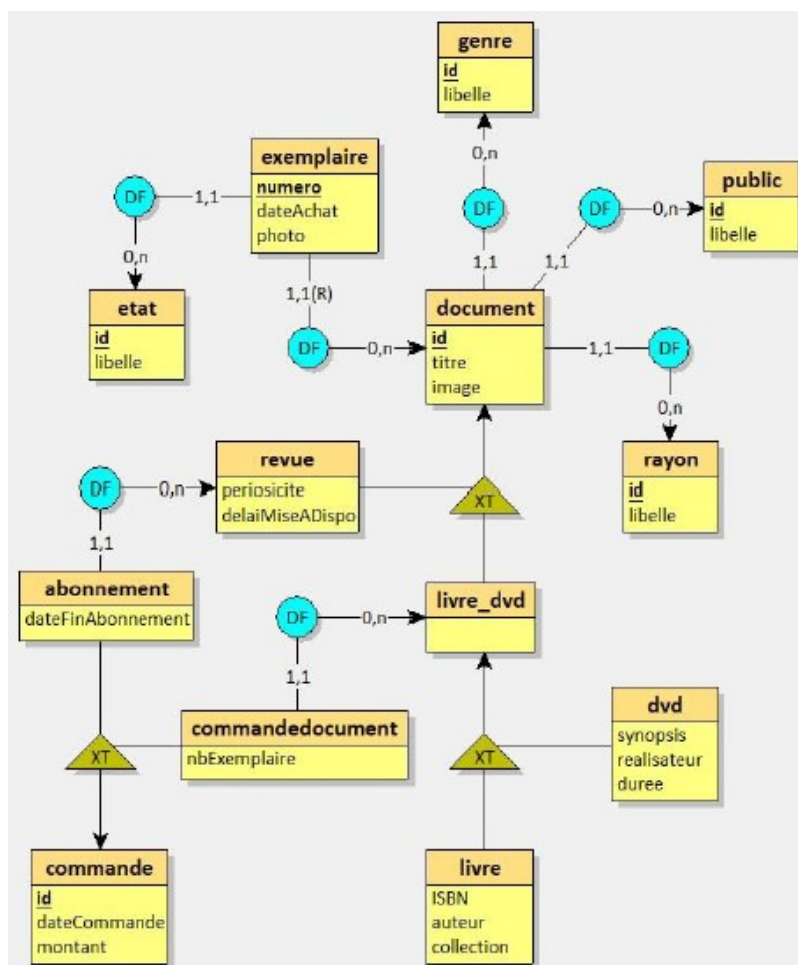
Présentation du projet :

MediaTek86 constitue un réseau responsable de la gestion des médiathèques dans le département de la Vienne. Sa principale mission est de coordonner les emprunts de livres, de DVD et de CD entre les différentes médiathèques tout en promouvant le développement de ressources médiathèques numériques pour l'ensemble du département.

Environnement de travail :

Dans le cadre de ce projet j'ai travaillé sous un OS Windows avec les IDE Visual Studio pour la partie C# et Apache NetBeans pour l'API

Mission 0 : préparer l'environnements de travail :



Importation de la bdd avec wamp et phpmyadmin ; installations des nugets et extensions nécessaire à l'aplication ; création des dépôt distant avec mise en place du kanban.



Mission 1 : gérer les documents :

Créations des boutons :

A screenshot of a web application window titled "Gestion des documents de la médiathèque". The window has a menu bar with "Livre", "DVD", "Revue", and "Autres documents". The main content area is divided into two sections. The top section is for "Recherches" (Search) and contains four input fields: "Saisir le titre ou la partie d'un titre :", "Saisir un numéro de document :", "Ou sélectionner le genre :", and "Ou sélectionner le public :". There are also buttons for "Rechercher" and "Annuler". The bottom section is for "Informations détaillées" (Detailed information) and contains several input fields: "Numéro de document :", "Code ISBN :", "Titre :", "Auteur(s) :", "Collection :", "Genre :", "Public :", "Rayon :", and "Chemin de l'image :". There is also a large text area for "Image :". At the bottom of the form are buttons for "Ajouter", "Modifier", "Supprimer", "Annuler", and "Valider".

La méthode `enCoursModifLivres` appelé lors de l'ouverture de l'onglet avec comme paramètre « false », permet une grande partie de la logique événementielle de l'application.

Un booléen a été rajouté en variable globale pour permettre de différencier les ajouts des modifications. `enCoursModif` (livres, DVD, revues) le valorise toujours a « false », `btnAjouterLivres` le valorise a « true ».

```

/// <summary>
/// démarre la procédure d'ajout de livre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void btnAjouterLivres_Click(object sender, EventArgs e)
{
    enCoursModifLivres(true);
    ajouterBool = true;
    string id = PlusUnIdString(controller.GetNbLivreMax());
    if (id == "1")
        id = "00001";
    txbLivresNumero.Text = id;
    txbLivresTitre.Text = "";
    txbLivresAuteur.Text = "";
    cbxLivresPublicInfo.SelectedIndex = -1;
    txbLivresCollection.Text = "";
    cbxLivresGenresInfo.SelectedIndex = -1;
    cbxLivresRayonInfo.SelectedIndex = -1;
    txbLivresImage.Text = "";
    txbLivresIsbn.Text = "";
}

```

```

private void TabLivres_Enter(object sender, EventArgs e)
{
    lesLivres = controller.GetAllLivres();
    RemplirComboCategorie(controller.GetAllGenres(), bdgGenres, cbxLivresGenres);
    RemplirComboCategorie(controller.GetAllPublics(), bdgPublics, cbxLivresPublics);
    RemplirComboCategorie(controller.GetAllRayons(), bdgRayons, cbxLivresRayons);
    enCoursModifLivres(false); // Nouvelle methode
    RemplirLivresListeComplete();
}

```

Les boutons 'annuler' et 'valider' ne sont accessibles qu'en cours de modification. Dans la situation inverse, les trois autres boutons sont inaccessibles.

L'API est asynchrone, chaque requête est unique. En cas de requête multiple (plusieurs envoyé a la suite). Il est possible que les premières ne soient pas encore traitées, ce qui peut être problématique en cas de requête interdépendante.

Les entités composées dans la base de données (comme un livre, un DVD, ou une revue) doivent donc être échangées en une seule requête puis « dispatchées » depuis une méthode procédurale de l'API. On doit donc faire des méthodes pour les règles particulières.

```

/// <summary>
/// Créer un livre dans la BDD
/// </summary>
/// <param name="livre"></param>
/// <returns>true si opération valide</returns>

public bool CreerLivre(Livre livre)
{
    return access.CreerEntite("livre", JsonConvert.SerializeObject(livre));
}

```

```

/**
 * requete arrivée en POST (insert)
 * @param string $table nom de la table
 * @param array $champs nom et valeur des champs
 */
public function post($table, $champs){
    if ($table == "livre"){
        $result = $this->accessBDD->insertLivre($champs);
    }elseif ($table == "dvd"){
        $result = $this->accessBDD->insertDvd($champs);
    }elseif ($table == "revue"){
        $result = $this->accessBDD->insertRevue($champs);
    }else{
        $result = $this->accessBDD->insertOne($table, $champs);
    }
    if ($result == null || $result == false){
        $this->reponse(400, "requete invalide");
    }else{
        $this->reponse(200, "OK");
    }
}

```

Voilà à quoi ressemble l'application à ce stade :

The screenshot shows a web application interface with the following components:

- Navigation Bar:** Lignes DVD, Revues, Pautons des revues.
- Search Section:**
 - Rechercher: Saisir le titre ou la partie d'un titre: [input] Ou sélectionner le genre: [dropdown] [X]
 - Saisir un numéro de document: [input] Recherche Ou sélectionner le public: [dropdown] [X]
 - Ou sélectionner le rayon: [dropdown] [X]
- Table of Items:**

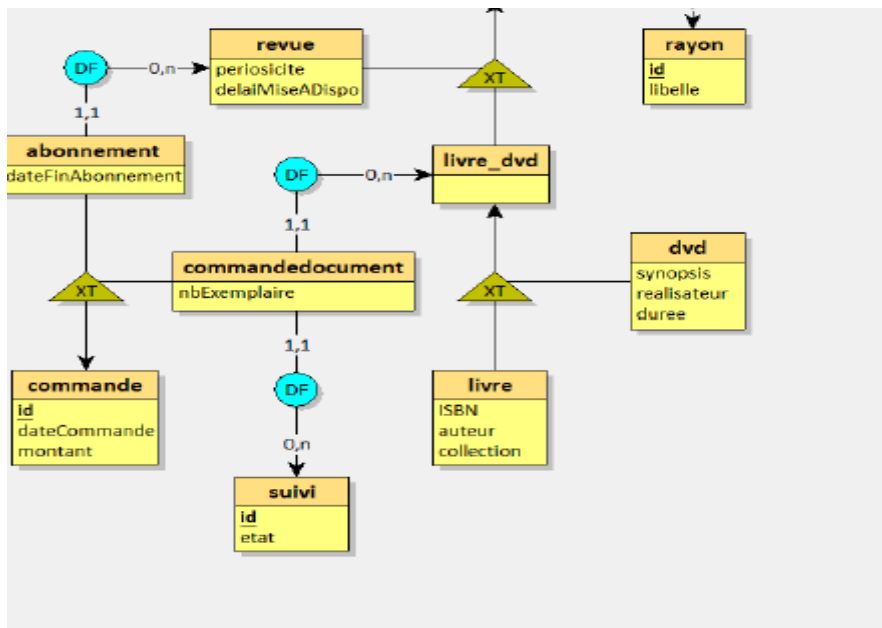
Id	Titre	Durée	Réalisateur	Genre	Public	Rayon
20003	Jurassic Park	128	Steven Spielberg	Science Fiction	Tous publics	DVD films
20002	Le seigneur des anneaux : la communauté de l'anneau	228	Peter Jackson	Fantasy	Tous publics	DVD films
20004	Matrix	136	Les Wachowski	Science Fiction	Tous publics	DVD films
20001	Star Wars 5 L'empire contre attaque	124	George Lucas	Science Fiction	Tous publics	DVD films
- Detailed View (Informations détaillées):**
 - Numéro de document: 20001 Durée: 124
 - Titre: Star Wars 5 L'empire contre attaque
 - Réalisateur(trice): George Lucas
 - Synopsis: Luc est entraîné par Yoda pendant que Han et Leia tentent de se cacher dans la cité des nuages.
 - Genre: Science Fiction
 - Public: Tous publics
 - Rayon: DVD films
 - Chemin de l'image: [input]
 - Buttons: Ajouter, Modifier, Supprimer, Annuler, Valider

Créations des triggers dans la base de données (MySQL) pour générer les contraintes d'exclusion entre les entités filles des différents héritages (Livres/DVD, Revue/Livre_DVD).

```
DROP TRIGGER if EXISTS upRevue ;
DROP TRIGGER if EXISTS insRevue ;
DROP PROCEDURE if EXISTS majRevue ;
DELIMITER //
CREATE PROCEDURE majRevue(IN Idrevue varchar(10))
BEGIN
    DECLARE nb INTEGER ;
    SELECT COUNT(*) INTO nb
    FROM livres_dvd
    WHERE BINARY id = BINARY idrevue ;
    IF (nb = 1) THEN
        SIGNAL SQLSTATE "45000"
        SET MESSAGE_TEXT = "opération impossible";
    END IF ;
END
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER insRevue
BEFORE INSERT ON revue
FOR EACH ROW
BEGIN
    CALL majRevue(NEW.id);
END
//
DELIMITER ;
```

Mission 2 : gérer les commandes :



Modification de la base de données :

Création de la table Suivi dans la base de données.

Puis ajout de la clef idSuivi dans la table commandeDocument.

Création de l'interface graphique :

L'interface graphique est divisée en plusieurs sections :

- Barre de menu :** Livres, DVD, Revues, Paiements des revues, Commandes de livres, Commandes de DVD, Abonnements.
- Section Recherche :**
 - Rechercher
 - Saisir le titre ou la partie d'un titre : [Champ de saisie]
 - Ou sélectionner : le genre : [Menu déroulant], le public : [Menu déroulant], le rayon : [Menu déroulant]
 - Saisir un numéro de document : [Champ de saisie] [Bouton Rechercher]
- Section Commandes :**
 - Titre : [Champ de saisie]
 - Numéro de commande : [Champ de saisie]
 - Date de commande : [Menu déroulant]
 - Montant : [Champ de saisie]
 - Nombre d'exemplaires : [Champ de saisie]
 - Numéro du livre : [Champ de saisie]
 - Etat de la commande : [Menu déroulant]
 - Les commandes livrées ne peuvent pas être supprimées
 - [Bouton Ajouter] [Bouton Modifier] [Bouton Supprimer]
 - [Bouton Annuler] [Bouton Valider]

Ajout et configuration des nouveaux onglets sur le modèle des précédents.

Création et configuration du premier onglet sur le modèle des précédents. Les groupes pouvant être copiés, les deux autres onglets sont une copie « configurée » du premier.

Tous les livres sont affichés dans la DataGridView située dans la partie haute. Les commandes associées au livre sélectionné dans la partie basse. Si une

commande est sélectionnée, ses informations détaillées sont affichées dans la partie en bas à droite.

```

/// <summary>
/// Sur la sélection d'une ligne ou cellule dans le grid
/// affichage des commandes du livre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void dgvLivresComListe_SelectionChanged(object sender, EventArgs e)
{
    if (dgvLivresComListe.CurrentCell != null)
    {
        try
        {
            Livre livre = (Livre)dgvLivresComListe.List[bdgLivresComListe.Position];
            AfficheLivresCommandeInfos(livre);
            txbLivresComNumLivre.Text = livre.Id;
        }
        catch
        {
            VideLivresComZones();
        }
    }
    else
    {
        txbLivresComNumLivre.Text = "";
        VideLivresComInfos();
    }
}

```

Extrait du
code

Des méthodes sont ajoutées dans Access pour effectuer les requêtes et récupérer les données.

```

/// <summary>
/// Retourne les commandes d'un livre
/// </summary>
/// <param name="idLivre"></param>
/// <returns></returns>
/// <remarks> 0 modification | Source, 0 modification
public List<CommandeDocument> GetCommandesLivres(string idLivre)
{
    string jsonIdDocument = convertToJson("id", idLivre);
    List<CommandeDocument> lesCommandesLivres = TraitementRecupereCommandeDocument(GET, "commandedocument/" + jsonIdDocument);
    return lesCommandesLivres;
}

```

```

3 references | Nicolas PECH, 4 y 4 10 jours | 1 select, 18 modifications
class FrmMediatekController
{
    Commune
    Onglet Livres
    Onglet Dvd
    Onglet Revues
    Onglet Parutions
    Commandes de livres et Dvd
    abonnements
}

```

Des régions ont été créées dans le contrôleur pour toutes les méthodes nécessaires aux trois onglets de la mission.

Coté API :

```

/**
 * récupération des lignes concernées
 * @param string $table nom de la table
 * @param array $champs nom et valeur de chaque champs de recherche
 * @return lignes répondant aux critères de recherches
 */
public function select($table, $champs)
{
    if ($this->conn != null && $champs != null)
    {
        switch ($table)
        {
            case "exemplaire" :
                return $this->selectExemplairesRevue($champs['id']);
            case "commandedocument" :
                return $this->selectCommandesDocument($champs['idLivreDvd']);
            case "abonnements" :
                return $this->selectAbonnementsRevue($champs['idRevue']);
            default:
                // cas d'un select sur une table avec recherche sur des champs
                return $this->selectTableOnConditions($table, $champs);
        }
    }
    else
    {
        return null;
    }
}

```


Gestion des suivis :

Une commande ne peut être créée avec le Suivi «en cours».

Il n'est pas possible statuer une commande en « réglé » si elle n'a pas le statut « livré ».

```
/// <summary>
/// démarre la procédure d'ajout d'une commande
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnDvdComAjouter_Click(object sender, EventArgs e)
{
    EnCoursModifDvdCom(true);
    txbDvdComNumLivres.ReadOnly = true;
    ajouterBool = true;
    string id = PlusUnIdString(controller.GetNbCommandeMax());
    if (id == "1")
        id = "00001";
    VideDvdComInfos();
    cbxDvdComEtat.SelectedIndex = 0; //selectionne le suivi "en cours"
    txbDvdComNbCommande.Text = id;
    cbxDvdComEtat.Enabled = false; //rend le ComBoBar non modifiable par l'utilisateur
}

/// <summary>
/// démarre la procédure d'ajout de commande
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnDvdComModifier_Click(object sender, EventArgs e)
{
    if (dgvDvdComListeCom.CurrentCell != null && txbDvdComNbCommande.Text != "")
    {
        //filtre les suivis inférieur a l'etat actuel
        List<Suivi> lesSuivi = controller.GetAllSuivis().FindAll(o => o.Id >= ((Suivi)cbxDvdComEtat.SelectedItem).Id).ToList();
        if (lesSuivi.Count > 2) // seule les commande livrée peuvent etre payé
            lesSuivi = lesSuivi.FindAll(o => o.Id < 4).ToList();
        EnCoursModifDvdCom(true);
        RemplirComboSuivi(lesSuivi, bdgDvdComEtat, cbxDvdComEtat);
        cbxDvdComEtat.SelectedIndex = 0;
    }
    else
    {
        MessageBox.Show("Aucune commande sélectionné");
    }
}
}
```

Quand le Suivi d'une commande passe a « livré », la création d'exemplaire se fait avec trigger, depuis la base de données.

```
DELIMITER ;
DROP TRIGGER IF EXISTS 'insExemplaire';
DELIMITER $$
CREATE TRIGGER 'insExemplaire' AFTER INSERT ON 'commandedocument' FOR EACH ROW BEGIN
    DECLARE dateAchat DATE ;
    IF (NEW.idsuivi > 2) THEN
        SELECT dateCommande INTO dateAchat FROM commande WHERE id = NEW.id ;
        CALL creerExemplaire(NEW.nbExemplaire, NEW.idLivreDvd, dateAchat);
    END IF;
END
$$
DELIMITER ;
DROP TRIGGER IF EXISTS 'upExemplaire';
DELIMITER $$
CREATE TRIGGER 'upExemplaire' AFTER UPDATE ON 'commandedocument' FOR EACH ROW BEGIN
    DECLARE dateAchat DATE ;
    IF (OLD.idsuivi < 3) THEN
        IF (NEW.idsuivi > 2) THEN
            SELECT dateCommande INTO dateAchat FROM commande WHERE id = NEW.id ;
            CALL creerExemplaire(NEW.nbExemplaire, NEW.idLivreDvd, dateAchat);
        END IF;
    END IF;
END
$$
DROP PROCEDURE IF EXISTS 'creerExemplaire'$$
CREATE DEFINER='root'@'localhost' PROCEDURE 'creerExemplaire' (IN 'nbExemplaire' INTEGER, IN 'idDocument'
    VARCHAR(10), IN 'dateAchatC' DATE) BEGIN
    DECLARE nb INTEGER;
    DECLARE maxId INTEGER;
    SET nb = 0;
    WHILE nb < nbExemplaire DO
        SELECT MAX(numero) INTO maxId FROM exemplaire WHERE id = idDocument;
        IF (maxId IS NULL) THEN
            SET maxId = 0;
        END IF;
        SET maxId = maxId + 1;
        INSERT INTO exemplaire(id, numero, dateAchat, idEtat, photo)
        VALUES (idDocument, maxId, dateAchatC, '00001', "");
        SET nb = nb + 1;
    END WHILE ;
END
$$
```


Etat de l'application à ce moment là :

Gestion des documents de la médiathèque

Liens: DVD, Revues, Partitions des revues, Commandes de livres, Commandes de DVD, Abonnements

*Rechercher

Saisir le titre ou la partie d'un titre : Revises dont un abonnement se termine dans moins de 30 jours le genre : X

Saisir un numéro de document : Rechercher le public : X le rayon : X

Id	Titre	Périodicité	DélaiMiseADispo	Genre	Public	Rayon
10002	Alternatives Economiques	MS	52	Presse Economique	Adultes	Magazines
10001	Arts Magazine	MS	52	Presse Culturelle	Adultes	Magazines
10003	Challenges	HB	15	Presse Economique	Adultes	Magazines
10011	Géo	MS	52	Presse Culturelle	Tous publics	Beaux Livres
10013	heygg toi la	MS	2	Bande dessinée	Ados	Beaux Livres
10009	L'Equipe	QT	5	Presse sportive	Adultes	Presse quotidienne
10010	L'Equipe Magazine	HB	12	Presse sportive	Adultes	Magazines
10006	L'Océ	HB	26	Actualités	Adultes	Magazines

Alternatives Economiques

Filtrer les abonnements se terminant dans moins de 30 jours :

Id	DateCommande	DateFinAbonnement	Montant
00023	26/11/2023	31/01/2024	21.2
00024	13/03/2023	08/02/2024	2

Numéro de commande : 00023

Date de commande : lundi 20 novembre 2023

Montant : 21.2

Fin d'abonnement : mercredi 31 janvier 2024

N° Revue : 10002

Pour chaque revue, si un de ses abonnements arrive a expiration dans moins d'un mois, son titre est rajouté au message qui sera affiché par MessageBox. Si aucun titre n'est rajouté, rien n'est affiché.

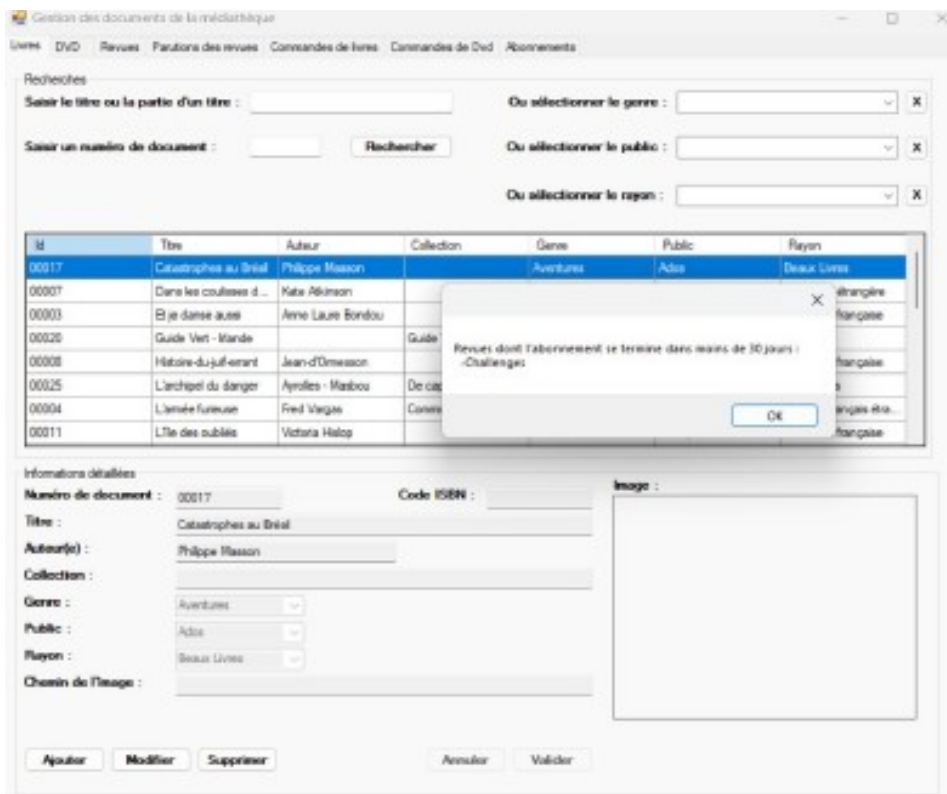
```

/// <summary>
/// Cuvre une MessageBox au lancement de FrmMediatek.cs
/// si des abonnements sont proches de se terminer
/// </summary>

private void afficherAlerteAbo()
{
    bool interrupteur = false;
    List<Revue> revues = controller.GetAllRevises();
    string alerteRevises = "Revises dont l'abonnement se termine dans moins de 30 jours : \n";
    foreach (Revue revue in revues)
    {
        List<Abonnement> abonnements = controller.GetAbonnements(revue.Id);
        if (abonnements.FindAll(o => {o.DateFinAbonnement <= DateTime.Now.AddMonths(1)}
            && (o.DateFinAbonnement >= DateTime.Now)).Count > 0)
        {
            alerteRevises += " - " + revue.Titre + "\n";
            interrupteur = true;
        }
    }

    if(interrupteur)
        MessageBox.Show(alerteRevises);
}

```



Message d'avertissement

Vérification qu'aucun abonnement n'est en cours avant suppression (dans la vue de l'application C#).

La méthode `VerrifExemplaireAbo` renvoie vrai ou faux à la question : un exemplaire de la revue, a-t-il été acquis pendant la période où l'abonnement est en cours ?

Dans le cas où cela est vrai, l'abonnement ne peut être supprimé.

```

/// <summary>
/// renvoie vrai si un exemplaire a été acquis pendant la période de l'abonnement
/// </summary>
/// <param name="abonnement"></param>
/// <returns></returns>

private bool VerrifExemplaireAbo(Abonnement abonnement)
{
    List<Exemplaire> lesExemplairesAbo = controller.GetExemplairesRevue(abonnement.IdRevue);
    return lesExemplairesAbo.FindAll(o => (o.DateAchat >= abonnement.DateCommande) && (o.DateAchat <= abonnement.DateCommande)).Count > 0;
}

/// <summary>
/// annule les modifications en cours
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void btnAboSupprimer_Click(object sender, EventArgs e)
{
    Abonnement abonnement = (Abonnement)bdgAboListeCommande[bdgAboListeCommande.Position];
    if (dgvAboListeCom.CurrentRow != null && txtAboNbCommande.Text != "")
    {
        if (VerrifExemplaireAbo(abonnement))
        {
            MessageBox.Show("Une revue a été livrée le temps de cet abonnement, il ne peut être supprimée");
        }
        else if (MessageBox.Show("Êtes vous sûr de vouloir supprimer la commande n°" + abonnement.Id +
            " concernant " + lesRevueAbo.Find(o => o.Id == abonnement.IdRevue).Titre + " ?",
            "Validation suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            if (controller.SupprimerAbonnement(abonnement))
            {
                Thread.Sleep(50);
                try
                {
                    Revue Revue = (Revue)bdgAboListe.List[bdgAboListe.Position];
                    AfficheAboInfos(Revue);
                    txtAboNumRevue.Text = Revue.Id;
                }
                catch
                {
                    VideAboZones();
                }
            }
            else
            {
                MessageBox.Show("Erreur");
            }
        }
    }
    else
    {
        MessageBox.Show("Veuillez sélectionner une abonnement");
    }
}

```

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

Recherches

Saisir le titre ou la partie d'un titre : Ou sélectionner le genre :

Saisir un numéro de document : Rechercher Ou sélectionner le public :

Ou sélectionner le rayon :

Informations détaillées

Numéro de document : Code ISBN : Image :

Titre :

Auteur(s) :

Collection :

Genre :

Public :

Rayon :

Chemin de l'image :

Ajouter Modifier Supprimer Annuler Valider

Exemplaires

Numéro d'exemplaire :

Date d'achat :

Photo :

Etat de l'exemplaire :

Supprimer Modifier

Les exemplaire étant déjà utilisés par l'application, les méthodes pour consulter existent déjà (classe existante coté C#, méthode `selectExemplairesRevue` dans l'API).

```
#region Onglet Livres
private readonly BindingSource bdsLivresListe = new BindingSource();
private List< Livre > lesLivres = new List< Livre >();
private readonly BindingSource bdsGenresInfo = new BindingSource();
private readonly BindingSource bdsPublicsInfo = new BindingSource();
private readonly BindingSource bdsRayonsInfo = new BindingSource();
private readonly BindingSource bdsFormalistes = new BindingSource(); // pour "bind" ou débind les listes avec la liste en dessous
private List< Exemplaire > lesExemplaires = new List< Exemplaire >(); // pour remplir ma datagridview du bas
private List< Etat > lesEtats = new List< Etat >(); // pour remplir mon combobox des etats en bas
```

Les États :

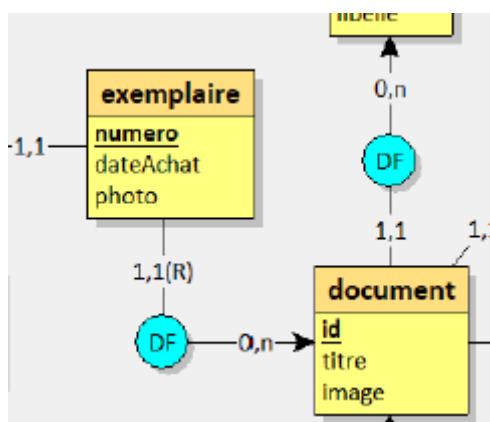
Création de la classe Etat

```
public Etat(string id, string libelle)
{
    this.Id = id;
    this.Libelle = libelle;
}

/// <summary>
/// Récupération du libellé pour l'affichage dans les combos
/// </summary>
/// <returns>Libelle</returns>
0 références | 0 modification | 0 auteur, 0 modification
public override string ToString()
{
    return this.Libelle;
}
```

Modification d'exemplaire :

Un exemplaire étant composé de deux clefs primaire, l'API va devoir être modifiée pour que les requêtes PUT (modification) puissent prendre en compte deux clefs.



```

/**
 * requete arrivée en PUT (update)
 * @param string $table nom de la table
 * @param string $id valeur de l'id
 * @param array $champs nom et valeur des champs
 */
public function put($table, $id, $champs){
    if ($table == "livre"){
        $result = $this->access800->updateLivres($id, $champs);
    }elseif ($table == "dvd"){
        $result = $this->access800->updateDvd($id, $champs);
    }elseif ($table == "revue"){
        $result = $this->access800->updateRevue($id, $champs);
    }elseif ($table == "commandedocument"){
        $result = $this->access800->updateCommande($id, $champs);
    }elseif ($table == "abonnement"){
        $result = $this->access800->updateAbonnement($id, $champs);
    }elseif ($table == "exemplaire"){
        $result = $this->access800->updateOne($table, $id, $champs, $champs["Numero"]);
    }else{
        $result = $this->access800->updateOne($table, $id, $champs);
    }
    if ($result == null || $result == false){
        $this->reponse(400, "requete invalide");
    }else{
        $this->reponse(200, "OK");
    }
}
  
```

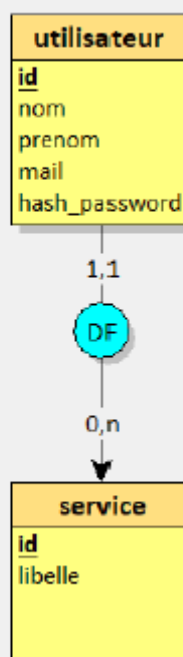
```

/** <summary>
 * Update un exemplaire dans la bdd
 * </summary>
 * @param name="exemplaire"</param>
 * @returns</returns>
 * Référence : 0 modification, 0 auteur, 0 modification
 */
public bool UpdateExemplaire(Exemplaire exemplaire)
{
    return access.UpdateEntite("exemplaire", exemplaire.Id, JsonConvert.SerializeObject(exemplaire, new CustomDateTimeConverter()));
}

/** <summary>
 * Supprime un exemplaire dans la bdd
 * </summary>
 * @param name="exemplaire"</param>
 * @returns</returns>
 * Référence : 0 modification, 0 auteur, 0 modification
 */
public bool SupprimerExemplaire(Exemplaire exemplaire)
{
    return access.SupprimerEntite("exemplaire", JsonConvert.SerializeObject(exemplaire, new CustomDateTimeConverter()));
}
  
```

Mission 4 : mettre en place des authentifications :

Structure de la table utilisateur et service, indépendants des autres tables.



```

1 DROP TABLE IF EXISTS 'service';
2 CREATE TABLE IF NOT EXISTS 'service' (
3   'id' varchar(5) NOT NULL,
4   'libelle' varchar(20) NOT NULL,
5   PRIMARY KEY ('id')
6 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
7
8 DROP TABLE IF EXISTS 'utilisateur';
9 CREATE TABLE IF NOT EXISTS 'utilisateur' (
10  'id' varchar(10) NOT NULL,
11  'nom' varchar(20) NOT NULL,
12  'prenom' varchar(20) NOT NULL,
13  'mail' varchar(20) NOT NULL,
14  'password' varchar(100) NOT NULL,
15  'idservice' varchar(5) NOT NULL,
16  PRIMARY KEY ('id'),
17  KEY 'idservice' ('idService')
18 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
19
20 ALTER TABLE 'utilisateur'
21 ADD CONSTRAINT 'utilisateur_ibfk_1' FOREIGN KEY ('idService') REFERENCES 'service' ('id');
22

```

Création de la classe Utilisateur. N'utilisant pas d'affichage, elle n'a pas besoin d'instancier un service, et n'a pas besoin toString().

```

public class Utilisateur
{
    /// <summary>0
    /// Valorise les propriétés
    /// </summary>
    /// <param name="id"></param>
    /// <param name="nom"></param>
    /// <param name="prenom"></param>
    /// <param name="mail"></param>
    /// <param name="idService"></param>
    /// <param name="service"></param>
    2 références Nicolas FREFE, il ya 15 jours | 1 auteur, 2 modifications
    public Utilisateur(string id, string nom, string prenom, string mail, string idService, string service)
    {
        this.Id = id;
        this.Nom = nom;
        this.Prenom = prenom;
        this.Mail = mail;
        this.Service = new Service(idService, service);
    }

    public string Id { get; }

    public string Nom { get; set; }

    public string Prenom { get; set; }

    public string Mail { get; set; }

    public Service Service { get; }
}

```

Nouvelle interface de connexion :

The image shows a simple login window with a title bar that says 'FirstLogin'. Inside the window, there are two text input fields. The first one is labeled 'Identifiant' and the second one is labeled 'Password'. Below these fields is a button labeled 'Connexion'.

Import de
System.Security.Cryptography, pour utiliser les fonctions d'encodage.

Si la tentative de connexion est valide. Que le hash et salt du password donnée par l'utilisateur lui corresponde dans la base de données. La vue principale est instanciée avec l'utilisateur en paramètre.

```
/// <summary>
/// Lance la vue principale
/// </summary>

private void Init()
{
    FrmMediatek mediatek = new FrmMediatek(utilisateur);
    mediatek.Show();
}

/// <summary>
/// Retourne vrai si l'utilisateur existe dans la BDD
/// </summary>
/// <param name="mail"></param>
/// <param name="password"></param>
/// <returns></returns>

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;
}
```

Fonction pour encrypter le mot de passe et le retourner encodé en base64 au format « string » .

```
/// <summary>
/// Retourne le hash au format string
/// </summary>
/// <param name="hashAlgorithm"></param>
/// <param name="input"></param>
/// <returns></returns>

private static string GetHash(HashAlgorithm hashAlgorithm, string input)
{
    // Convert the input string to a byte array and compute the hash.
    byte[] data = hashAlgorithm.ComputeHash(Encoding.UTF8.GetBytes(input));

    // Create a new StringBuilder to collect the bytes
    // and create a string.
    var sBuilder = new StringBuilder();

    // Loop through each byte of the hashed data
    // and format each one as a hexadecimal string.
    for (int i = 0; i < data.Length; i++)
    {
        sBuilder.Append(data[i].ToString("x2"));
    }

    // Return the hexadecimal string.
    return sBuilder.ToString();
}
```

Pour gérer les droits, FrmMediatek est instancié avec l'utilisateur connecté. Des méthodes (à l'ouverture du programme, ou dans les TabOngletEnter()) permettent de restreindre l'accès.

```
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 BindingSource bdgEtats = new BindingSource();
    private List<Etat> lesEtatsEx = new List<Etat>();
    private readonly Utilisateur utilisateur;
    private bool ajouterBool = false;
    private bool modifierEtat = false;
    private bool firstLoad = true;

    /// <summary>
    /// Constructeur : création du contrôleur lié à ce formulaire
    /// </summary>

    public FrmMediatek(Utilisateur lutilisateur)
    {
        InitializeComponent();
        this.controller = new FrmMediatekController();
        this.utilisateur = lutilisateur;
        VerifDroitAccueil(lutilisateur);
    }
}
```

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

Pour sécuriser le code de l'API, nous ne voulons pas que l'utilisateur puisse accéder à l'ensemble de ses fichiers. Pour éviter cela, redirection de la racine vers la page de base « 404 erreur » apache2 dans htaccess. _« RewriteRule ^\$ mediatekdocuments.php?error=404 »

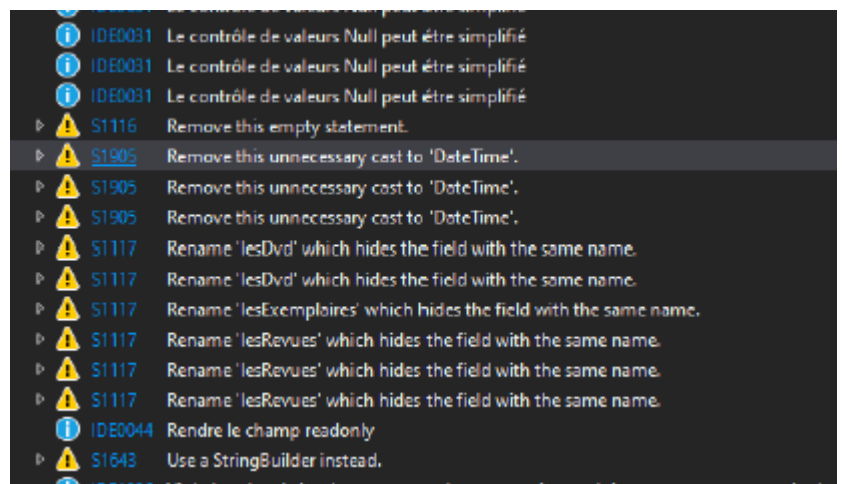
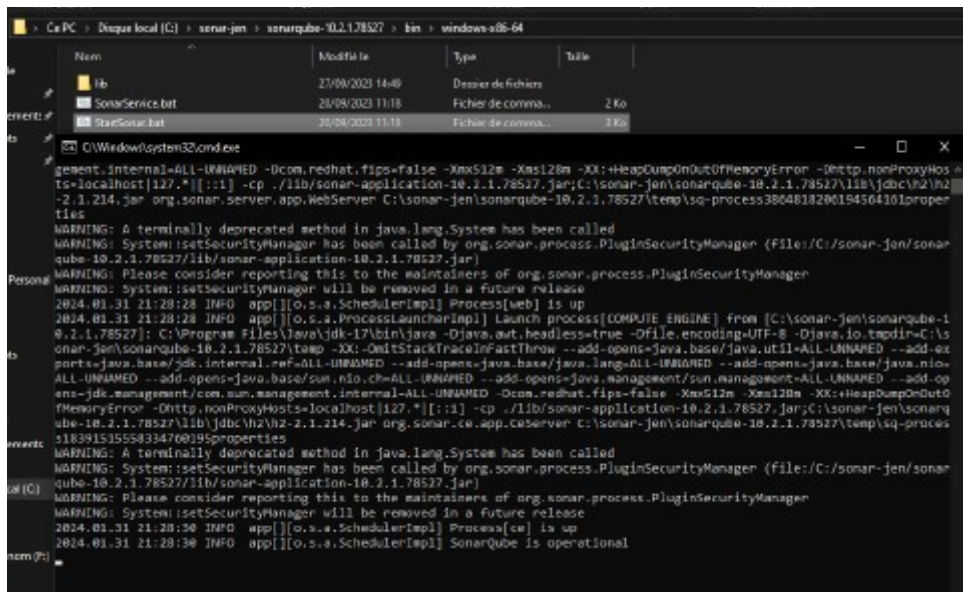
Du côté de l'application C# les identifiants de connexion sont déplacés dans le fichier app.config.

```
/// <summary>
/// adresse de l'API
/// </summary>
private static readonly string urlApiName = "MediatekDocuments.Properties.Settings.mediatekconnectionstring";
/// <summary>
/// url et login de l'API
/// </summary>
private static readonly string authenticationName = "MediatekDocuments.Properties.Settings.mediatekAuthenticationString";
/// <summary>
```

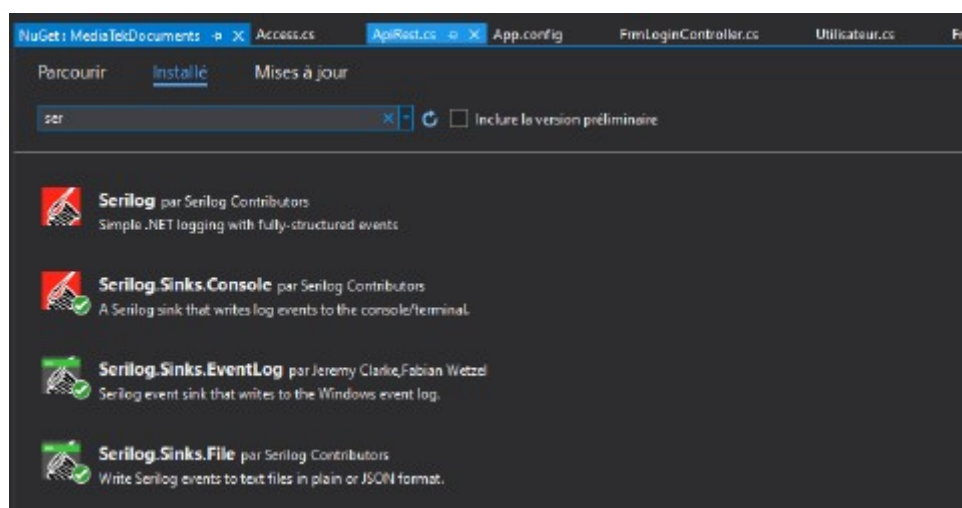
```
static string GetAuthenticationString(string name)
{
    string returnValue = null;
    connectionstringssettings settings = ConfigurationManager.ConnectionStrings[name];
    if (settings != null)
        returnValue = settings.ConnectionString;
    return returnValue;
}
```


Contrôler la qualité :

Analyse du code avec SonarQube. Lancement du serveur local.



Mise en place des logs :



```
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 erreur={0}", e.Message);
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}
```

L'écriture dans le fichier de log est initialisée avec le constructeur d'Access. Les logs sont capturés quand le programme rencontre des erreurs.

M

```

{
    Console.WriteLine("code erreur = " + code + " message = " + (String)retour["message"]);
    Log.Error("Access.TraitementAccup code erreur = " + code + " message = " + (String)retour["message"]);
}
catch (Exception e)
{
    Console.WriteLine("Erreur lors de l'accès à l'API : " + e.Message);
    Log.Fatal("Erreur lors de l'accès à l'API : " + e.Message);
    Environment.Exit(0);
}

return liste;
}
```

Mission 6 : tester et documenter :

Tests Unitaire :

Test	Date	Cacher...	Message
MediaTekDocumentsTests (20)	15 ms		
MediaTekDocumentsTestsTests	15 ms		
MediaTekDocumentsTestsTests	8 ms		
MediaTekDocumentsTestsTests	8 ms		
CategoryTests (2)	< 1 ms		
CategoryTest	< 1 ms		
TalkingTest	< 1 ms		
CommandeDocumentTests (1)	1 ms		
CommandeTests (3)	< 1 ms		
DocumentTests (1)	< 1 ms		
DvdTests (1)	< 1 ms		
DvdTest	< 1 ms		
EtatTests (2)	< 1 ms		
EtatTest	< 1 ms		
TalkingTest	< 1 ms		
ExemplaireTests (1)	< 1 ms		
GenreTests (1)	< 1 ms		
GenreTest	< 1 ms		
UtilisateurTests (1)	< 1 ms		
UtilisateurTest	< 1 ms		
ServiceTests (2)	< 1 ms		
ServiceTest	< 1 ms		
TalkingTest	9 ms		
SuiviTests (2)	< 1 ms		
UtilisateurTests (1)	< 1 ms		

Pour toutes les classes modèle, test de l'instanciation et des méthodes.

MediaTekDocumentsTests
Properties
Références
model
AbonnementTests.cs
CategorieTests.cs
CommandeDocumentTests.cs
CommandeTests.cs
DocumentTests.cs
DvdTests.cs
EtatTests.cs
ExemplaireTests.cs
GenreTests.cs
PublicTests.cs
RayonTests.cs
RevueTests.cs
ServiceTests.cs
SuiviTests.cs
UtilisateurTests.cs
app.config
packages.config

Documentation technique :

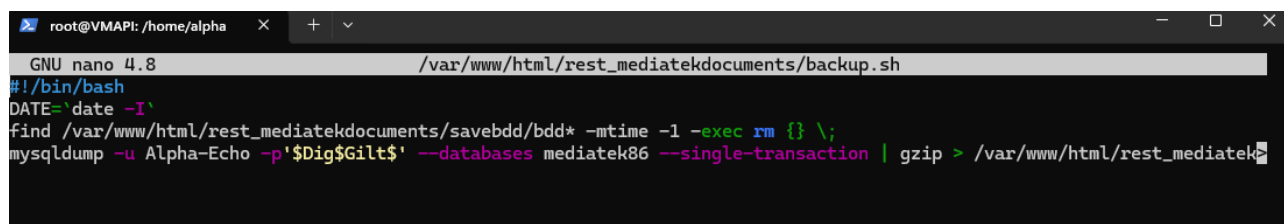
Les documentations techniques sont disponibles sur la page de présentation du projet.

Mission 7 : déployer et gérer les sauvegardes de données :

L'API a pour nom de domaine a l'adresse `vm-alpha.francecentral.cloudapp.azure.com`

Elle est hébergée dans une VM Azure Ubuntu cependant l'accès à cette VM par l'application retourne une erreur 500 l'application n'est donc pas entièrement fonctionnelle.

Les sauvegardes la bdd se fait via un script bash lancé par un cronjob

A screenshot of a terminal window titled 'root@VMAPI: /home/alpha'. The terminal shows the GNU nano 4.8 editor editing a file at '/var/www/html/rest_mEDIATEKdocuments/backup.sh'. The script content is as follows:

```
#!/bin/bash
DATE=$(date +%I)
find /var/www/html/rest_mEDIATEKdocuments/savebdd/bdd* -mtime -1 -exec rm {} \;
mysqldump -u Alpha-Echo -p'$Dig$Gilt$' --databases mediatek86 --single-transaction | gzip > /var/www/html/rest_mEDIATEK
```