

Bilan Atelier 2

Étant donné que je suis seul sur le projet je me suis permis de ne pas réaliser la Mission 1 et 3.

Contexte du projet

L'application MediatekDocuments est un outil de gestion destiné au personnel du réseau de médiathèques Mediatek86. L'application existante permettait la consultation du catalogue (et la réception des parutions de revues. Son architecture était la suivante : une base de données, une api REST, une application cliente en C#.

Le demande

Faire évoluer cette application pour y intégrer un système de gestion des commandes, gérer les abonnements aux revues et sécuriser les accès par un système d'authentification.

Technologies et Langages Utilisés

BDD : MySQL (WampServer en local et AwardSpace pour l'api en ligne)

API : PHP

Application : C#, windows forms

Outils : Visual studio 2026, NetBeans, phpMyAdmin

Qualité et Tests : SonarLint, Serilog, MSTest, Postamn

Versioning et Documentation : Git/Github, Sandcastle , phpDocumentor

Bilan Atelier 2	1
Contexte du projet	1
Mission 2 Tache 1 Gérer les commandes de livres ou de DVD	4
La requête SQL.....	4
Modification dans l'api.....	6
Modification C#.....	7
Gérer les commandes de revues (Mission 2.2).....	12
Modification dans l'api.....	13
Modifications dans le contrôleur, model et dal	15
Modification de la vue	16
Mission 4 : mettre en place des authentifications.....	21
Ajout du SQL.....	22
Ajout dans l'API.....	23
Ajout en C#	23
Mission 5 Assurer la sécurité, la qualité et intégrer des logs	29
Tache 2 – Contrôler la qualité	32
Tache 3 - Intégrer les logs.....	34
Mission 6 – Tester et documenter.....	35
Test unitaires C#.....	36
Tests Postman API.....	37
Générer la documentation technique	38
Création de la vidéo	39
Mission 7 – Déployer et gérer les sauvegardes de données.....	39
Déploiement de l'API en ligne	39
Création de l'installateur	41
Automatiser la sauvegarde de la BDD	41
Conclusion.....	42

Mission 2 Tache 1 Gérer les commandes de livres ou de DVD



La requête SQL

Création de la table etape_suivi et ajout d'une propriété rang pour tri logique (1,2,3,4)

Elle est ensuite relié à commandedocument via la clé étrangère idSuivi pour garantir l'intégrité.

```
-- 1. Création de la table de suivi des étapes
CREATE TABLE IF NOT EXISTS etape_suivi (
  id varchar(5) NOT NULL,
  intitule varchar(20) NOT NULL,
  rang tinyint NOT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY uk_etape_intitule (intitule)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO etape_suivi (id, intitule, rang) VALUES
('00001', 'En cours', 1),
('00002', 'Relancée', 2),
('00003', 'Livrée', 3),
('00004', 'Réglée', 4);

-- 2. Mise à jour de la table commandedocument
ALTER TABLE commandedocument |
ADD idSuivi varchar(5) NOT NULL DEFAULT '00001',
ADD KEY idx_suivi (idSuivi),
ADD CONSTRAINT fk_cmd_suivi FOREIGN KEY (idSuivi) REFERENCES etape_suivi (id);
```

Création de plusieurs trigger pour bloquer certaines situations (ex : une commande livrée ne peut redevenir « en cours » ou encore elle doit être livrée avec d'être réglée). Si une règle est trigger ça lève une erreur sql (SIGNAL SQLSTATE '45000 ')

```
-- 3. Trigger : Sécurités sur les changements d'états
DELIMITER $$
CREATE TRIGGER trg_verif_suivi_update
BEFORE UPDATE ON commandedocument
FOR EACH ROW
BEGIN
    -- Empêcher le retour en arrière depuis Livrée ou Réglée
    IF (OLD.idSuivi = '00003' AND NEW.idSuivi IN ('00001','00002'))
        OR (OLD.idSuivi = '00004' AND NEW.idSuivi <> '00004') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Changement de statut invalide : retour arrière interdit.';
    END IF;
    -- Obligation d'être Livrée avant d'être Réglée
    IF NEW.idSuivi = '00004' AND OLD.idSuivi <> '00003' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La commande doit être livrée avant paiement.';
    END IF;
END$$

-- 4. Trigger : Sécurité suppression
CREATE TRIGGER trg_verif_suppression_cmd
BEFORE DELETE ON commandedocument
FOR EACH ROW
BEGIN
    IF OLD.idSuivi >= '00003' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Suppression bloquée : la commande est déjà finalisée.';
    END IF;
END$$

-- 5. Trigger : Nettoyage automatique
CREATE TRIGGER trg_clean_cmd_after_delete
AFTER DELETE ON commande
FOR EACH ROW
BEGIN
    DELETE FROM commandedocument WHERE id = OLD.id;
END$$
```

Lorsqu'une commande est à l'état livrée ce trigger exécute une boucle while pour insérer autant de ligne que nécessaire dans la table exemplaire. Il calcule ensuite le prochain numéro d'exemplaire disponible et affecte l'état « Neuf ».

```
-- 6. Trigger : Génération des exemplaires
CREATE TRIGGER trg_generation_exemplaires_auto
AFTER UPDATE ON commandedocument
FOR EACH ROW
BEGIN
    DECLARE date_cmd DATE;
    DECLARE num_max INT DEFAULT 0;
    DECLARE boucle INT DEFAULT 1;

    IF NEW.idSuivi = '00003' AND OLD.idSuivi <> '00003' THEN
        SELECT dateCommande INTO date_cmd FROM commande WHERE id = NEW.id;
        SELECT IFNULL(MAX(numero),0) INTO num_max FROM exemplaire WHERE id = NEW.idLivreDvd;

        WHILE boucle <= NEW.nbExemplaire DO
            INSERT INTO exemplaire (id, numero, dateAchat, photo, idEtat)
            VALUES (NEW.idLivreDvd, num_max + boucle, date_cmd, '', '00001');
            SET boucle = boucle + 1;
        END WHILE;
    END IF;
END$$
DELIMITER ;
```

Modification dans l'api

Pour servir de pont entre les erreurs générés par les Triggers SQL et l'application C# j'ai créer un classe AppException. Dans le contrôleur (Controle.php), j'intercepte cette exception spécifique pour renvoyer un code HTTP 400 avec le message d'erreur clair du Trigger.

```
public function demande(string $methodeHTTP, string $table, ?string $id, ?array $champs){
    try {
        $result = $this->myAccessBDD->demande($methodeHTTP, $table, $id, $champs);
        $this->controleResult($result);
    } catch (AppException $ex) {
        // si règle métier bloquée par un trigger
        $this->reponse(400, $ex->getMessage());
    } catch (Exception $e) {
        // Si c'est une erreur de code ou de serveur
        $this->reponse(500, "erreur serveur ou base de données");
    }
}
```

Dans MyAccessBDD.php j'ai créer une route spécifique pour l'insertion des commandes.

L'api calcule le prochain id disponible au format « 0000X »

```
protected function traitementInsert(string $table, ?array $champs) : ?int{
    switch($table){
        case "gestion_cmd":
            $id = $this->creerIdAuto();
            $res1 = $this->insertOneTupleOneTable("commande", [
                "id" => $id,
                "dateCommande" => $champs["dateCommande"],
                "montant" => $champs["montant"]
            ]);

            if ($res1 !== null) {
                $res2 = $this->insertOneTupleOneTable("commandedocument", [
                    "id" => $id,
                    "nbExemplaire" => $champs["nbExemplaire"],
                    "idLivreDvd" => $champs["idLivreDvd"]
                ]);

                if ($res2 !== null) {
                    return 1;
                } else {
                    $this->deleteTuplesOneTable("commande", ["id" => $id]);
                    return null;
                }
            }

            return null;

        default:
            return $this->insertOneTupleOneTable($table, $champs);
    }
}
```


Modification C#

Création d'une classe CommandeDocument qui hérite de Commande et utilisation des JsonProperty de la librairie Newtonsoft.json.

```
using Newtonsoft.Json;
using System;

namespace MediaTekDocuments.model
{
    20 references
    public class CommandeDocument : Commande
    {
        [JsonProperty("nbExemplaire")]
        3 references
        public int NbExemplaire { get; set; }
        [JsonProperty("idLivreDvd")]
        1 reference
        public string IdLivreDvd { get; set; }
        [JsonProperty("idSuivi")]
        3 references
        public string IdSuivi { get; set; }

        [JsonProperty("libelle")]
        1 reference
        public string LibelleSuivi { get; set; }

        0 references
        public CommandeDocument(string id, DateTime dateCommande, double montant, int nbExemplaire, string idLivreDvd, string idSuivi, string libelleSuivi)
            : base(id, dateCommande, montant)
        {
            this.NbExemplaire = nbExemplaire;
            this.IdLivreDvd = idLivreDvd;
            this.IdSuivi = idSuivi;
            this.LibelleSuivi = libelleSuivi;
        }
    }
}
```

Ajout dans Acess.cs pour pouvoir interagir avec les nouvelles routes de l'api.

```
public List<Suivi> GetSuivis()
{
    return TraitementRecup<Suivi>(GET, "niveaux_suivi", null);
}

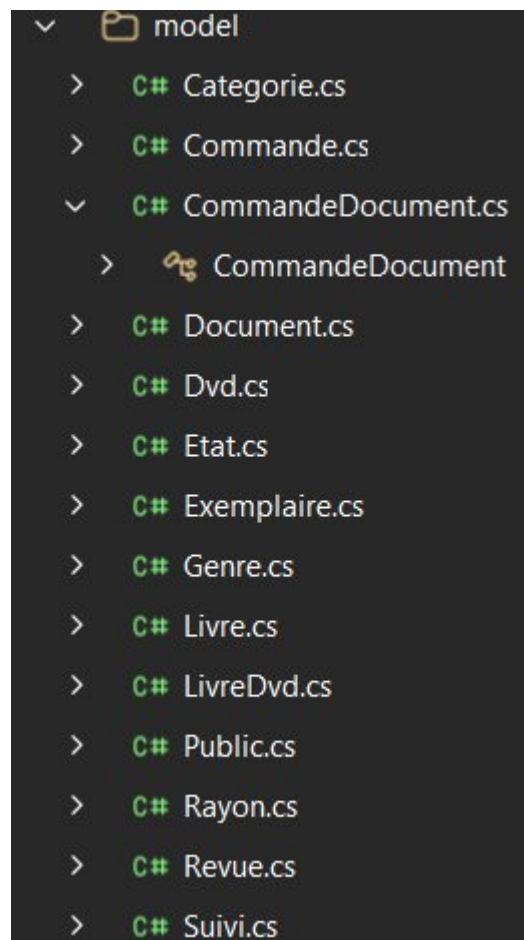
/// <summary>
/// Récupère l'historique des commandes d'un document (Livre ou DVD)
/// </summary>
1 reference
public List<CommandeDocument> GetCommandes(string idDoc)
{
    string jsonId = convertToJson("id", idDoc);
    return TraitementRecup<CommandeDocument>(GET, "commandes_doc/" + jsonId, null);
}

/// <summary>
/// Enregistre une nouvelle commande via la route 'gestion_cmd'
/// </summary>
1 reference
public bool EnregistrerCommande(object data)
{
    string json = JsonConvert.SerializeObject(data, new CustomDateTimeConverter());
    return TraitementRecup<CommandeDocument>(POST, "gestion_cmd", "champs=" + json) != null;
}

/// <summary>
/// Modifie l'étape de suivi d'une commande
/// </summary>
1 reference
public bool ModifierSuiviCommande(string idCommande, string idSuivi)
{
    string jsonIdSuivi = convertToJson("idSuivi", idSuivi);
    return TraitementRecup<CommandeDocument>(PUT, "commandedocument/" + idCommande, "champs=" + jsonIdSuivi) != null;
}

/// <summary>
/// Supprime une commande
/// </summary>
1 reference
public bool SupprimerCommande(string idCommande)
{
    string jsonIdCommande = convertToJson("id", idCommande);
    return TraitementRecup<Commande>(DELETE, "commande/" + jsonIdCommande, null) != null;
}
```

Création également des nouvelles classes nécessaire pour les ajouts de la BDD.



Création d'une interface graphique pour les deux onglets commandes livres et DVD

Exemplaires	Étape	Numéro	Date	Montant
3	En cours	00001	21/03/2026	20

groupBox1

Numéro Commande : 00001

Date : samedi 21 mars 2026

Montant : 20.00

Exemplaires : 3

Étape :

Ajouter

Modifier

Supprimer

Pour ça création de deux régions dans le fichiers FrmMediatek.cs contenant la logique pour les boutons et champs.

```
#region Onglet Commandes Livres

private readonly BindingSource bdgCommandesLivres = new BindingSource();

1 reference
private void tabCommandesLivres_Enter(object sender, EventArgs e)
{
    cbxLivresCommandesSuivi.DataSource = controller.GetSuivis();
    cbxLivresCommandesSuivi.DisplayMember = "Libelle";
    cbxLivresCommandesSuivi.ValueMember = "Id";
}
```


Cette méthode interroge l'api via le contrôleur pour récupérer l'historique des commandes d'un livre spécifique. Masque également les clés étrangères et renomme les en-têtes pour l'utilisateur

```
4 references
private void ChargerCommandesLivres()
{
    string idDoc = txbCommandesLivresNumero.Text;
    List<CommandeDocument> commandes = controller.GetCommandes(idDoc);

    dgvLivresCommandes.DataSource = null;
    bdgCommandesLivres.DataSource = commandes;
    dgvLivresCommandes.DataSource = bdgCommandesLivres;

    if (dgvLivresCommandes.Columns.Count > 0)
    {
        dgvLivresCommandes.Columns["IdLivreDvd"].Visible = false;
        dgvLivresCommandes.Columns["IdSuivi"].Visible = false;

        dgvLivresCommandes.Columns["Id"].HeaderText = "Numéro";
        dgvLivresCommandes.Columns["DateCommande"].HeaderText = "Date";
        dgvLivresCommandes.Columns["Montant"].HeaderText = "Montant";
        dgvLivresCommandes.Columns["NbExemplaire"].HeaderText = "Exemplaires";
        dgvLivresCommandes.Columns["LibelleSuivi"].HeaderText = "Étape";
        dgvLivresCommandes.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    }
}
```

En appuyant sur rechercher cette méthode vérifie que le numéro saisi correspond bien à un livre existant . Si oui affiche le titre et appelle ChergerCommandesLivres.

```
1 reference
private void btnCommandesLivresRechercher_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(txbCommandesLivresNumero.Text))
    {
        Livre livre = controller.GetAllLivres().Find(x => x.Id == txbCommandesLivresNumero.Text);
        if (livre != null)
        {
            lblCommandesLivresTitre.Text = livre.Titre;
            ChargerCommandesLivres();
        }
        else
        {
            MessageBox.Show("Livre introuvable.");
            lblCommandesLivresTitre.Text = "";
            bdgCommandesLivres.DataSource = null;
        }
    }
}
```

Cette méthode gère la synchronisation. A chaque fois que l'utilisateur clique sur une ligne du tableau elle récupère l'objet CommandeDocument correspondant et inject ses valeurs dans les champs de saisie du formulaire en bas de la page.

```
1 reference
private void dgvLivresCommandes_SelectionChanged(object sender, EventArgs e)
{
    if (dgvLivresCommandes.CurrentCell != null && bdgCommandesLivres.List.Count > 0)
    {
        CommandeDocument cmd = (CommandeDocument)bdgCommandesLivres.List[bdgCommandesLivres.Position];
        txbLivresCommandesNumeroCmd.Text = cmd.Id;
        dtplivresCommandesDate.Value = cmd.DateCommande;
        nudLivresCommandesMontant.Value = (decimal)cmd.Montant;
        nudLivresCommandesExemplaires.Value = cmd.NbExemplaire;
        cbxLivresCommandesSuivi.SelectedValue = cmd.IdSuivi;
    }
    else
    {
        txbLivresCommandesNumeroCmd.Text = "";
        nudLivresCommandesMontant.Value = 0;
        nudLivresCommandesExemplaires.Value = 1;
    }
}
```

Cette méthode récupère les valeurs saisies par l'utilisateur et les envoie au contrôleur et gère également le retour de l'api. Si succès rafraîchissement du tableau et fait apparaître la nouvelle commande instantanément.

```
1 reference
private void btnLivresCommandesAjouter_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txbCommandesLivresNumero.Text)) return;

    var payload = new
    {
        dateCommande = dtpLivresCommandesDate.Value.ToString("yyyy-MM-dd"),
        montant = (double)nudLivresCommandesMontant.Value,
        nbExemplaire = (int)nudLivresCommandesExemplaires.Value,
        idLivreDvd = txbCommandesLivresNumero.Text
    };

    if (controller.EnregistrerCommande(payload))
    {
        ChargerCommandesLivres();
    }
    else
    {
        MessageBox.Show("Erreur lors de l'enregistrement.", "Erreur");
    }
}
```

Cette méthode permet la mise à jour de l'étape de suivi d'une commande. Elle transmet l'ordre de modification à l'api. Si la modification est bloquée par les Triggers SQL la méthode intercepte l'échec et affiche un message d'erreur à l'utilisateur.

```
1 reference
private void btnLivresCommandesModifier_Click(object sender, EventArgs e)
{
    if (dgvLivresCommandes.CurrentCell == null) return;
    CommandeDocument cmd = (CommandeDocument)bdgCommandesLivres.List[bdgCommandesLivres.Position];
    string nouveauSuivi = cbxLivresCommandesSuivi.SelectedValue.ToString();

    if (controller.ModifierSuiviCommande(cmd.Id, nouveauSuivi))
    {
        ChargerCommandesLivres();
    }
    else
    {
        MessageBox.Show("Action refusée. Vérifiez les règles métier (ex: impossible de revenir en arrière ou de régler avant livraison).", "Erreur");
    }
}
```

Avant suppression cette méthode affiche une alerte demandant confirmation à l'utilisateur pour éviter les erreurs de manipulation.

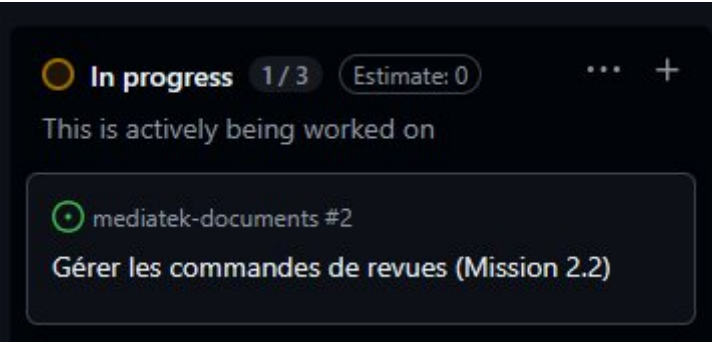
```
1 reference
private void btnLivresCommandesSupprimer_Click(object sender, EventArgs e)
{
    if (dgvLivresCommandes.CurrentCell == null) return;
    CommandeDocument cmd = (CommandeDocument)bdgCommandesLivres.List[bdgCommandesLivres.Position];

    if (MessageBox.Show("Voulez-vous vraiment supprimer cette commande ?", "Confirmation", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        if (controller.SupprimerCommande(cmd.Id))
        {
            ChargerCommandesLivres();
        }
        else
        {
            MessageBox.Show("Suppression refusée. Une commande livrée ou réglée ne peut être supprimée.", "Erreur");
        }
    }
}
#endregion
```

La logique pour l'onglet dvd est la même.

Temps estimé	Temps réel
8h	10-11h

Gérer les commandes de revues (Mission 2.2)



Modification dans l'api

Dans MyAccessBDD.php

Ajout des case pour abonnements_revue et expirants dans traitementSelect

```
case "abonnements_revue":  
    return $this->getAbonnementsRevue($champs);  
case "abonnements_expirants":  
    return $this->getAbonnementsExpirants();
```

Et également dans traitementInsert la route pour l'insertion d'un abonnement. Elle créer un id unique et insère la ligne dans la table commande puis insère les détails dans la table abonnement. Si la seconde requête échoue le code supprime la ligne créer dans commande pour éviter qu'il y ait des données orpheline dans la base.

```
case "gestion_abonnement":  
    $id = $this->creerIdAuto();  
    $res1 = $this->insertOneTupleOneTable("commande", [  
        "id" => $id,  
        "dateCommande" => $champs["dateCommande"],  
        "montant" => $champs["montant"]  
    ]);  
  
    if ($res1 !== null) {  
        $res2 = $this->insertOneTupleOneTable("abonnement", [  
            "id" => $id,  
            "dateFinAbonnement" => $champs["dateFinAbonnement"],  
            "idRevue" => $champs["idRevue"]  
        ]);  
  
        if ($res2 !== null) {  
            return 1;  
        } else {  
            $this->deleteTuplesOneTable("commande", ["id" => $id]);  
            return null;  
        }  
    }  
    return null;  
  
default:  
    return $this->insertOneTupleOneTable($table, $champs);  
}
```


Ainsi que dans `traitementDelete`. Cette méthode gère la suppression d'un abonnement. Le script supprime obligatoirement dans la table fille abonnement avant de supprimer l'enregistrement correspondant dans la table mère commande .

```
protected function traitementDelete(string $table, ?array $champs) : ?int{
    switch($table){
        case "" :
            // return $this->uneFonction(parametres);
        case "gestion_abonnement":
            $res1 = $this->deleteTuplesOneTable("abonnement", ["id" => $champs["id"]]);
            if ($res1 !== null) {
                $this->deleteTuplesOneTable("commande", ["id" => $champs["id"]]);
                return 1;
            }
            return null;
        default:
            // cas général
            return $this->deleteTuplesOneTable($table, $champs);
    }
}
```

Ajouts de deux nouvelles fonctions pour récupérer l'historique des abonnements d'une revue dans la BDD. La fonction `getAbonnementsExpirants` vérifie si une commande expire dans les 30 jours et servira pour l'alerte dans l'app.

```
/**
 * Récupère l'historique des abonnements d'une revue
 * @param array|null $champs
 * @return array|null
 */
private function getAbonnementsRevue(?array $champs) : ?array {
    if (empty($champs) || !array_key_exists('id', $champs)) return null;
    $sql = "SELECT a.id, c.dateCommande, c.montant, a.dateFinAbonnement, a.idRevue ";
    $sql .= "FROM abonnement a JOIN commande c ON c.id = a.id ";
    $sql .= "WHERE a.idRevue = :id ORDER BY c.dateCommande DESC";
    return $this->conn->queryBDD($sql, ['id' => $champs['id']]);
}

/**
 * Récupère les abonnements expirant dans moins de 30 jours
 * @return array|null
 */
private function getAbonnementsExpirants() : ?array {
    $sql = "SELECT a.id, c.dateCommande, c.montant, a.dateFinAbonnement, a.idRevue, d.titre as Titre ";
    $sql .= "FROM abonnement a JOIN commande c ON c.id = a.id ";
    $sql .= "JOIN document d ON a.idRevue = d.id ";
    $sql .= "WHERE DATEDIFF(a.dateFinAbonnement, CURDATE()) BETWEEN 0 AND 30 ";
    $sql .= "ORDER BY a.dateFinAbonnement ASC";
    return $this->conn->queryBDD($sql);
}
```

Modifications dans le contrôleur, model et dal

Création de la classe Abonnement qui hérite de la classe Commande dans model. Elle a les propriétés DateFinAbonnement, IdRevue et Titre qui permet de récupérer le nom de la revue pour faciliter l'affichage de l'alerte.

```
namespace MediaTekDocuments.model
{
    1 reference
    public class Abonnement : Commande
    {
        [JsonProperty("dateFinAbonnement")]
        1 reference
        public DateTime DateFinAbonnement { get; set; }

        [JsonProperty("idRevue")]
        1 reference
        public string IdRevue { get; set; }

        // Ajout d'une propriété Titre . Sert uniquement pour le fenêtre d'alerte des 30 jours
        [JsonProperty("Titre")]
        1 reference
        public string Titre { get; set; }

        0 references
        public Abonnement(string id, DateTime dateCommande, double montant, DateTime dateFinAbonnement, string idRevue, string titre = "")
            : base(id, dateCommande, montant)
        {
            this.DateFinAbonnement = dateFinAbonnement;
            this.IdRevue = idRevue;
            this.Titre = titre;
        }
    }
}
```

Dans Aaccess.cs

```
/// <summary>
/// Récupère l'historique des abonnements d'une revue
/// </summary>
0 references
public List<Abonnement> GetAbonnements(string idDoc)
{
    string jsonId = convertToJson("id", idDoc);
    return TraitementRecup<Abonnement>(GET, "abonnements_revue/" + jsonId, null);
}

/// <summary>
/// Récupère les abonnements expirant dans moins de 30 jours
/// </summary>
0 references
public List<Abonnement> GetAbonnementsExpirants()
{
    return TraitementRecup<Abonnement>(GET, "abonnements_expirants", null);
}

/// <summary>
/// Enregistre un nouvel abonnement
/// </summary>
0 references
public bool EnregistrerAbonnement(object data)
{
    string json = JsonConvert.SerializeObject(data, new CustomDateTimeConverter());
    return TraitementRecup<Abonnement>(POST, "gestion_abonnement", "champs=" + json) != null;
}

/// <summary>
/// Supprime un abonnement
/// </summary>
0 references
public bool SupprimerAbonnement(string idCommande)
{
    string jsonIdCommande = convertToJson("id", idCommande);
    return TraitementRecup<Abonnement>(DELETE, "gestion_abonnement/" + jsonIdCommande, null) != null;
}
```


Dans FrmMediatekController création de la méthode ParutionDansAbonnement qui retourne un boolean vérifiant si une date de parution physique se trouve dans l'intervalle de l'abonnement.

```
0 references
public List<Abonnement> GetAbonnements(string idDoc) => access.GetAbonnements(idDoc);
0 references
public List<Abonnement> GetAbonnementsExpirants() => access.GetAbonnementsExpirants();
0 references
public bool EnregistrerAbonnement(object data) => access.EnregistrerAbonnement(data);
0 references
public bool SupprimerAbonnement(string idCommande) => access.SupprimerAbonnement(idCommande);

/// <summary>
/// Méthode pour vérifier si une parution est dans l'abonnement
/// </summary>
0 references
public bool ParutionDansAbonnement(DateTime dateCommande, DateTime dateFinAbonnement, DateTime dateParution)
{
    return (dateParution >= dateCommande && dateParution <= dateFinAbonnement);
}
```

Modification de la vue

Création de l'interface de commande des revues et de la logique pour les boutons.

Gestion des documents de la médiathèque

Livres DVD Revue Parutions des revues Commandes Livres Commandes Dvd Commandes Revues

Rechercher label60

GroupBox3

Numéro Commande: Ajouter

Date: dimanche 22 mars 2026 Supprimer

Montant: 0.00

Fin d'abonnement: dimanche 22 mars 2026

Création ensuite d'une région avec la logique de chaque bouton.

A la saisie d'un numéro de revue cette méthode vérifie l'existence du document dans le catalogue. Si elle existe elle met un titre et appelle ChargerCommandeRevue

```
#region Onglet Commandes Revues

private readonly BindingSource bdgCommandesRevue = new BindingSource();

// Recherche d'une revue et chargement de ses abonnements
0 references
private void btnCommandesRevueRechercher_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(txbCommandesRevueNumero.Text))
    {
        Revue revue = controller.GetAllRevue().Find(x => x.Id == txbCommandesRevueNumero.Text);
        if (revue != null)
        {
            lblCommandesRevueTitre.Text = revue.Titre;
            ChargerCommandesRevue();
        }
        else
        {
            MessageBox.Show("Revue introuvable.", "Information");
            lblCommandesRevueTitre.Text = "";
            bdgCommandesRevue.DataSource = null;
        }
    }
}
```

Sert à rafraîchir le tableau. Cache les colonnes non nécessaires et renomme les autres pour qu'elle soit plus compréhensible pour l'utilisateur .

```
// Rafraîchir le tableau des abonnements
3 references
private void ChargerCommandesRevue()
{
    string idDoc = txbCommandesRevueNumero.Text;
    List<Abonnement> abonnements = controller.GetAbonnements(idDoc);

    dgvRevueCommandes.DataSource = null;
    bdgCommandesRevue.DataSource = abonnements;
    dgvRevueCommandes.DataSource = bdgCommandesRevue;

    if (dgvRevueCommandes.Columns.Count > 0)
    {
        dgvRevueCommandes.Columns["IdRevue"].Visible = false;
        dgvRevueCommandes.Columns["Titre"].Visible = false;

        dgvRevueCommandes.Columns["Id"].HeaderText = "Numéro";
        dgvRevueCommandes.Columns["DateCommande"].HeaderText = "Date Commande";
        dgvRevueCommandes.Columns["Montant"].HeaderText = "Montant";
        dgvRevueCommandes.Columns["DateFinAbonnement"].HeaderText = "Date Fin Abonnement";

        dgvRevueCommandes.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    }
}
```

Lors d'un click sur une ligne du tableau récupération de l'objet Abonnement et envoi vers les autres composants visuels de l'interface. Vérification pour les DateTimePicker que la date de fin de la BDD ne dépasse pas les limites acceptées par le composant du Windows Forms

```
// Clic sur une ligne du tableau
1 reference
private void dgvRevuesCommandes_SelectionChanged(object sender, EventArgs e)
{
    if (dgvRevuesCommandes.CurrentCell != null && bdgCommandesRevues.List.Count > 0)
    {
        Abonnement abo = (Abonnement)bdgCommandesRevues.List[bdgCommandesRevues.Position];
        txbRevuesCommandesNumeroCmd.Text = abo.Id;
        dtpRevuesCommandesDate.Value = abo.DateCommande;
        nudRevuesCommandesMontant.Value = (decimal)abo.Montant;

        if (abo.DateFinAbonnement >= dtpRevuesCommandesDateFin.MinDate && abo.DateFinAbonnement <= dtpRevuesCommandesDateFin.MaxDate)
            dtpRevuesCommandesDateFin.Value = abo.DateFinAbonnement;
    }
    else
    {
        txbRevuesCommandesNumeroCmd.Text = "";
        nudRevuesCommandesMontant.Value = 0;
        dtpRevuesCommandesDate.Value = DateTime.Now;
        dtpRevuesCommandesDateFin.Value = DateTime.Now.AddMonths(1);
    }
}
```

Gère l'ajout d'un nouvel abonnement. Avant d'envoyer la requête réseau à l'api vérification que la date de fin d'abonnement est strictement postérieure à la date de commande sinon bloque.

```
// Ajouter un abonnement
0 references
private void btnRevuesCommandesAjouter_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txbCommandesRevuesNumero.Text)) return;

    if (dtpRevuesCommandesDateFin.Value.Date <= dtpRevuesCommandesDate.Value.Date)
    {
        MessageBox.Show("La date de fin d'abonnement doit être postérieure à la date de commande.", "Erreur Saisie");
        return;
    }

    var payload = new
    {
        dateCommande = dtpRevuesCommandesDate.Value.ToString("yyyy-MM-dd"),
        montant = (double)nudRevuesCommandesMontant.Value,
        dateFinAbonnement = dtpRevuesCommandesDateFin.Value.ToString("yyyy-MM-dd"),
        idRevue = txbCommandesRevuesNumero.Text
    };

    if (controller.EnregistrerAbonnement(payload))
    {
        ChargerCommandesRevues();
    }
    else
    {
        MessageBox.Show("Erreur lors de l'enregistrement de l'abonnement.", "Erreur API");
    }
}
```

Pour la logique de suppression on ne peut pas supprimer un abonnement si des numéros de la revue ont déjà été réceptionné pendant cette période . Lors du click ça récupère tous les exemplaires physique liés à la revue concerné. Ensuite boucle foreach sur ces exemplaires en appelant la méthode ParutionsDansAbonnement pour chaque exemplaire. Si un exemplaire correspond à l'intervalle l'action est bloquée et une MessageBox s'affiche.

```
// Supprimer un abonnement (Avec sécurité Parution)
0 references
private void btnRevuesCommandesSupprimer_Click(object sender, EventArgs e)
{
    if (dgvRevuesCommandes.CurrentCell == null) return;
    Abonnement abo = (Abonnement)bdgCommandesRevues.List[bdgCommandesRevues.Position];

    // Vérifier si des numéros de cette revue ont été publiés pendant l'abonnement
    List<Exemplaire> exemplairesRevue = controller.GetExemplairesRevue(abo.IdRevue);
    bool suppressionBloquee = false;

    foreach (Exemplaire exemplaire in exemplairesRevue)
    {
        if (controller.ParutionDansAbonnement(abo.DateCommande, abo.DateFinAbonnement, exemplaire.DateAchat))
        {
            suppressionBloquee = true;
            break;
        }
    }

    if (suppressionBloquee)
    {
        MessageBox.Show("Suppression refusée : Des exemplaires (parutions) sont rattachés à cet abonnement.", "Sécurité Métier");
        return;
    }

    if (MessageBox.Show("Voulez-vous vraiment supprimer cet abonnement ?", "Confirmation", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        if (controller.SupprimerAbonnement(abo.Id))
        {
            ChargerCommandesRevues();
        }
        else
        {
            MessageBox.Show("Erreur lors de la suppression.", "Erreur API");
        }
    }
}
#endregion
```

Enfin, pour la fonctionnalité d'alerte pour les abonnements qui expirent dans moins de 30 jours. FrmMediatek_Load est lié à la fenêtre principale de l'application est se lance dont au chargement. Elle interroge la route de l'API des expirations imminente et si la liste n'est pas vide affiche un message listant les titres et les dates d'expirations.

```
/// <summary>
/// Événement déclenché au chargement de l'application
/// Affiche l'alerte pour les abonnements qui expirent dans moins de 30 jours
/// </summary>
0 references
private void FrmMediatek_Load(object sender, EventArgs e)
{
    List<Abonnement> abosExpirants = controller.GetAbonnementsExpirants();
    if (abosExpirants != null && abosExpirants.Count > 0)
    {
        string messageAlerte = "Attention, les abonnements suivants arrivent à expiration dans moins de 30 jours :\n\n";

        foreach (Abonnement abo in abosExpirants)
        {
            messageAlerte += $"- {abo.Titre} (Fin le : {abo.DateFinAbonnement.ToString("dd/MM/yyyy")})\n";
        }

        MessageBox.Show(messageAlerte, "Alerte Fin d'Abonnements", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Exemple d'un ajout de commande.

Gestion des documents de la médiathèque

Livres DVD Revues Parutions des revues Commandes Livres Commandes Dvd Commandes Revues

10001 Rechercher Arts Magazine

	Date Fin Abonnement	Numéro	Date Commande	Montant
▶	10/07/2027	00003	22/03/2026	1

groupBox3

Numéro Commande : 00003 Ajouter
 Date : dimanche 22 mars 2026 Supprimer
 Montant : 1.00
 Fin d'abonnement : samedi 10 juillet 2026

Exemple de la sécurité lors de la suppression d'un abonnement avec exemplaires lié.

Gestion des documents de la médiathèque

Livres DVD Revues Parutions des revues Commandes Livres Commandes Dvd Commandes Revues

10011 Rechercher Geo

	Date Fin Abonnement	Numéro	Date Commande	Montant
▶	31/12/2022	00004	01/01/2021	45

groupBox3

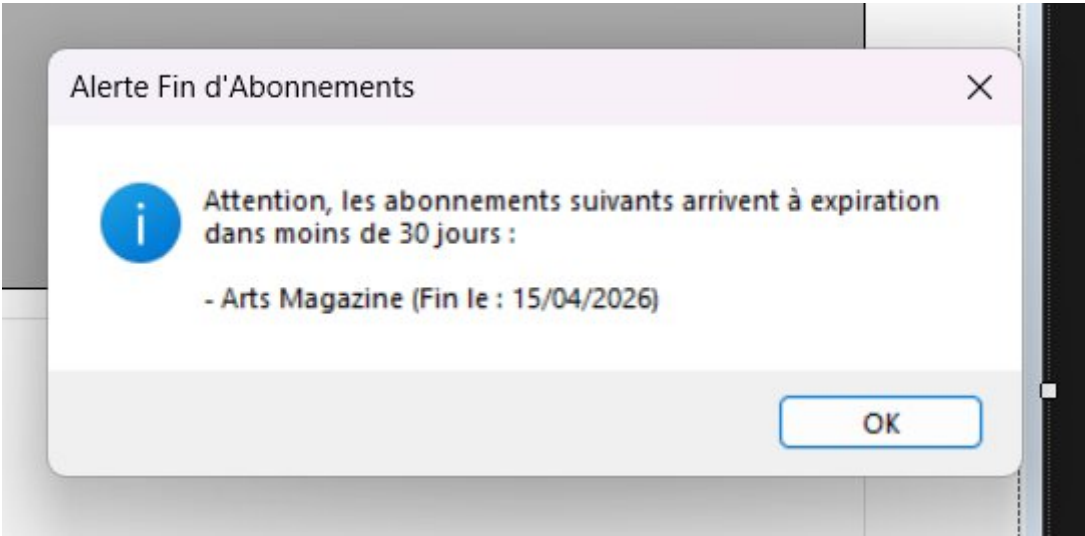
Numéro Commande : 00004
 Date : 01/01/2021
 Montant : 45
 Fin d'abonnement : samedi 31 décembre 2022

Sécurité Métier

Suppression refusée : Des exemplaires (parutions) sont rattachés à cet abonnement.

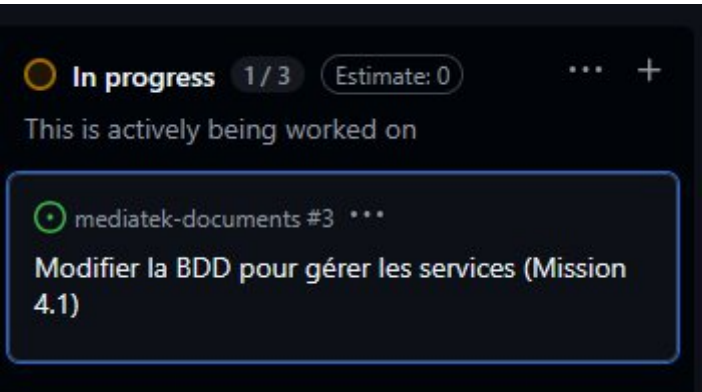
OK

Exemple du message d'alerte sur les abonnements qui expirent bientôt au lancement de l'application.



Temps estimé :	Temps réel
4h	4h

Mission 4 : mettre en place des authentifications



Ajout du SQL.

Creation des tables requises.


```

-- Création de la table des Services
CREATE TABLE IF NOT EXISTS service (
    id varchar(5) NOT NULL,
    libelle varchar(50) NOT NULL,
    PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Les 4 services
INSERT INTO service (id, libelle) VALUES
('00001', 'Administrateur'),
('00002', 'Administratif'),
('00003', 'Prêts'),
('00004', 'Culture');

-- Création de la table Utilisateur
CREATE TABLE IF NOT EXISTS utilisateur (
    id varchar(5) NOT NULL,
    login varchar(50) NOT NULL,
    password varchar(255) NOT NULL,
    idService varchar(5) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY uk_utilisateur_login (login),
    CONSTRAINT fk_utilisateur_service FOREIGN KEY (idService) REFERENCES service (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Insertion d'un utilisateur de test pour chaque service avec les mdp en clairs
INSERT INTO utilisateur (id, login, password, idService) VALUES
('U0001', 'admin', 'admin', '00001'),
('U0002', 'bureau', 'bureau123', '00002'),
('U0003', 'pret', 'pret123', '00003'),
('U0004', 'culture', 'culture123', '00004');

```



Dans MyAccessBDD.php ajout du case dans traitementSelect pour l'authentification

```

case "authentification":
    return $this->getAuthentification($champs);

```

Creation de la methode getAuthentification qui fait une jointure entre utilisateur et service.

```
/**
 * Vérifie les identifiants d'un utilisateur
 * Retourne les informations de l'utilisateur et son service si ok, sinon null
 * @param array|null $champs
 * @return array|null
 */
private function getAuthentification(?array $champs) : ?array {
    if (empty($champs) || !array_key_exists('login', $champs) || !array_key_exists('password', $champs)) {
        return null;
    }
    // Récupération de l'utilisateur et du libellé de son service grâce à une jointure
    $sql = "SELECT u.id, u.login, u.idService, s.libelle as service ";
    $sql .= "FROM utilisateur u JOIN service s ON u.idService = s.id ";
    $sql .= "WHERE u.login = :login AND u.password = :password";

    return $this->conn->queryBDD($sql, ['login' => $champs['login'], 'password' => $champs['password']]);
}
```

Ajout en C#

Creation des classes dans model Service et Utilisateurs

Utilisation de JsonProperty de Newtonsoft pour la conversion.

```
namespace MediaTekDocuments.model
{
    1 reference
    public class Service
    {
        [JsonProperty("id")]
        1 reference
        public string Id { get; set; }

        [JsonProperty("libelle")]
        1 reference
        public string Libelle { get; set; }

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

```

namespace MediaTekDocuments.model
{
    1 reference
    public class Utilisateur
    {
        [JsonProperty("id")]
        1 reference
        public string Id { get; set; }

        [JsonProperty("login")]
        1 reference
        public string Login { get; set; }

        [JsonProperty("idService")]
        1 reference
        public string IdService { get; set; }
        [JsonProperty("service")]
        1 reference
        public string LeService { get; set; }

        0 references
        public Utilisateur(string id, string login, string idService, string leService)
        {
            this.Id = id;
            this.Login = login;
            this.IdService = idService;
            this.LeService = leService;
        }
    }
}

```

Dans Access.cs Authentification fait le pont avec l'api et structure les données avec le dictionnaire.

```

public Utilisateur Authentification(string login, string password)
{
    // Creation d'un dictionnaire pour envoyer les deux paramètres dans le JSON
    var param = new Dictionary<string, string>
    {
        { "login", login },
        { "password", password }
    };

    string jsonParam = JsonConvert.SerializeObject(param);

    List<Utilisateur> liste = TraitementRecup<Utilisateur>(GET, "authentification/" + jsonParam, null);

    // S'il y a un résultat, c'est que les identifiants sont bons
    if (liste != null && liste.Count > 0)
    {
        return liste[0];
    }
    return null;
}
}

```

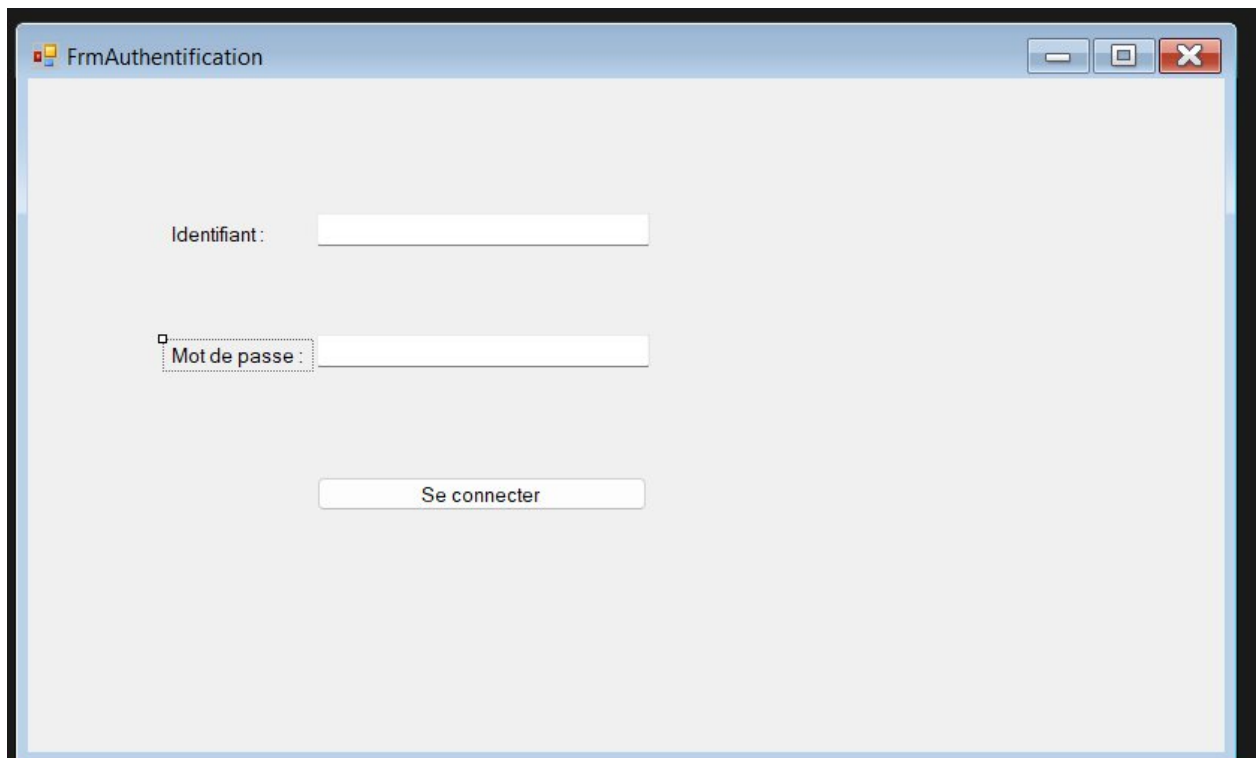
Et ajout du relais dans le Controller.

```
0 references
public Utilisateur Authentification(string login, string password)
{
    return access.Authentification(login, password);
}
```

Vu que l'application doit s'ouvrir sur la page de connexion il faut créer un windows form et dire dans Program.cs que l'on veut que l'application s'ouvre dessus.

```
Application.Run(new FrmAuthentification());
```

Création de l'interface du windows form pour la connexion.



The screenshot shows a Windows Form titled "FrmAuthentification". The form has a light gray background and a blue title bar. It contains two text input fields: the first is labeled "Identifiant:" and the second is labeled "Mot de passe:". Below the password field, there is a button labeled "Se connecter". The form is displayed in a standard Windows window with minimize, maximize, and close buttons in the top right corner.

Le bouton se connecter vérifie si les champs sont remplis, bloque la connexion si l'utilisateur fait parti du service culture.

```
1 reference
private void btnSeConnecter_Click(object sender, EventArgs e)
{
    string login = txtLogin.Text;
    string password = txtPassword.Text;

    if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))
    {
        MessageBox.Show("Veuillez saisir tous les champs.", "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    Utilisateur utilisateur = controller.Authentification(login, password);

    if (utilisateur != null)
    {
        //Le service Culture n'a pas accès à l'application
        if (utilisateur.LeService == "Culture")
        {
            MessageBox.Show("Vos droits ne sont pas suffisants pour accéder à cette application.", "Accès refusé", MessageBoxButtons.OK, MessageBoxIcon.Error);
            Application.Exit();
        }
        else
        {
            // Si c'est bon, ouverture de la fenêtre principale en lui passant le nom du service
            FrmMediatek frmMediatek = new FrmMediatek(utilisateur.LeService);
            frmMediatek.Show();
            this.Hide();
        }
    }
    else
    {
        MessageBox.Show("Identifiant ou mot de passe incorrect.", "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Pour le masquage des onglets pour les roles non admin modification du constructeur FrmMediatek pour qu'il cache les onglets TabPages auquel ils ne doivent pas avoir accès

```
1 reference
public FrmMediatek(string service)
{
    InitializeComponent();
    this.controller = new FrmMediatekController();

    if (service == "Prêts")
    {
        // Cherche le composant tabControl
        foreach (Control control in this.Controls)
        {
            if (control is TabControl tabControl)
            {
                // Supprime les onglets en partant de la fin pour ne pas faire planter la boucle
                for (int i = tabControl.TabPages.Count - 1; i >= 0; i--)
                {
                    string titreOnglet = tabControl.TabPages[i].Text;

                    // Cache si conteint un de ces mots.
                    if (titreOnglet.Contains("Commandes") ||
                        titreOnglet.Contains("Parutions") ||
                        titreOnglet.Contains("Ajouter"))
                    {
                        tabControl.TabPages.RemoveAt(i);
                    }
                }
            }
        }
    }

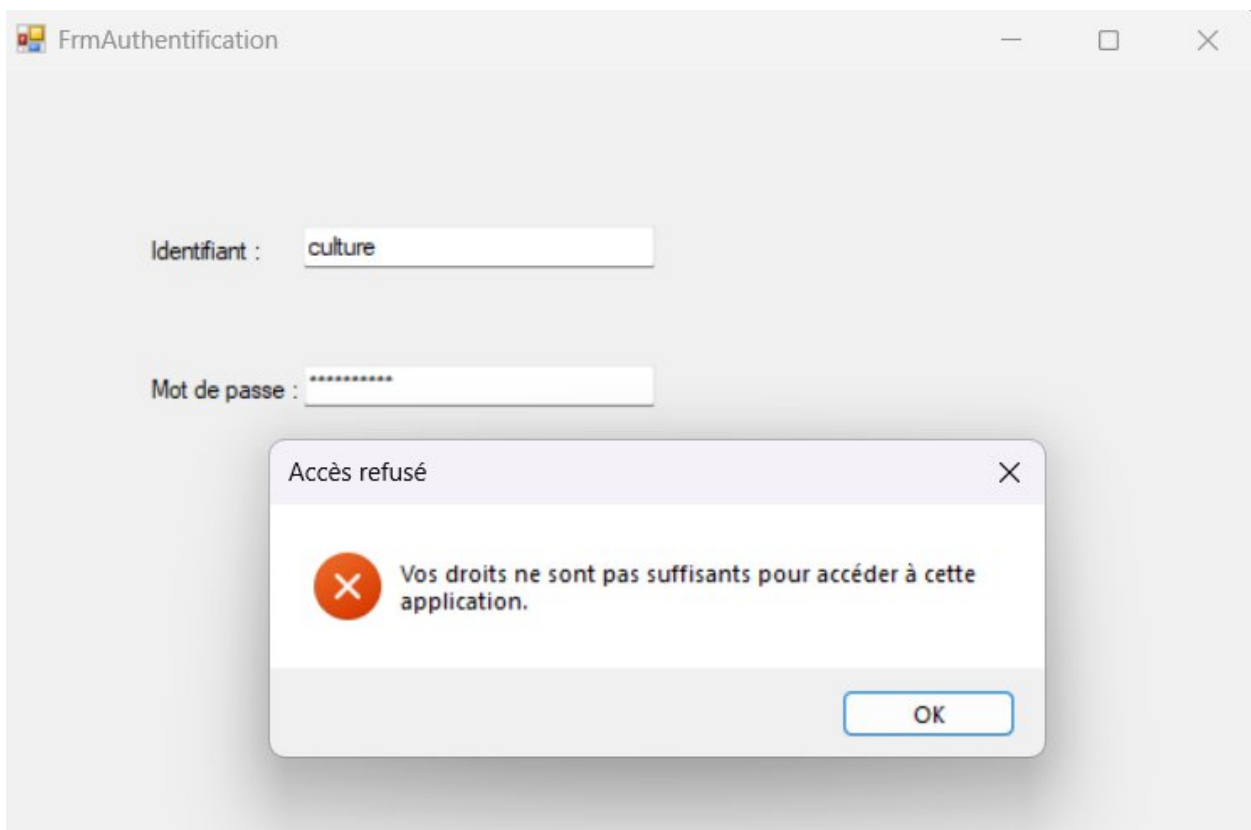
    if (service == "Administrateur" || service == "Administratif")
    {
        // Si admin uniquement
        AlerteFinAbonnements();
    }
}
```


La méthode qui affiche l'alerte des expirations à 30 jours à été déplacé dans le constructeur est n'est appelé que si l'utilisateur appartient aux services Admin.

```
/// <summary>
/// Affiche l'alerte pour les abonnements qui expirent dans moins de 30 jours
/// </summary>
1 reference
private void AlerteFinAbonnements()
{
    List<Abonnement> abosExpirants = controller.GetAbonnementsExpirants();

    if (abosExpirants != null && abosExpirants.Count > 0)
    {
        string messageAlerte = "Attention, les abonnements suivants arrivent à expiration dans moins de 30 jours :\n\n";
        foreach (Abonnement abo in abosExpirants)
        {
            messageAlerte += $"- {abo.Titre} (Fin le : {abo.DateFinAbonnement.ToString("dd/MM/yyyy")})\n";
        }
        MessageBox.Show(messageAlerte, "Alerte Fin d'Abonnements", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Connexion en tant que culture



Connexion en tant que pret (uniquement accès à Livres, DVD et Revues)

Gestion des documents de la médiathèque

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

Informations détaillées

Numéro de document : 00017 Code ISBN :

Titre : Catastrophes au Brésil

Auteur(e) : Philippe Masson


Collection :

Genre : Policier

Public : Ados

Rayon : Jeunesse romans

Chemin de l'image :

Image : 

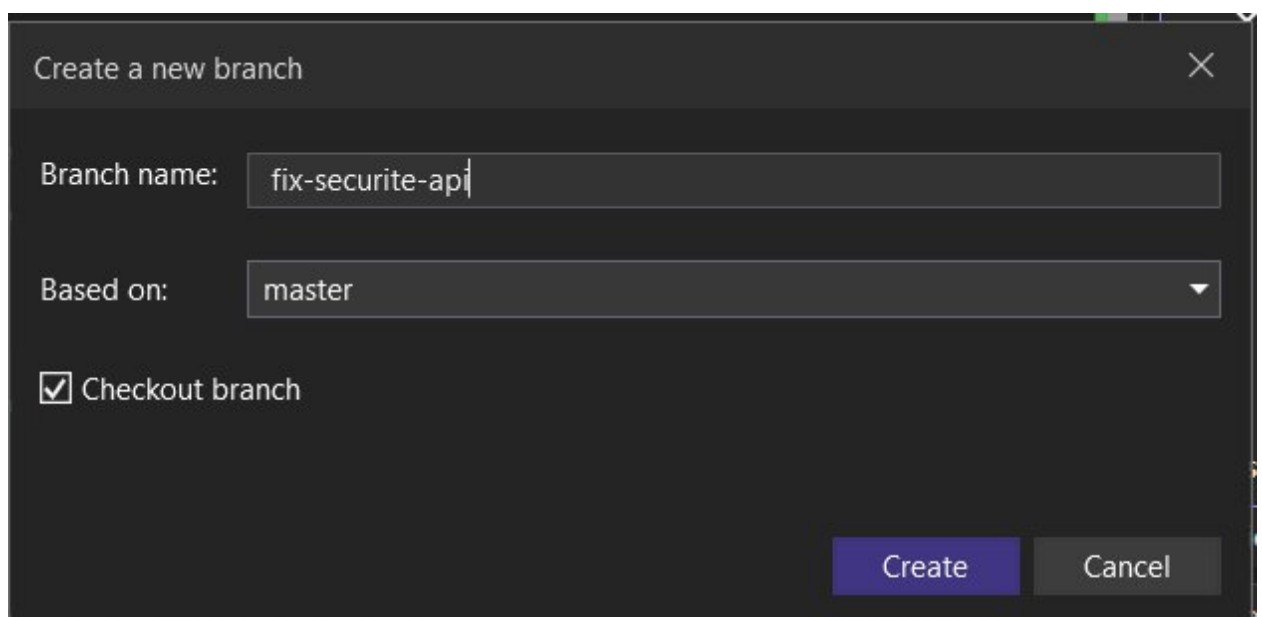
Et si on se connecte en tant qu'Admin on a accès a tout ainsi que le message d'alerte de fin d'abonnements.

Temps estimé :	Temps réel :
4 heures	5 heures

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



Creation d'une nouvelle branche nommé fix-securite-api

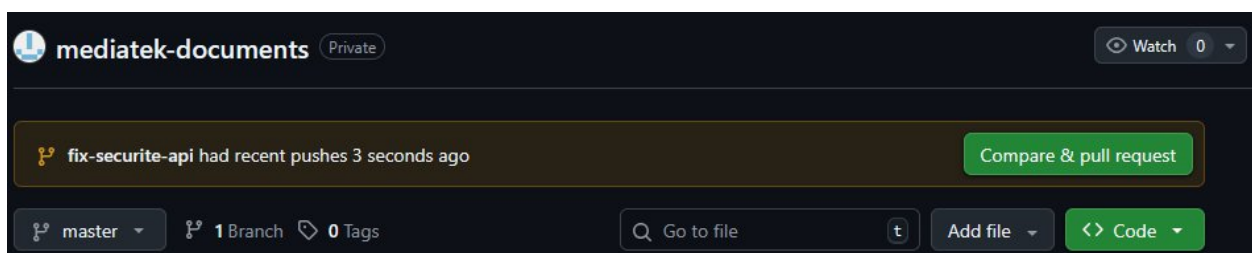


Déplacement de dal/Acess.cs vers le fichier App.config depuis où il sera appelé.

```
<startup>
  <appSettings>
    <add key="apiAuthenticationString" value="admin:adminpwd" />
  </appSettings>
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
</startup>
```

```
private Access()
{
    String authenticationString;
    try
    {
        authenticationString = ConfigurationManager.AppSettings["apiAuthenticationString"];
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
}
```

Merge de la branche avec master



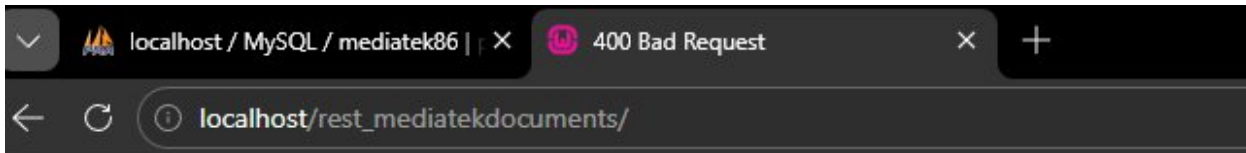
Pour bloquer l'accès non autorisé à la racine de l'API je crée une branche fix-htaccess dans l'interface Netbeans.



Ajout de la règle dans .htaccess

```
RewriteRule ^$ - [R=400,L]
```

On à bien maintenant une erreur 400

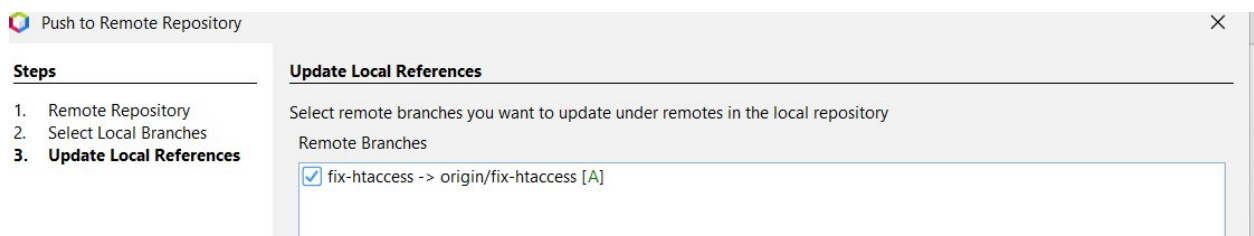


Bad Request

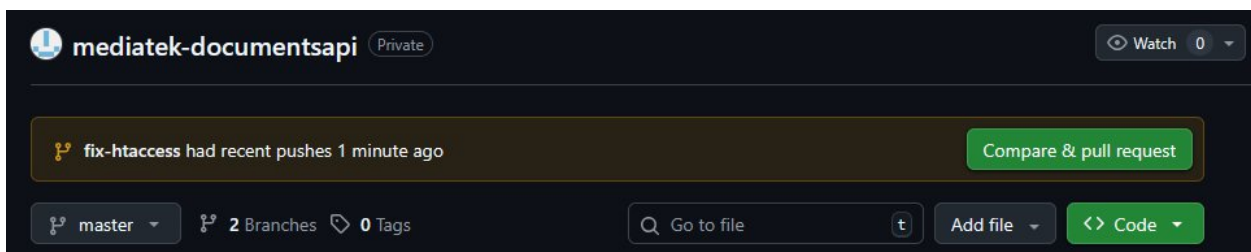
Your browser sent a request that this server could not understand.

Apache/2.4.65 (Win64) PHP/8.3.28 mod_fcgid/2.3.10-dev Server at localhost Port 80

Push sur la branche fix-htaccess



Et enfin merge avec la branche master



Tache 2 – Contrôler la qualité



Listes des erreurs Sonarlint

Error List						
Entire Solution		0 Errors	13 Warnings	0 of 62 Messages	Build + IntelliSense	Search Error List
	Code	Description	Project	File	Li...	Suppression State
		The referenced component 'Google.Protobuf' could not be found.	MediaTekDocuments			
		The referenced component 'System.Memory' could not be found.	MediaTekDocuments			
		The referenced component 'ZstdNet' could not be found.	MediaTekDocuments			
		The referenced component 'System.Net.Http.Formatting' could not be found.	MediaTekDocuments			
		The referenced component 'K4os.Compression.LZ4.Streams' could not be found.	MediaTekDocuments			
		The referenced component 'BouncyCastle.Crypto' could not be found.	MediaTekDocuments			
		The referenced component 'Ubiety.Dns.Core' could not be found.	MediaTekDocuments			
		The referenced component 'MySQL.Data' could not be found.	MediaTekDocuments			
		The element 'startup' has invalid child element 'appSettings'. List of possible elements expected: 'supportedRuntime, requiredRuntime'.	MediaTekDocuments	App.config	6	
	S3776	Refactor this constructor to reduce its Cognitive Complexity from 18 to the 15 allowed.	MediaTekDocuments	FrmMediatek.cs	27	Active
	S1186	Add a nested comment explaining why this method is empty, throw a 'NotSupportedException' or complete the implementation.	MediaTekDocuments	FrmMediatek.cs	1701	Active
	S1186	Add a nested comment explaining why this method is empty, throw a 'NotSupportedException' or complete the implementation.	MediaTekDocuments	FrmMediatek.cs	1709	Active
	S1186	Add a nested comment explaining why this method is empty, throw a 'NotSupportedException' or complete the implementation.	MediaTekDocuments	FrmMediatek.cs	1714	Active

Problème dans App.config. J'ai mis app setting à l'intérieur de startup.

Correction :

```
<startup>
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
</startup>

<appSettings>
  <add key="apiAuthenticationString" value="admin:adminpwd" />
</appSettings>
```

Les erreurs sont S1186 sont liée à des éléments de l'interface auquel j'ai créés des événements inutilisés il faut les délié et ensuite en supprimer le code.

Pour S3778 je l'ai refactorisé pour qu'elle soit moins longue elle utilise maintenant une fonction RestreindreAccesPrets qui est appelé dans FrmMediatek.

Avant :

```
1 reference
public FrmMediatek(string service)
{
    InitializeComponent();
    this.controller = new FrmMediatekController();

    1 if (service == "Prêts")
    {
        // Cherche le composant tabControl
        2 foreach (Control control in this.Controls)
        {
            3 if (control is TabControl tabControl)
            {
                // Supprime les onglets en partant de la fin pour ne pas faire planter la boucle
                4 for (int i = tabControl.TabPages.Count - 1; i >= 0; i--)
                {
                    string titreOnglet = tabControl.TabPages[i].Text;

                    // Cache si conteint un de ces mots.
                    5 if (titreOnglet.Contains("Commandes") 6 ||
                        titreOnglet.Contains("Parutions") ||
                        titreOnglet.Contains("Ajouter"))
                    {
                        tabControl.TabPages.RemoveAt(i);
                    }
                }
            }
        }

        7 if (service == "Administrateur" 8 || service == "Administratif")
        {
            // Si admin uniquement
            AlerteFinAbonnements();
        }
    }
}
```

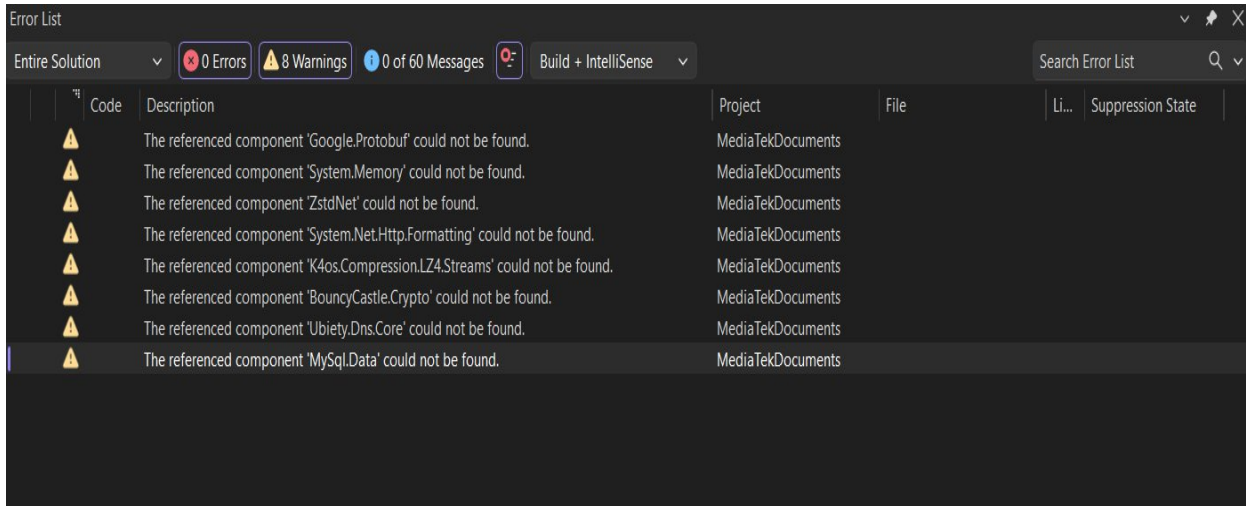
Après :

```
/// <summary>
/// Constructeur : création du contrôleur lié à ce formulaire
/// </summary>
1 reference
public FrmMediatek(string service)
{
    InitializeComponent();
    this.controller = new FrmMediatekController();

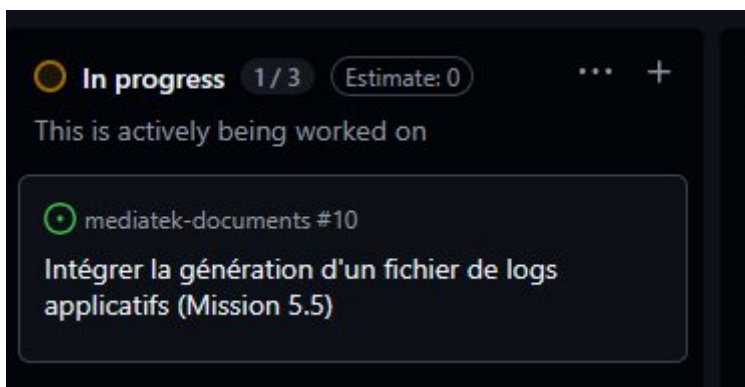
    if (service == "Prêts")
    {
        RestreindreAccesPrets();
    }

    if (service == "Administrateur" || service == "Administratif")
    {
        AlerteFinAbonnements();
    }
}

/// <summary>
/// Cache les onglets de gestion pour le service Prêts
/// </summary>
1 reference
private void RestreindreAccesPrets()
{
    foreach (Control control in this.Controls)
    {
        if (control is TabControl tabControl)
        {
            for (int i = tabControl.TabPages.Count - 1; i >= 0; i--)
            {
                string titreOnglet = tabControl.TabPages[i].Text;
                if (titreOnglet.Contains("Commandes") || titreOnglet.Contains("Parutions") || titreOnglet.Contains("Ajouter"))
                {
                    tabControl.TabPages.RemoveAt(i);
                }
            }
        }
    }
}
```

Tache 3 - Intégrer les logs



Pour les logs j'ai installer Serilog et Serilog.sinks.file avec les packages nuget.

Il faut maintenant configurer Access.cs pour logger au démarrage ce que l'ont fait dans l'application.

```
using Serilog;
```

Modification de private Access pour utiliser Serilog

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

    String authenticationString;
    try
    {
        authenticationString = ConfigurationManager.AppSettings["apiAuthenticationString"];
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
    catch (Exception e)
    {
        Log.Fatal(e, "Erreur fatale lors de l'initialisation de l'accès API");
        Environment.Exit(0);
    }
}
```

Dans traitement recup utilisation de Log.Error pour les catch de messages.

```
private List<T> TraitementRecup<T> (String methode, String message, String parametres)
{
    // trans
    List<T> liste = new List<T>();
    try
    {
        JObject retour = api.RecupDistant(methode, message, parametres);
        // extraction du code retourné
        String code = (String)retour["code"];
        if (code.Equals("200"))
        {
            // dans le cas du GET (select), récupération de la liste d'objets
            if (methode.Equals(GET))
            {
                String resultString = JsonConvert.SerializeObject(retour["result"]);
                // construction de la liste d'objets à partir du retour de l'api
                liste = JsonConvert.DeserializeObject<List<T>>(resultString, new CustomBooleanJsonConverter());
            }
        }
        else
        {
            Log.Error("Erreur API : méthode {methode}, message {message}. Code retour : {code}", methode, message, code);
        }
    }
    catch (Exception e)
    {
        Log.Error(e, "Erreur lors de l'accès à l'API ou du traitement des données");
        System.Windows.Forms.MessageBox.Show("Erreur API : " + e.Message, "Erreur Critique");
    }
    return liste;
}
```

Les erreurs sont maintenant loggées dans un fichier .txt situé dans /bin/Debug/logs

Temps estimé :	Temps réel :
4h	3-4h

Mission 6 – Tester et documenter

Backlog

4 / 5

Estimate: 0

...

+

This item hasn't been started

mediatek-documents #14

Écrire les tests unitaires du package Model (Mission 6.1)

mediatek-documents #15

Créer les tests Postman pour l'API (Mission 6.2)

mediatek-documents #16

Générer la documentation technique C# (Mission 6.3)

mediatek-documents #17

Générer la documentation technique API REST (Mission 6.4)

Test unitaires C#

Création des tests unitaires dans un projet MediatekDocumentsTests

```
namespace MediatekDocumentsTests
{
    [TestClass]
    0 references
    public class MediatekTests
    {
        /// <summary>
        /// Contrôler la méthode ToString() de la classe Categorie via un objet Genre.
        /// </summary>
        [TestMethod]
        0 references
        public void Categorie_ToString_ReturnsLibelle()
        {
            Genre genreHorreur = new Genre("10000", "Horreur");
            string resultat = genreHorreur.ToString();
            Assert.AreEqual("Horreur", resultat, "La méthode doit retourner le libellé exact.");
        }
        [TestMethod]
    }
}
```

```
[TestMethod]

/// <summary>
/// Vérifier que ParutionDansAbonnement retourne VRAI si la date est comprise dans l'abonnement.
/// </summary>
0 references
public void ParutionDansAbonnement_DateValide_ReturnsTrue()
{
    FrmMediatekController controller = new FrmMediatekController();
    DateTime dateCommande = new DateTime(2024, 01, 01, 0, 0, 0, DateTimeKind.Local);
    DateTime dateFinAbonnement = new DateTime(2024, 12, 31, 0, 0, 0, DateTimeKind.Local);
    DateTime dateParution = new DateTime(2024, 06, 15, 0, 0, 0, DateTimeKind.Local);

    bool resultat = controller.ParutionDansAbonnement(dateCommande, dateFinAbonnement, dateParution);

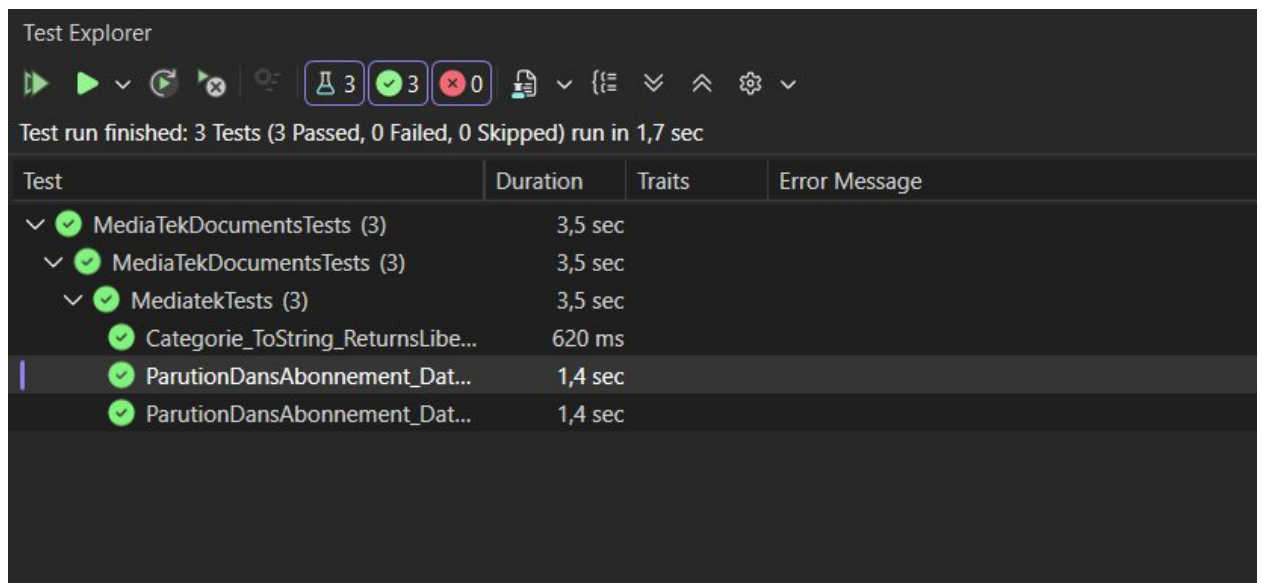
    Assert.IsTrue(resultat, "La parution devrait être valide car elle est comprise dans les dates de l'abonnement.");
}
```

```
/// <summary>
/// Vérifier que ParutionDansAbonnement retourne FAUX si la date de parution dépasse la date de fin d'abonnement.
/// </summary>
[TestMethod]
0 references
public void ParutionDansAbonnement_DateInvalide_ReturnsFalse()
{
    FrmMediatekController controller = new FrmMediatekController();
    DateTime dateCommande = new DateTime(2024, 01, 01, 0, 0, 0, DateTimeKind.Local);
    DateTime dateFinAbonnement = new DateTime(2024, 12, 31, 0, 0, 0, DateTimeKind.Local);
    DateTime dateParution = new DateTime(2025, 01, 15, 0, 0, 0, DateTimeKind.Local);

    bool resultat = controller.ParutionDansAbonnement(dateCommande, dateFinAbonnement, dateParution);

    Assert.IsFalse(resultat, "La parution devrait être invalide car elle dépasse la date de fin d'abonnement.");
}
```

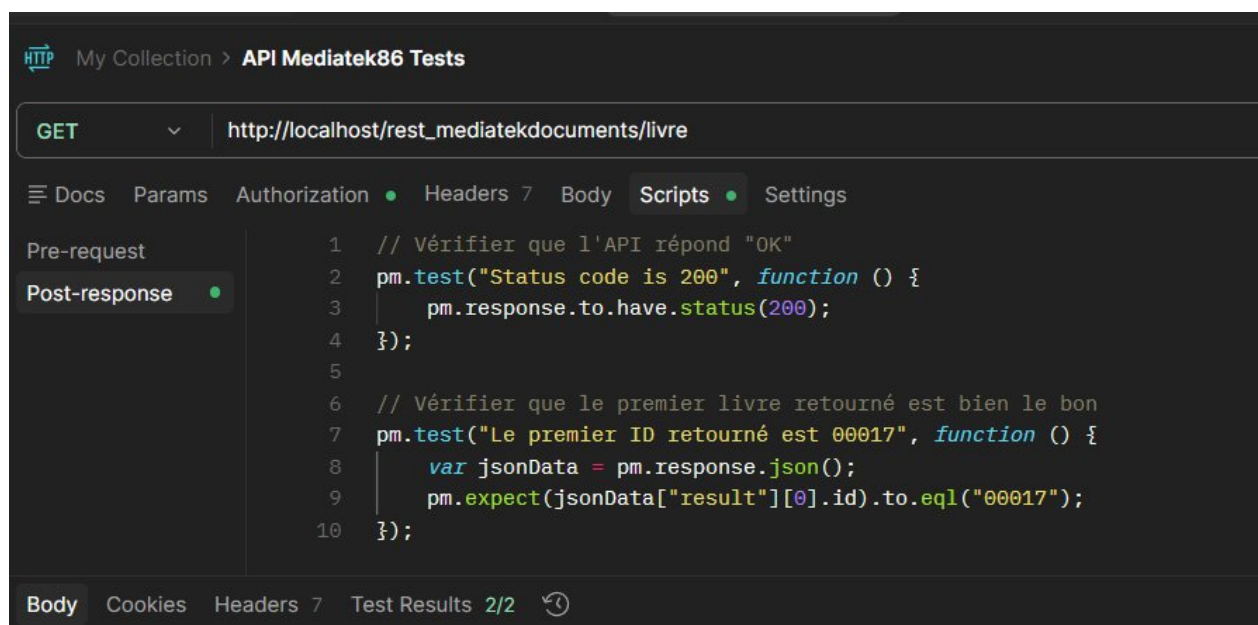
Tout les test sont validés dans l'explorateur de tests.



The screenshot shows the Test Explorer interface in VS Code. At the top, a status bar indicates 'Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 1,7 sec'. Below this is a table with columns: Test, Duration, Traits, and Error Message. The table lists several test suites and individual tests, all of which are marked with a green checkmark, indicating they passed.

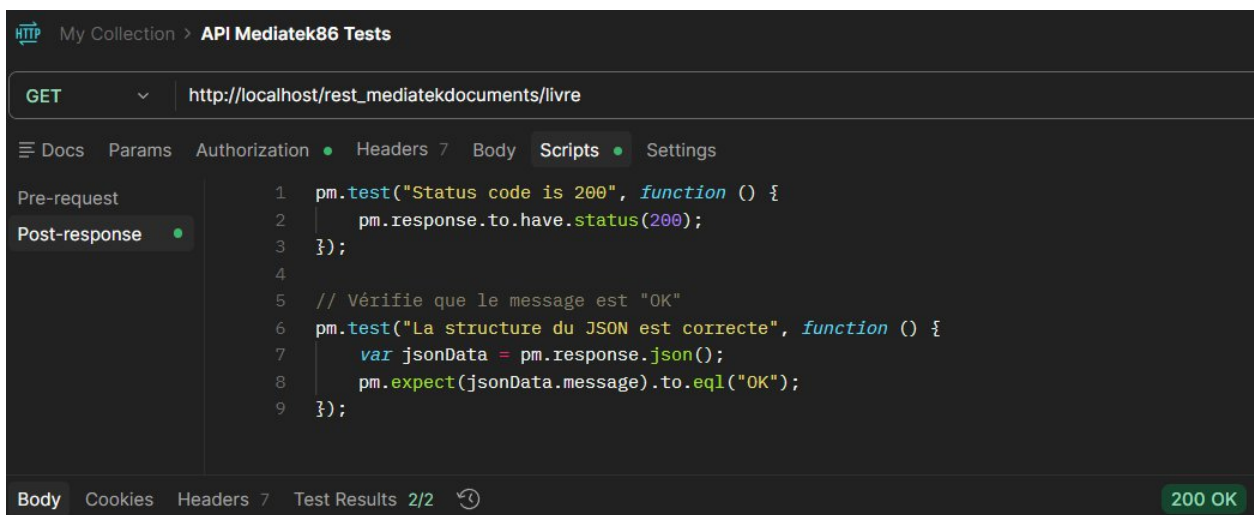
Test	Duration	Traits	Error Message
MediaTekDocumentsTests (3)	3,5 sec		
MediaTekDocumentsTests (3)	3,5 sec		
MediatekTests (3)	3,5 sec		
Categorie_ToString_ReturnsLibe...	620 ms		
ParutionDansAbonnement_Dat...	1,4 sec		
ParutionDansAbonnement_Dat...	1,4 sec		

Tests Postman API



This screenshot shows a Postman test script for the endpoint `http://localhost/rest_mediatekdocuments/livre`. The script is written in JavaScript and includes comments in French. It uses `pm.test` to verify the status code and the first book ID in the response.

```
1 // Vérifier que l'API répond "OK"
2 pm.test("Status code is 200", function () {
3   pm.response.to.have.status(200);
4 });
5
6 // Vérifier que le premier livre retourné est bien le bon
7 pm.test("Le premier ID retourné est 00017", function () {
8   var jsonData = pm.response.json();
9   pm.expect(jsonData["result"][0].id).to.eql("00017");
10  });
```



This screenshot shows the same Postman test script as above, but now with the test results. The status bar at the bottom right indicates a successful response: `200 OK`. The test results section shows that all tests passed.

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 // Vérifie que le message est "OK"
6 pm.test("La structure du JSON est correcte", function () {
7   var jsonData = pm.response.json();
8   pm.expect(jsonData.message).to.eql("OK");
9 });
```


Générer la documentation technique

La doc pour visual studio à été générée avec SandCastle et celle de netbeans avec phpDocumentor.

A Sandcastle Documented Class Library

☾ C# ▾ 🔍 Search

Unable to load TOC information:

This will not work if loaded from the file system directly.
Use the View Help option to view it using a local web server instance.

FrmMediatekController Class

Contrôleur lié à FrmMediatek

Definition

Namespace: [MediaTekDocuments.controller](#)
Assembly: [MediaTekDocuments](#) (in [MediaTekDocuments.exe](#)) Version: 1.0.0.0 (1.0.0.0)

C#

Copy

public class FrmMediatekController

Inheritance

Object → FrmMediatekController

Documentation

Search (Press "/" to focus)

Packages

Application

Reports

Deprecated

Errors

Markers

Indices

Files

Documentation

Table of Contents

Packages

Application

Classes

AccessBDD

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

Connexion

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

Controle

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

MyAccessBDD

Classe de construction des requêtes SQL hérite de AccessBDD qui contient les requêtes de base Pour ajouter une requête : - créer la fonction qui crée une requête (prendre modèle sur les fonctions existantes qui ne commencent pas par 'traitement') - ajouter un 'case' dans un des switch des fonctions redéfinies - appeler la nouvelle fonction dans ce 'case'

Url

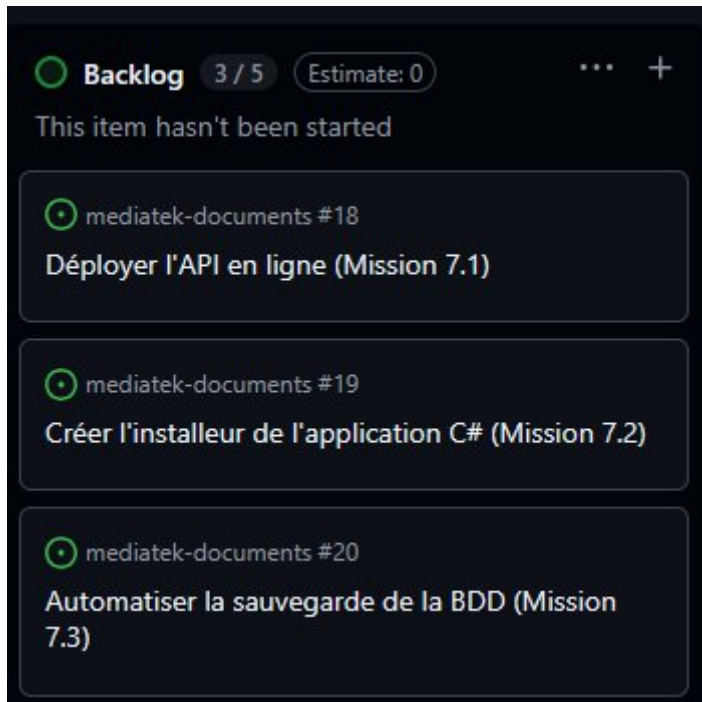
Singleton car la récupération des données ne peut se faire qu'une fois Permet de gérer le contenu de l'URL qui sollicite l'API

Création de la vidéo

Cette vidéo de présentation de 5min présentant le fonctionnement de l’application à été créer avec OBS Studio et est disponible sur le portfolio dans la page dédié à cet atelier.

Temps Estimé :	Temps Réel
8h	8-9h

Mission 7 – Déployer et gérer les sauvegardes de données



Déploiement de l'API en ligne

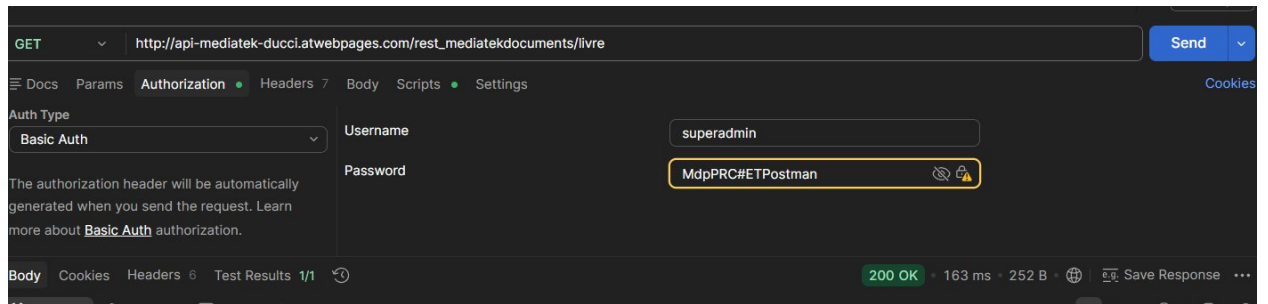
L'API est déployé sur awardspace.net sur le sous-domaine

api-mediatek-ducci.atwebpages.com

Le .env est modifié avec les paramètres de l'hébergeur

```
AUTHENTIFICATION=basic  
  
AUTH_USER=superadmin  
AUTH_PW=MdpPRCETPostman  
  
BDD_LOGIN=4745410_mediatekapi  
BDD_PWD=HFexCEr9R  
BDD_BD=4745410_mediatekapi  
BDD_SERVER=fdb1033.awardspace.net  
BDD_PORT=3306
```

l'API est atteignable depuis Postman.



Modification dans le C#

Dans App.config il faut renseigner le nouveau identifiant et mot de passe.

```
<appSettings>
  <add key="apiAuthenticationString" value="superadmin:MdpPRCETPostman" />
</appSettings>
```

Et modification de l'url de l'api dans Access.cs

```
/// <summary>
/// adresse de l'API
/// </summary>
private static readonly string uriApi = "http://api-mediatek-ducci.atwebpages.com/rest_mediatekdocuments/";
```

J'ai également ajouter dans le .htaccess pour laisser passer les headers.

```
RewriteCond %{HTTP:Authorization} ^(.+)$
RewriteRule ^(.*)$ - [E=HTTP_AUTHORIZATION:%1]
```

L'app à maintenant accès à l'API et la BDD en ligne.

Création de l'installateur

Depuis l'interface graphique click droit sur le projet et publish

Nom	Modifié le	Type	Taille
Application Files	24/03/2026 15:47	Dossier de fichiers	
MediaTekDocuments.application	24/03/2026 15:47	Application Manifest	6 Ko
setup.exe	24/03/2026 15:47	Application	537 Ko

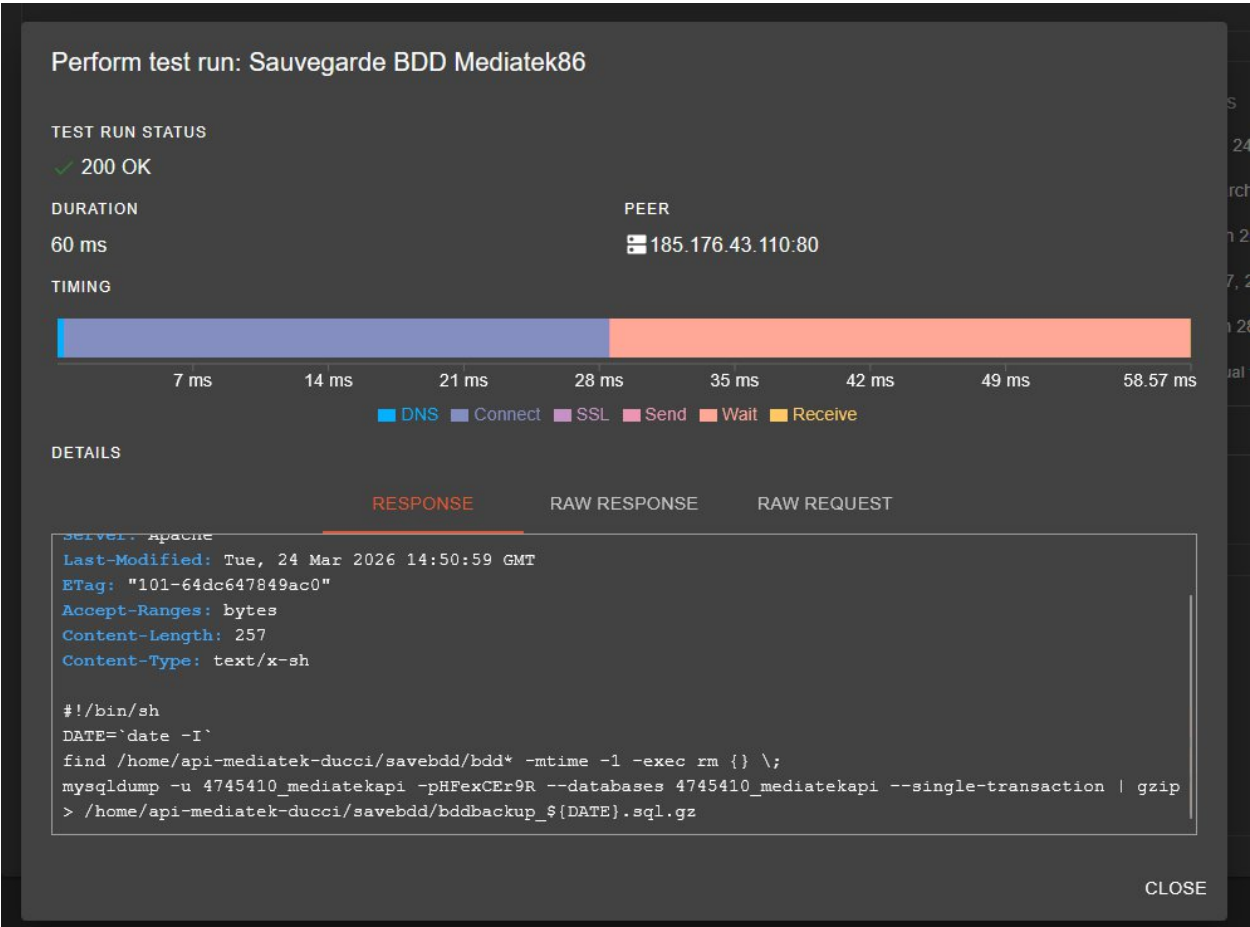
Automatiser la sauvegarde de la BDD

Creation d'un script de dump SQL de la BDD avec la commande mysqldump.

Ce script met une date au fichier de sauvegarde et supprime l'ancien backup.

```
#!/bin/sh
DATE=`date -I`
find /home/api-mediatek-ducci/savebdd/bdd* -mtime -1 -exec rm {} \;
mysqldump -u 4745410_mediatekapi -pHFexCER9R --databases 4745410_mediatekapi --single-transaction | gzip > /home/api-mediatek-ducci/savebdd/bddbbackup_${DATE}.sql.gz
```

Il est ensuite exécuté depuis cron-job.org tout les jours à 23h pour contourner le fait que l'hébergeur ne propose pas de CronJobs au comptes gratuits.



Temps estimé :	Temps réel :
4h	3-4h

Conclusion

Cette nouvelle version de MediatekDocuments est pleinement opérationnelle et les fonctionnalités du cahier des charges ont été implémenté avec succès.

Problèmes rencontrés :

Problème lors du déploiement à cause d'un caractère spécial mis dans un mdp qui bloquait les headers et me revoyait des erreurs « connexion non autorisé ». Réglé en changeants le mdp et ajoutant les règles dans le .htaccess.

Problème de gestion git avec fichiers trop gros pour github qui avait bloqué les commit avec pour solution de récréer les repo au propre.