

# MEDIATEK DOCUMENTS

## Sommaire :

Mission 2 : Gérer les documents (page 3)

- 1) Gérer les commandes de dvd et de livres
- 2) Gérer les commandes de revues

Mission 4 : Gérer les authentifications (page 13)

Mission 5 : Assurer la sécurité la qualité et intégrer des logs (page 18)

- 1) Corriger des problèmes de sécurité
- 2) Contrôler la qualité
- 3) Intégrer des logs

Mission 6 : Tester et documenter (page 22)

- 1) Gérer les tests
- 2) Créer les documentations techniques
- 3) Créer la documentation utilisateur

Mission 7 : Déployer et gérer les sauvegardes de données (page 25)

- 1) Déployer le projet
- 2) Gérer les sauvegardes de données

## Contexte :

Je vais travailler sur une application de gestion de documents d'une médiathèque. L'application est codée entièrement en c# et la base de données est accessible depuis une api en php.

Actuellement l'application permet de consulter les livres, les dvd, les revues et d'ajouter des parutions de revues.

Mon travail sera d'ajouter la possibilité de gérer des commandes de livres, de dvd et de revues via des abonnements. Ensuite de réaliser une authentification pour accéder à l'application avec certains droits suivant le statut de la personne. De gérer les tests, la qualité du code, des logs, de faire mes documentations techniques et utilisateur, ainsi que la sauvegarde programmée de la base de données.

## Mission 2 : Gérer les commandes

Intitulé : Permettre les commandes de livres, de dvd et les abonnements aux revues

1) Gérer les commandes de dvd et de livres

Temps estimé 8h

Temps réel 12h

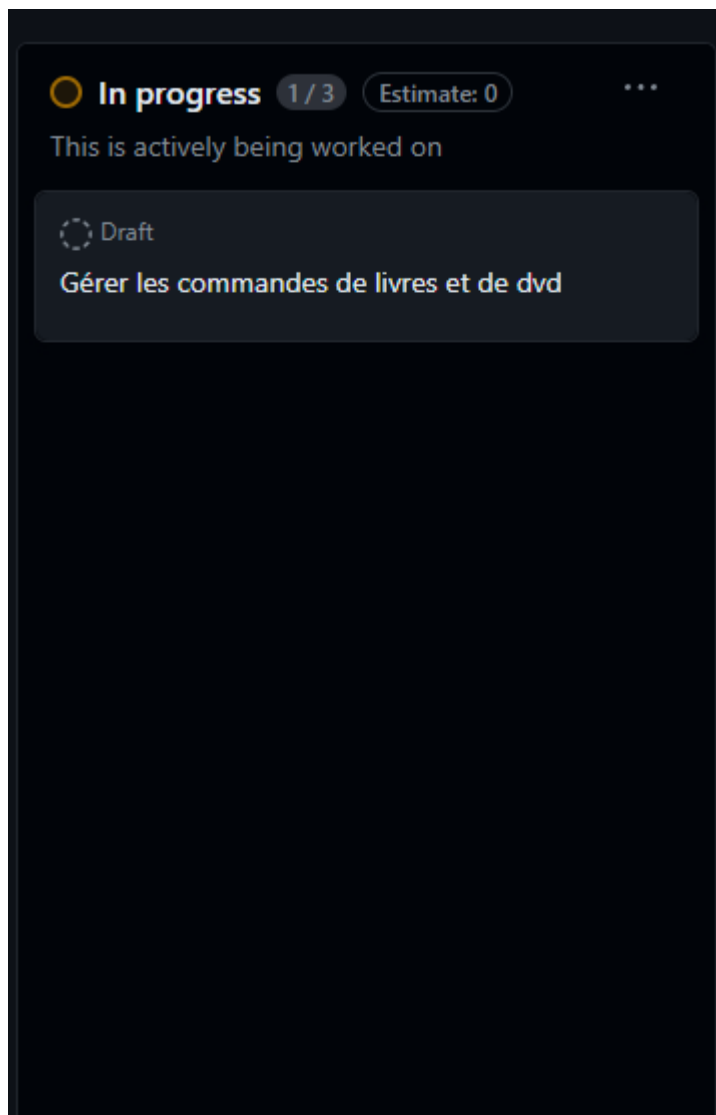
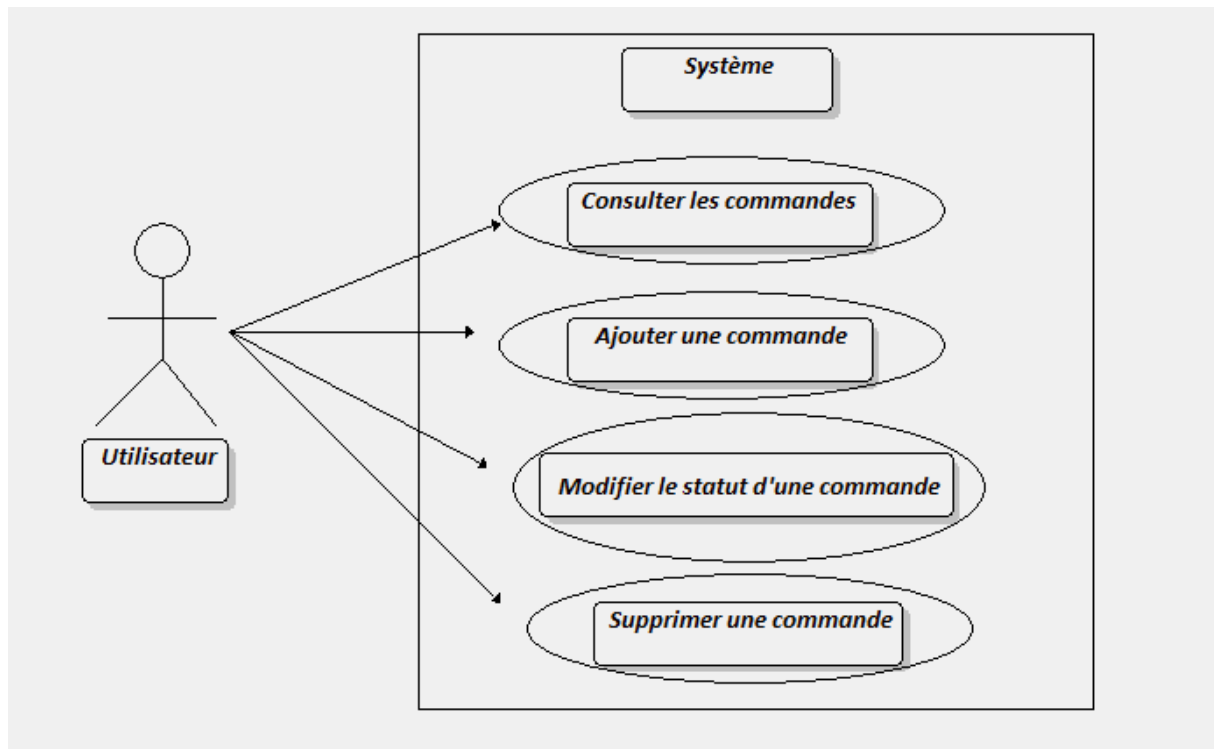
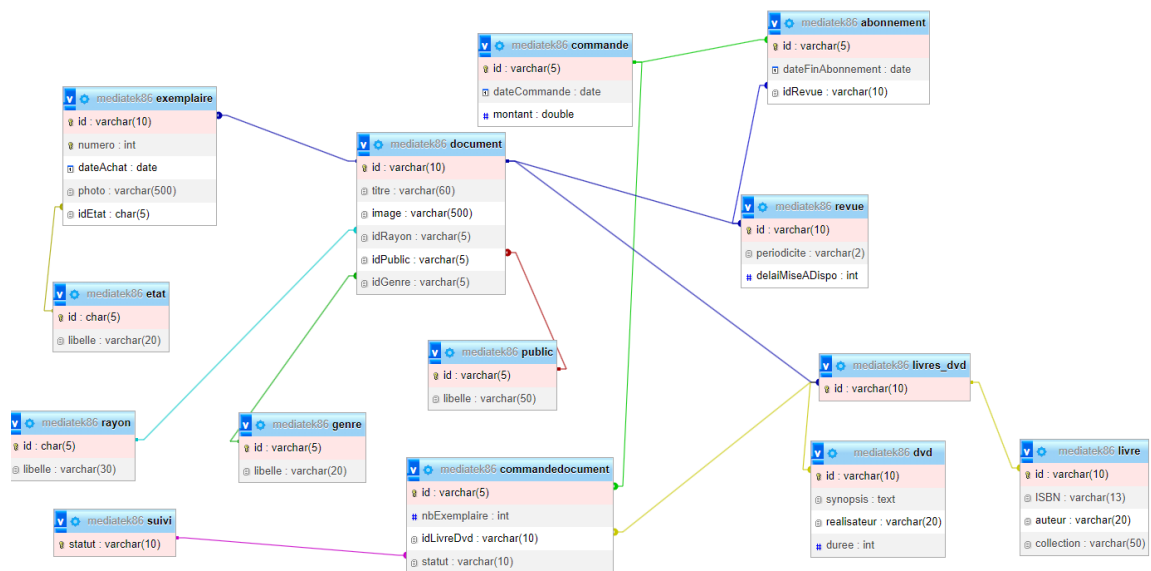


Diagramme UML de la tâche :



MCD de la tâche :



Maquette de la page :

Livres
DVD
Revue
Parutions des revue
Commandes de livre
Commandes de DVD
Commandes de revue

Infos Commande

Nom de la commande :

Date de la commande :

mardi 2 avril 2024

Montant de la commande

Nombre d'exemplaires

Statut de la commande

Livre commandé

Enregistrer la commande

Nouvelle Commande

Modifier Commande

Supprimer Commande

Liste de commandes

Produits

Saisir le titre ou la partie d'un titre :

Saisir un numéro de document :

Rechercher

Ou sélectionner le genre :

X

Ou sélectionner le public :

X

Ou sélectionner le rayon :

X

Explications :

Il faut saisir les informations d'une commande en haut à gauche (le nom est choisi automatiquement) ainsi que le statut est forcément en cours. Concernant le livre commandé il faut sélectionner un livre dans le tableau du bas avec tous les livres, cela ajuste automatiquement le textbox avec le titre du livre. Si l'on clique sur enregistrer, une commande est sauvegardée dans la base de données et s'affiche dans le tableau de droite.

Le principe est exactement le même pour les dvd.

Les méthodes et classes ajoutées :

Toutes les classes et méthodes concernant les commandes de livres sont les mêmes pour les dvd, elles changent juste de type : Livre => Dvd

Premièrement création d'une classe « commande » avec les caractéristiques d'une commande et « commandedocument » héritant de commande.

A l'ouverture de la page, les combo box de livres sont remplies et les livres sont affichés.

A chaque clic sur un item du tableau des commandes, les caractéristiques de la commande sont affichées dans le même formulaire que la création de commande, pour switcher il faut appuyer sur « Nouvelle commande » ce qui vide et rends modifiables les champs. La modification est possible que si on a une commande sélectionnée, il est alors possible de changer le statut de la commande depuis de cliquer sur modifier pour que cela se fasse sous certaines conditions.

L'ajout d'une commande se fait avec une nouvelle méthode dans la classe access :

```
/// <summary>
/// add a new order document
/// </summary>
1 référence
public bool AddCommandeDocument(CommandeDocument commandeDocument)
{
    String jsonCommandeDocument = JsonConvert.SerializeObject(commandeDocument, new CustomDateTimeConverter());
    try
    {
        Console.WriteLine(jsonCommandeDocument);
        // récupération soit d'une liste vide (requête ok) soit de null (erreur)
        List<CommandeDocument> liste = TraitementRecup<CommandeDocument>(POST, "commandedocument/" + jsonCommandeDocument);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}
```

Une requête post vers l'api avec les bonnes informations.

L'ajout dans la vue :

```
/// <summary>
/// Add a new order to the database
/// </summary>
1 référence
private void btnCommandesAjouter_Click(object sender, EventArgs e)
{
    if (txtCommandeMontant.Text == "" || txtCommandeNbExemplaires.Text == "")
    {
        MessageBox.Show("Veuillez remplir tous les champs");
    }
    else
    {
        CommandeDocument commandeDocument = new CommandeDocument(txtCommandeId.Text, dtpCommandeDate.Value, int.Parse(txtCommandeMontant.Text), int.Parse(txtCommandeNbExemplaires.Text), globalId, cbxStatut.Text);
        controller.AddCommandeDocument(commandeDocument);
        fillOrdersFullList();
    }
}
```

Pour cela l'api a été légèrement modifiée pour permettre l'ajout d'une commande (en premier) puis d'une commandedocument en une seule requête.

```

/**
 * ajout d'une commande dans les tables commande et commandedocument
 * @param array $champs nom et valeur de chaque champs de la commande
 * @return true si l'ajout a fonctionné
 */
1 reference | 0 overrides
public function insertCommande($champs)
{
    $champsCommande = [ "id" => $champs["id"], "dateCommande" => $champs["dateCommande"],
        "montant" => $champs["montant"]];
    $champsCommandeDocument = [ "id" => $champs["id"], "nbExemplaire" => $champs["nbExemplaire"],
        "idLivreDvd" => $champs["idLivreDvd"], "statut" => $champs["statut"]];
    $result = $this->insertOne("commande", $champsCommande);
    if ($result == null || $result == false){
        return null;
    }
    return $this->insertOne( "commandedocument", $champsCommandeDocument);
}

```

La modification et la suppression dans la vue :

```

/// <summary>
/// click on the modify button to save the change of the status of the order
/// </summary>
1 référence
private void btnCommandeModifier_Click(object sender, EventArgs e)
{
    bool refusable = false;
    if (globalStatut == "Livrée" || globalStatut == "Réglée")
    {
        if (cbxStatut.Text == "En cours" || cbxStatut.Text == "Relancée")
        {
            MessageBox.Show("La commande est déjà " + globalStatut + " elle ne peut pas revenir à une étape intermédiaire");
            refusable = true;
        }
        else
        {
            refusable = false;
        }
    }
    if (cbxStatut.Text == "Réglée" && globalStatut != "Livrée")
    {
        MessageBox.Show("Il faut que la commande soit livrée pour être réglée");
        refusable = true;
    }

    if (refusable == false)
    {
        controller.updateOrderStatus(txbCommandeId.Text, cbxStatut.Text);
        fillOrdersFullList();
    }
}

/// <summary>
/// click on the delete button to delete the order
/// </summary>
1 référence
private void btnCommandeSupprimer_Click(object sender, EventArgs e)
{
    if (cbxStatut.Text == "Livrée")
    {
        MessageBox.Show("La commande est livrée, elle ne peut pas être supprimée");
    }
    else
    {
        // delete the order
        controller.deleteOrder(txbCommandeId.Text);
        fillOrdersFullList();
        MessageBox.Show("La commande a été supprimée avec succès", "Suppression de commande");
    }
}

```

Dans la classe access :

```

/// <summary>
/// update the status of the order
/// </summary>
1 référence
public bool updateOrderStatus(string idCommande, string idStatus)
{
    String jsonIdStatus = convertToJson("statut", idStatus);
    try
    {
        // récupération soit d'une liste vide (requête ok) soit de null (erreur)
        List<Commande> liste = TraitementRecup<Commande>(PUT, "commandedocument/" + idCommande + "/" + jsonIdStatus);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

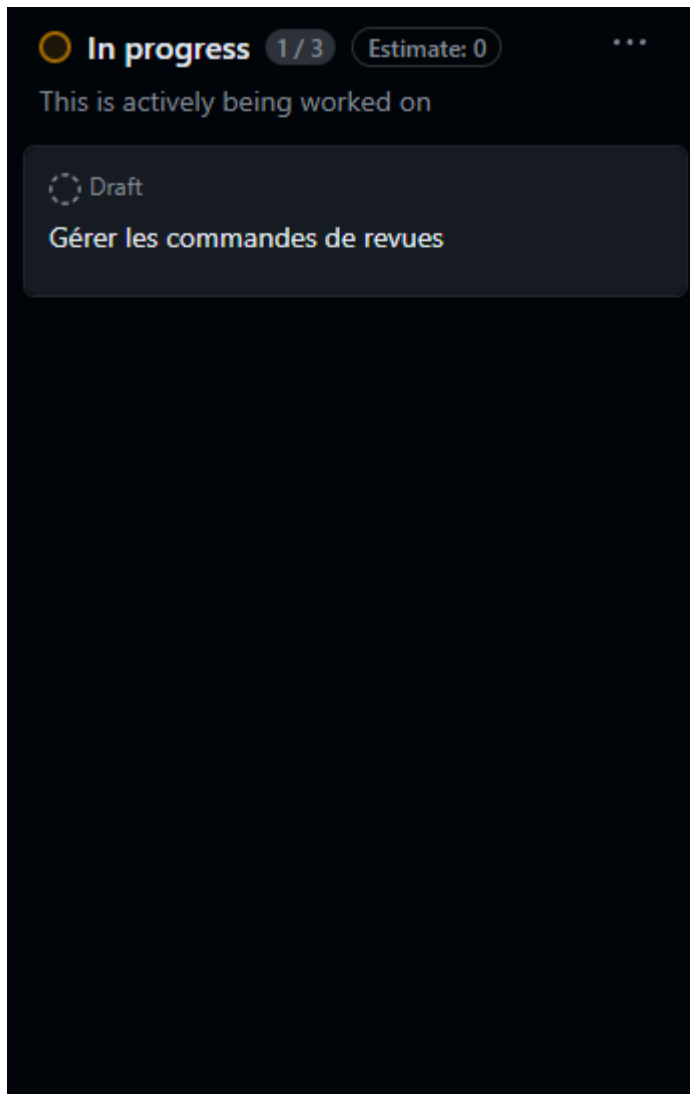
///<summary>
///delete a command document
/// </summary>
1 référence
public bool deleteOrder(string idCommande)
{
    // convert the id to json
    String jsonIdCommande = convertToJson("id", idCommande);
    try
    {
        // récupération soit d'une liste vide (requête ok) soit de null (erreur)
        List<CommandeDocument> liste = TraitementRecup<CommandeDocument>(DELETE, "commande/" + jsonIdCommande);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

```

Le reste des méthodes sont des dérivés très proches des méthodes déjà définies.



## 2) Gérer les commandes de revues



Temps estimé : 4h

Temps réel : 6h

Maquette de la page des revues :

Gestion des documents de la médiathèque

Livres DVD Revues Parutions des revues Commandes de livres Commandes de DVD Commandes de revues

Infos Commande

Nom de la commande : CMDR1

Date de la commande : mardi 2 avril 2024

Montant de la commande : 45

Date fin de la commande : mercredi 3 avril 2024

Revue commandé : Alternatives Economiques

Afficher les abonnements qui finissent dans -30 jours X  
 Enregistrer la commande Nouvelle Commande  
 Supprimer Commande

Liste de commandes

Nom	Date de commande	Montant	Date de fin	Revue commandé
CMDR1	02/04/2024	45	03/04/2024	Alternatives Econ...

Produits

Saisir le titre ou la partie d'un titre :

Saisir un numéro de document :  Rechercher

Ou sélectionner le genre : Actualités X  
 Ou sélectionner le public : Ados X  
 Ou sélectionner le rayon : BD Adultes X

Id	Titre	Periodicite	DelaiMiseADispo	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	Geo	MS	52	Presse Culturelle	Tous publics	Magazines
10009	L'Equipe	QT	5	Presse sportive	Adultes	Presse quotidienne
10010	L'Equipe Magazine	HB	12	Presse sportive	Adultes	Magazines
10008	L'Obs	HB	26	Actualités	Adultes	Magazines

C'est à peu près le même principe que pour les commandes de livres et de dvd mais l'abonnement n'est pas modifiable et qu'il y a un bouton pour afficher les abonnements prenant fin dans – de 30 jours.

Il y eu l'ajout de la classe abonnement prenant en paramètre les infos d'un abonnement :

```

namespace MediaTekDocuments.model
{
    29 références
    public class Abonnement : Commande
    {
        6 références
        public DateTime dateFinAbonnement { get; }
        3 références
        public string idRevue { get; }

        1 référence
        public Abonnement(string id, DateTime dateCommande, float montant, DateTime dateFinAbonnement, string idRevue) : base(id, dateCommande, montant)
        {
            this.dateFinAbonnement = dateFinAbonnement;
            this.idRevue = idRevue;
        }
    }
}

```

Methode d'ajout d'abonnement dans la vue :

```

/// <summary>
/// Add a new order to the database
/// </summary>
1 référence
private void btnCommandeAjouterRevue_Click(object sender, EventArgs e)
{
    DateTime lastDateExemplaire = controller.setDateFinAbonnement(globalIdRevue);
    if (txbCommandeMontantRevue.Text == "")
    {
        MessageBox.Show("Veuillez remplir tous les champs");
    }
    else if (dtpFinCommande.Value < DateTime.Now)
    {
        MessageBox.Show("La date de fin de l'abonnement doit être après la date d'aujourd'hui");
    }
    else if (dtpFinCommande.Value < lastDateExemplaire)
    {
        MessageBox.Show("La date de fin de l'abonnement doit être après la date de fin du dernier exemplaire qui est actuellement au : " + lastDateExemplaire);
    }
    else if (dtpFinCommande.Value < dtpCommandeDateRevue.Value)
    {
        MessageBox.Show("La date de fin de l'abonnement doit être après la date de commande");
    }
    else if (dtpFinCommande.Value == dtpCommandeDateRevue.Value)
    {
        MessageBox.Show("La date de fin de l'abonnement doit être après la date de commande");
    }
    else
    {
        Abonnement abonnement = new Abonnement(txbCommandeIdRevue.Text, dtpCommandeDateRevue.Value, int.Parse(txbCommandeMontantRevue.Text), dtpFinCommande.Value, globalIdRevue);
        controller.AddAbonnement(abonnement);
        fillOrdersFullListRevue();
    }
}

```

Methode setDateFinAbonnement qui permet de retrouver la date de la dernière parution de la revue :

```

1 référence
public DateTime setDateFinAbonnement(string id)
{
    List<Exemplaire> exemplaires = GetExemplairesRevue(id);
    DateTime dateFin = new DateTime();

    // for each exemplaire, we check if the date is the latest
    foreach (Exemplaire exemplaire in exemplaires)
    {
        if (exemplaire.DateAchat > dateFin)
        {
            dateFin = exemplaire.DateAchat;
        }
    }
    return dateFin;
}

```

Ajout d'un abonnement dans access :

```

/// <summary>
/// add abonnement
/// </summary>
1 référence
public bool AddAbonnement(Abonnement abonnement)
{
    String jsonAbonnement = JsonConvert.SerializeObject(abonnement, new CustomDateTimeConverter());
    try
    {
        Console.WriteLine(jsonAbonnement);
        // récupération soit d'une liste vide (requête ok) soit de null (erreur)
        List<Abonnement> liste = TraitementRecup<Abonnement>(POST, "abonnement/" + jsonAbonnement);
        return (liste != null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return false;
}

```

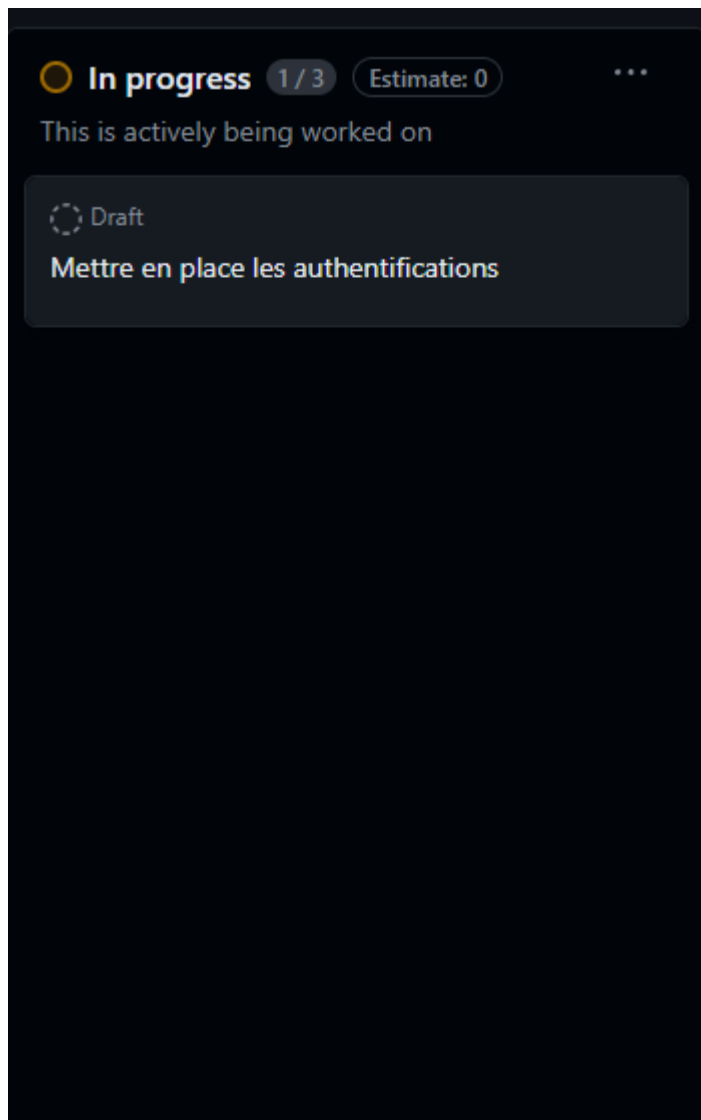
2 méthodes permettant de récupérer les abonnements prenant fin dans – de 30 jours, une permet l’affichage dans le tableau, l’autre dans un popup au lancement de l’app :

```
/// <summary>
/// display the abonnement where the end date is lower than 30 days
/// </summary>
1 référence
private void btnCommandeRelanceRevue_Click(object sender, EventArgs e)
{
    List<Abonnement> abonnements = controller.GetAllAbonnements();
    List<Abonnement> abonnementsRelance = new List<Abonnement>();
    foreach (var abonnement in abonnements)
    {
        if (abonnement.dateFinAbonnement < DateTime.Now.AddDays(30))
        {
            abonnementsRelance.Add(abonnement);
        }
    }
    fillOrdersListRevue(abonnementsRelance);
}

/// <summary>
/// display the abonnement where the end date is lower than 30 days
/// </summary>
1 référence
private void commandeRelanceRevue()
{
    List<Abonnement> abonnements = controller.GetAllAbonnements();
    List<Abonnement> abonnementsRelance = new List<Abonnement>();
    foreach (var abonnement in abonnements)
    {
        if (abonnement.dateFinAbonnement < DateTime.Now.AddDays(30))
        {
            abonnementsRelance.Add(abonnement);
        }
    }

    // show in a small pop up the list of abonnements to relance in string format
    string abonnementsRelanceString = "";
    foreach (var abonnement in abonnementsRelance)
    {
        abonnementsRelanceString += abonnement.id + " - Date de fin : " + abonnement.dateFinAbonnement.ToShortDateString();
        abonnementsRelanceString += "\n";
    }
    MessageBox.Show(abonnementsRelanceString, "Abonnements se terminant dans moins de 30 jours");
}
```

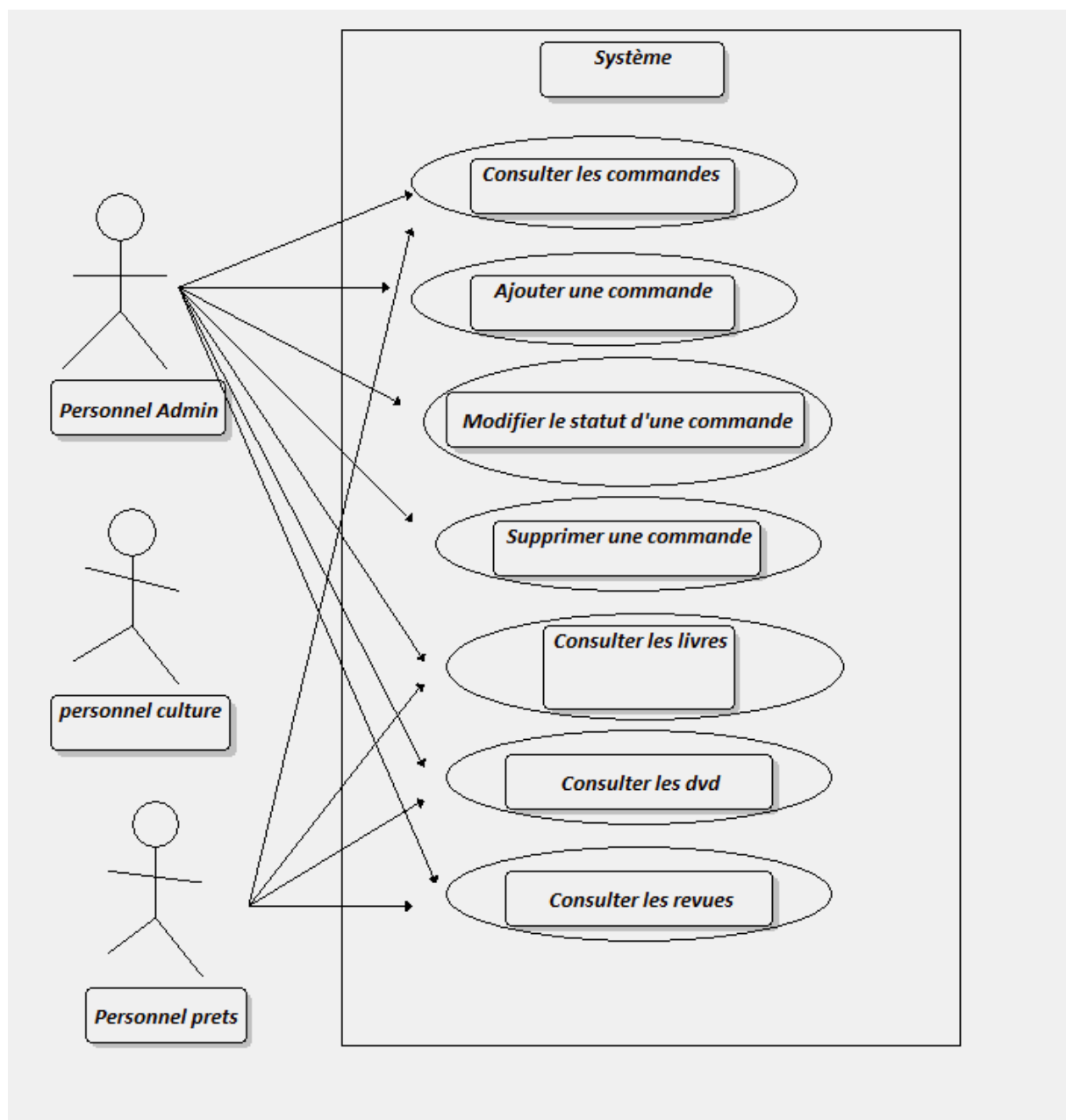
## Mission 4 : Mettre en place des authentifications



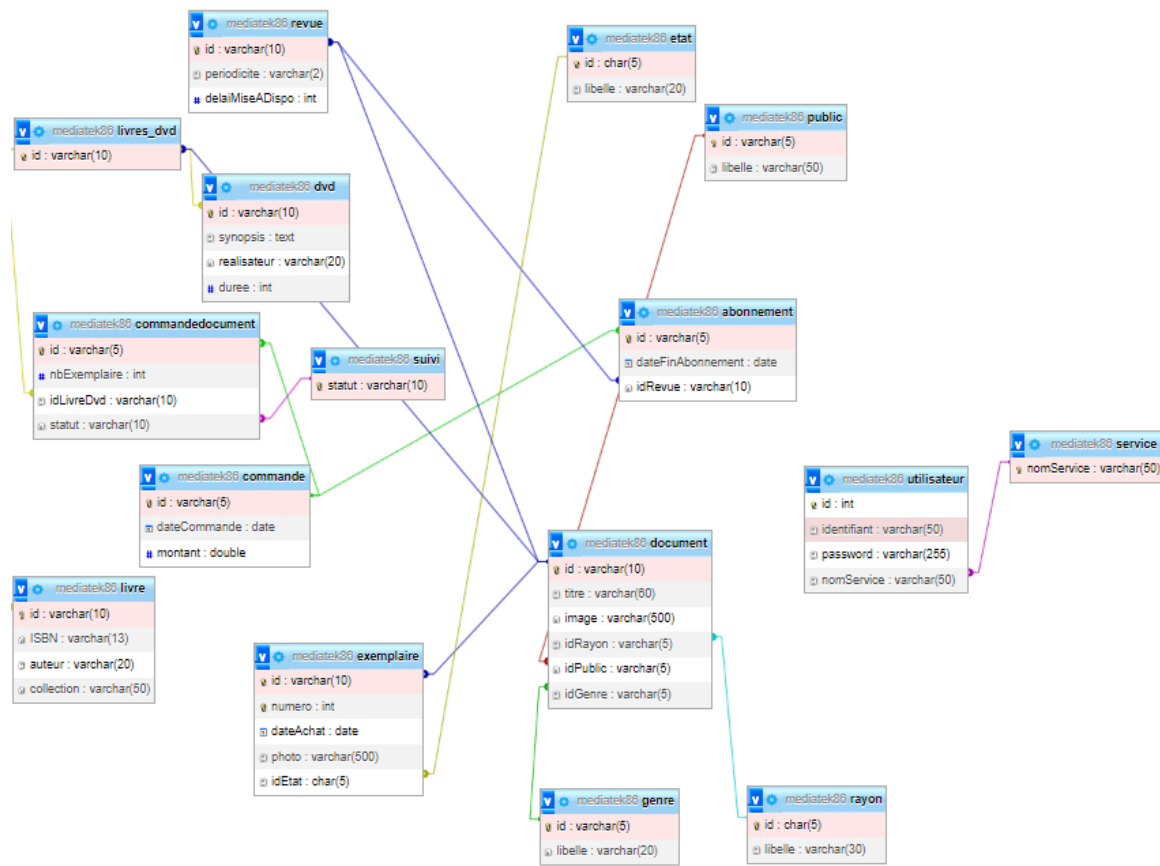
Temps estimé 4h

Temps réel 4h

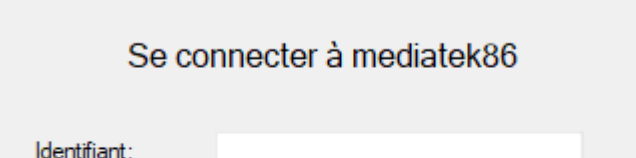
Diagramme UML de la tâche :



MCD (ajout de la table utilisateur et service) :



Maquette de la page d'authentification :



Form1

Se connecter à mediatek86

Identifiant:

Mot de passe:

Valider

### Fenêtre permettant d'accéder ou non à l'application

Création d'une classe utilisateur, ainsi que d'une nouvelle vue et un contrôleur.

Vue admin :

```
namespace MediatekDocuments.view
{
    3 références
    public partial class FrmMediatekLogin : Form
    {
        private readonly FrmMediatekControllerLogin controller;

        1 référence
        public FrmMediatekLogin()
        {
            InitializeComponent();
            controller = new FrmMediatekControllerLogin();
        }

        1 référence
        private void button1_Click(object sender, EventArgs e)
        {
            Utilisateur utilisateur = controller.checkLogin(textBox1.Text, textBox2.Text);
            if (utilisateur != null)
            {
                FrmMediatek frmMediatek = new FrmMediatek(utilisateur);
                try { frmMediatek.Show(); }
                catch { this.Hide(); }

                catch (Exception ex) { MessageBox.Show("Vous n'avez pas les droits pour utiliser cette application"); }
            }
            else
            {
                MessageBox.Show("Login ou mot de passe incorrect");
            }
        }
    }
}
```

Classe utilisateur :

```
12 références
public class Utilisateur
{
    0 références
    private int id { get; }
    1 référence
    private string identifiant { get; }
    2 références
    public string password { get; }
    4 références
    public string nomService { get; }

    0 références
    public Utilisateur(string identifiant, string password, string nomService)
    {
        this.identifiant = identifiant;
        this.password = password;
        this.nomService = nomService;
    }
}
```



Méthodes ajoutées dans la classe access :

```
1 référence
public Utilisateur checkLogin(string login, string password)
{
    String jsonLogin = convertToJson("identifiant", login);
    IEnumerable<Utilisateur> user = TraitementRecup<Utilisateur>(GET, "utilisateur/" + jsonLogin);
    if (user.Count() == 0)
    {
        return null;
    }

    Utilisateur utilisateur = new List<Utilisateur>(user)[0];
    string userPassword = decryptPassword(utilisateur.password);
    if (userPassword.Equals(password))
    {
        return utilisateur;
    }
    return null;
}

// two function to encrypt and decrypt the password
0 références
private string hashPassword(string password)
{
    string encryptedPassword = "";
    foreach (char c in password)
    {
        encryptedPassword += (char)(c + 1);
    }
    return encryptedPassword;
}

1 référence
private string decryptPassword(string password)
{
    string decryptedPassword = "";
    foreach (char c in password)
    {
        decryptedPassword += (char)(c - 1);
    }
    return decryptedPassword;
}
```

J'aurai pu utiliser un plug-in pour le hash mais j'ai décidé de l'encoder moi-même, ce n'est pas très sécurisé mais bon...

Modification du constructeur de la vue de l'application, permettant de refuser l'accès à l'utilisateur faisant parti du personnel culture, ou de retrouver le service des autres (pour ajuster les droits en fonctions).

```
1 référence
internal FrmMediatek(Utilisateur currentUser)
{
    InitializeComponent();
    this.controller = new FrmMediatekController();
    bool allowed = controller.accessApp(currentUser);
    if (!allowed)
    {
        this.Close();
    }
    service = controller.verifService(currentUser);
}
```

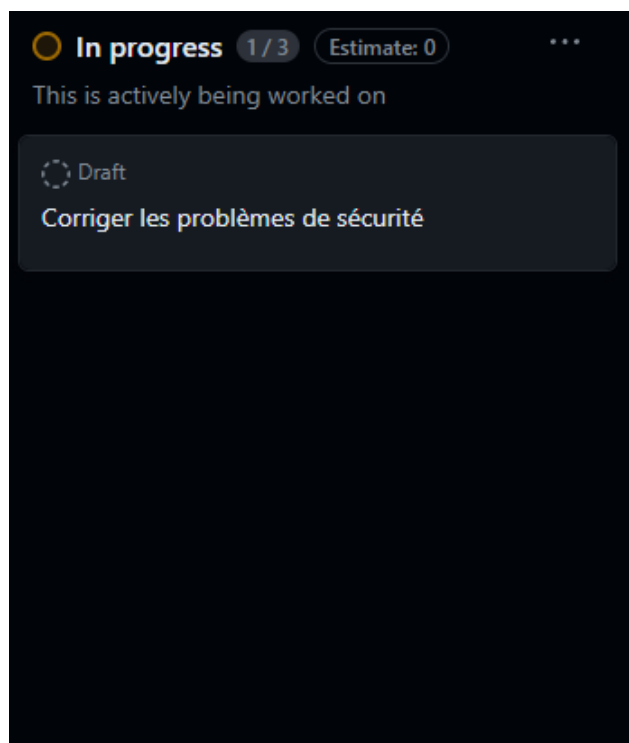
Un exemple de contrainte pour les commandes de revues concernant les membres du personnel prêts :

```
1 référence
private void TabCommandeRevue_Enter(object sender, EventArgs e)
{
    if (service == "Prêts")
    {
        tabOngletsApplication.TabPages[6].Enabled = false;
    }
    else
    {
        lesRevue = controller.GetAllRevue();
        RemplirComboCategorie(controller.GetAllGenres(), bdgGenres, cbxBooksGenresRevue);
        RemplirComboCategorie(controller.GetAllPublics(), bdgPublics, cbxBooksPublicsRevue);
        RemplirComboCategorie(controller.GetAllRayons(), bdgRayons, cbxBooksRayonsRevue);
        fillBooksFullListRevue();
        fillOrdersFullListRevue();
        btnCommandeNouveauRevue_Click(sender, e);
        commandeRelanceRevue();
    }
}
```

Ils n'ont donc pas accès aux commandes.

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

### 1) Corriger des problèmes de sécurité



Temps estimé : 3h

Temps réel : 2h

Pour régler le premier problème il fallait écrire les données dans le fichier settings. Et l'appeler depuis la classe access pour les utiliser.

Les paramètres de l'application vous permettent de stocker et de récupérer dynamiquement les paramètres de propriété et d'autres informations pour votre application la prochaine exécution de l'application. [En savoir plus sur les paramètres de l'application...](#)

	Nom	Type	Portée	Valeur
	mediatek86Aut...	(Chaîne de ...	Application	admin:adminpwd
	mediatek86Con...	(Chaîne de ...	Application	server=localhost;user id=root;database=mediatek86
*				

```
private Access()
{
    String authenticationString;
    try
    {
        Log.Logger = new LoggerConfiguration()
            .MinimumLevel.Verbose()
            .WriteTo.Console()
            .WriteTo.File("log.txt", rollingInterval: RollingInterval.Day)
            .CreateLogger();

        authenticationString = GetConnectionStringByName(connectionName);
        api = ApiRest.GetInstance(uriApi, authenticationString);
    }
}

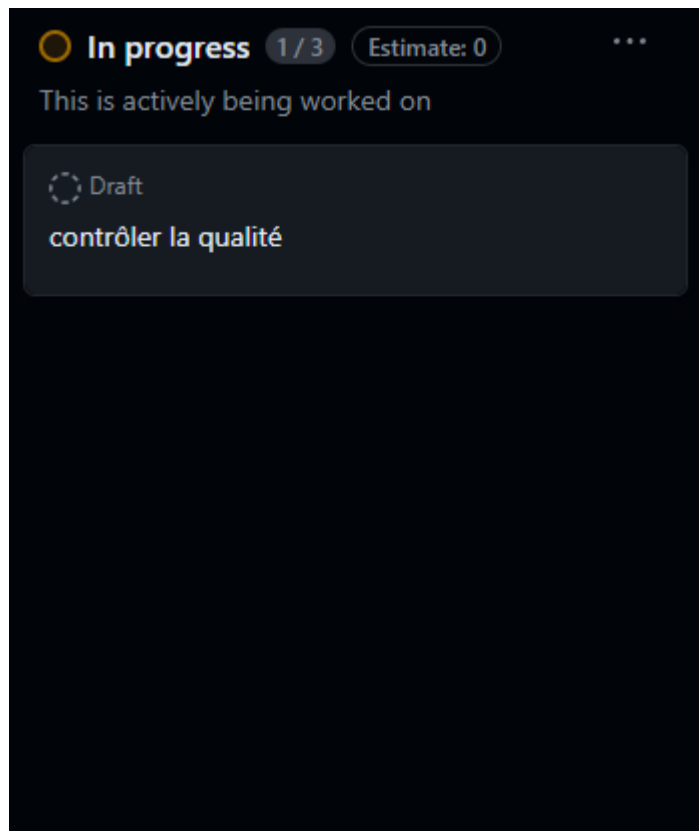
public class Access
{
    private static readonly string connectionName = "MediaTekDocuments.Properties.Settings.mediatek86AuthenticationString";
    /// <summary>
    /// adresse de l'API

```

Pour la deuxième tâche il suffisait juste d'écrire « Options -Indexes » dans htaccess de l'api et les fichiers ne sont plus accessibles.

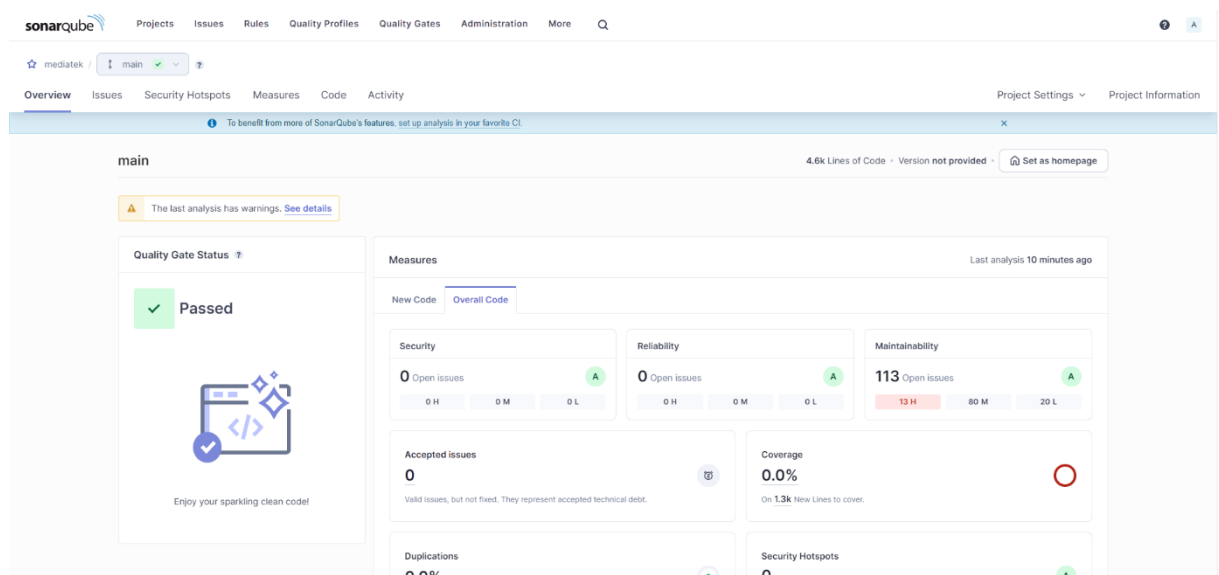
```
RewriteEngine on
RewriteRule ^([a-zA-Z]+)$ mediatekdocuments.php?table=$1
RewriteRule ^([a-zA-Z]+)/({.*})$ mediatekdocuments.php?table=$1&champs=$2
RewriteRule ^([a-zA-Z]+)/([a-zA-Z0-9]+)/({.*})$ mediatekdocuments.php?table=$1&id=$2&champs=$3
Options -Indexes
```

## 2) Contrôler la qualité

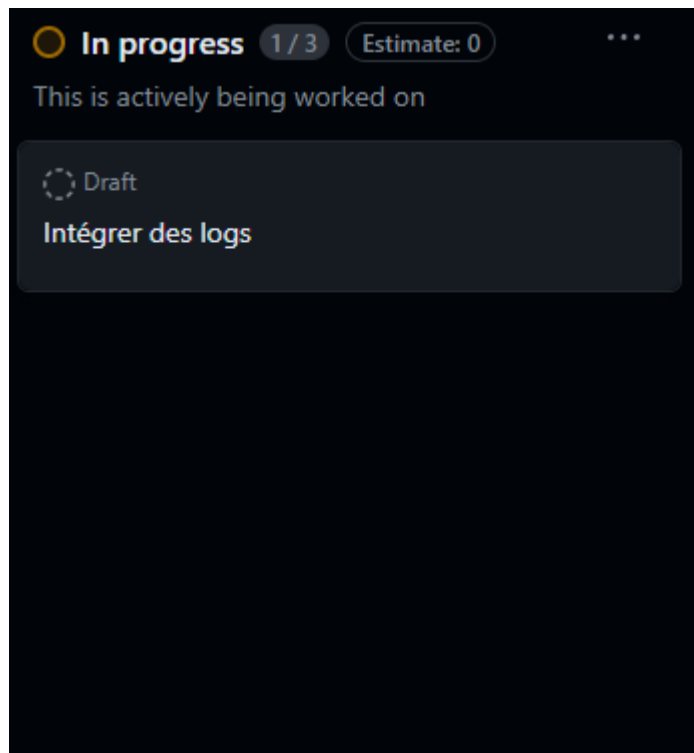


Temps estimé : 1 h

Temps réel : 1 h



## 3) Intégrer des logs



Temps estimé : 2h

Temps réel : 2h

Code pour générer des logs avec serilog :

```
Log.Logger = new LoggerConfiguration()  
    .MinimumLevel.Verbose()  
    .WriteTo.Console()  
    .WriteTo.File("log.txt", rollingInterval: RollingInterval.Day)  
    .CreateLogger();
```

Exemple :

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

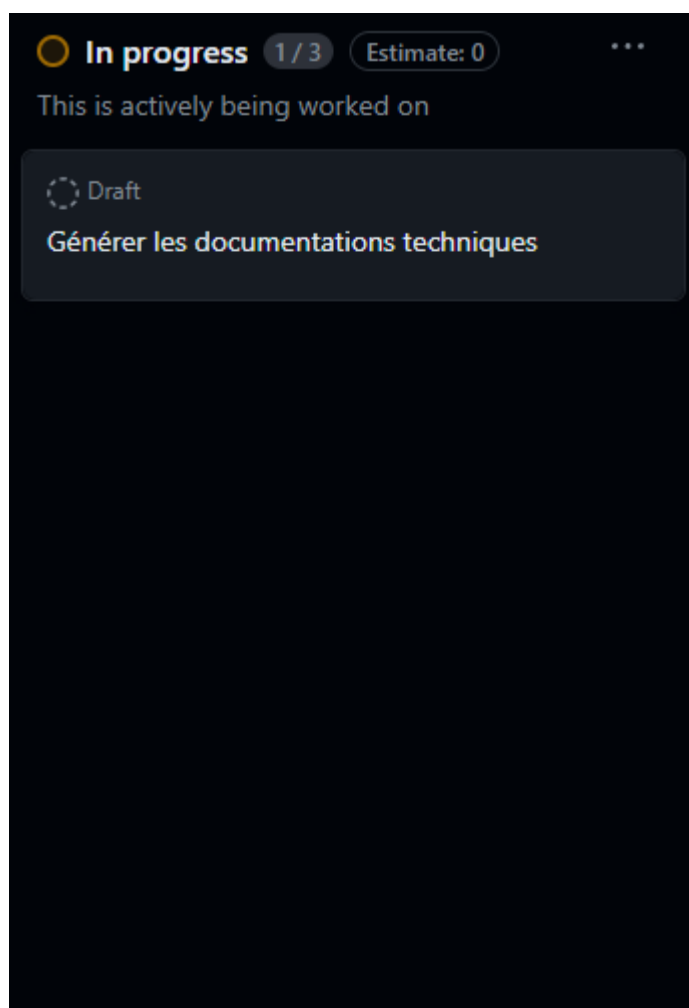
## Mission 6 : Tester et documenter

### 1) Gérer les tests

Tests unitaires : aucune méthode dans le package model

Tests fonctionnels : infaisable du a une erreur specflow que je ne peux pas résoudre.

### 2) Créer les documentations techniques



Temps estimé 1h

Temps réel 1h

Exemples de commentaires normalisés :

```

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

```

```

1 reference | 0 overrides
public function selectAll($table){
    if($this->conn != null){
        switch ($table) {
            case "livre" :

```

```

/// <summary>
/// getter sur la liste des genres
/// </summary>
/// <returns>Liste d'objets Genre</returns>
6 références
public List<Categorie> GetAllGenres()
{
    return access.GetAllGenres();
}

```

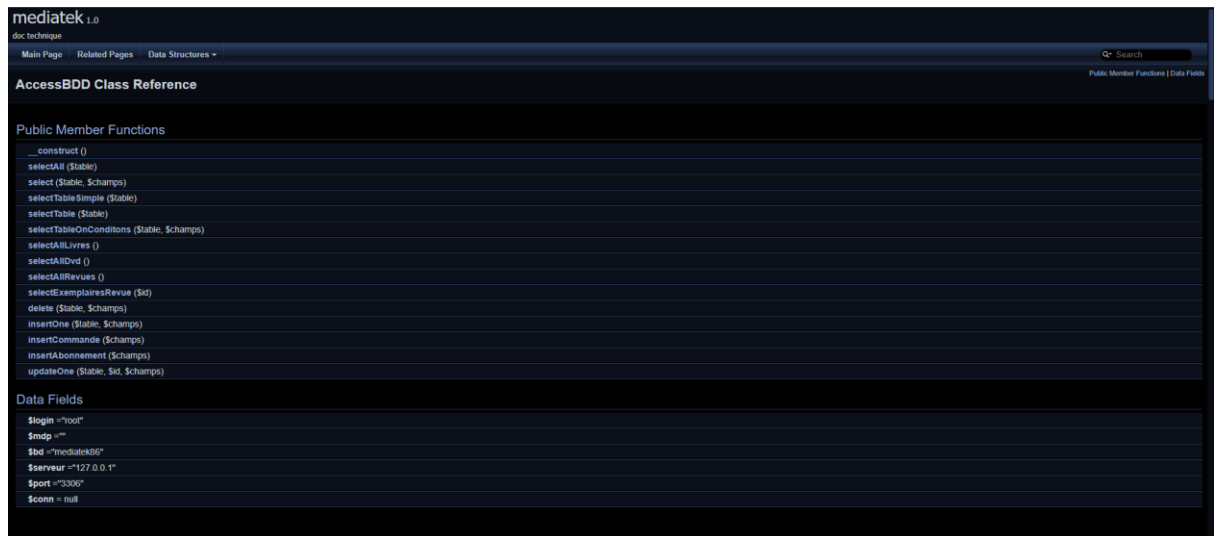
## Documentation de l'application :

### Doc Mediatek Documents 1.0

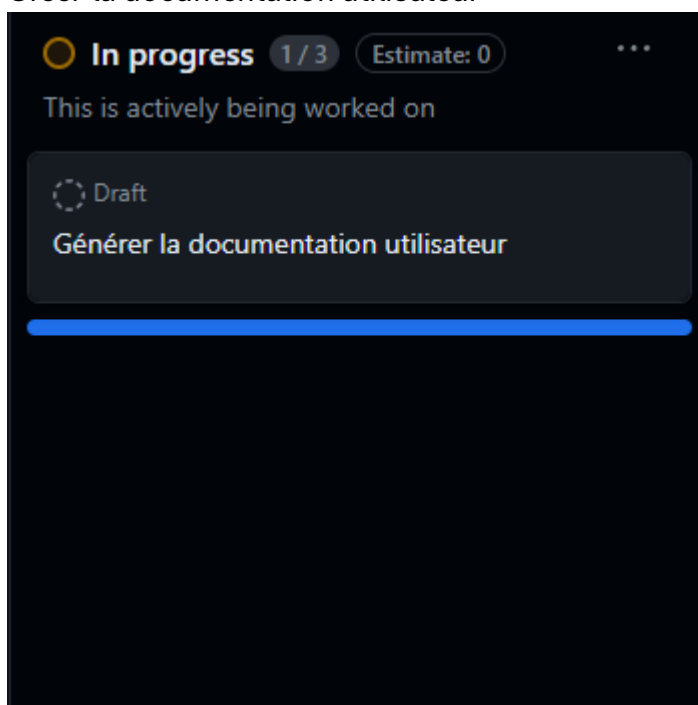
Documentation de l'application mediatek documents

Main Page	Packages	Classes
Package List		
<b>Package List</b>		
Here are the packages with brief descriptions (if available).		
[detail level: 1 2 3]		
MediaTekDocuments		
controllier		
FrmMediatekControllier	Contrôleur lié à FrmMediatek	
FrmMediatekControllierLogin		
dal		
Access	Classe d'accès aux données	
manager		
ApiRest	Classe indépendante d'accès à une api rest avec éventuellement une "basic authorization"	
model		
Abonnement		
Categorie	Classe métier Categorie (réunit les informations des classes Public, Genre et Rayon)	
Commande		
CommandeDocument		
Document	Classe métier Document (réunit les informations communes à tous les documents : Livre, Revue, Dvd)	
Dvd	Classe métier Dvd hérite de LivreDvd : contient des propriétés spécifiques aux dvd	
Etat	Classe métier Etat (état d'usure d'un document)	
Exemplaire	Classe métier Exemplaire (exemplaire d'une revue)	
Genre	Classe métier Genre : hérite de Categorie	
Livre	Classe métier Livre hérite de LivreDvd : contient des propriétés spécifiques aux livres	
LivreDvd	Classe métier LivreDvd hérite de Document	
Public	Classe métier Public (public concerné par le document) hérite de Categorie	
Rayon	Classe métier Rayon (rayon de classement du document) hérite de Categorie	

Documentation de l'api :



### 3) Créer la documentation utilisateur



Temps estimé 2h

Temps réel 1h

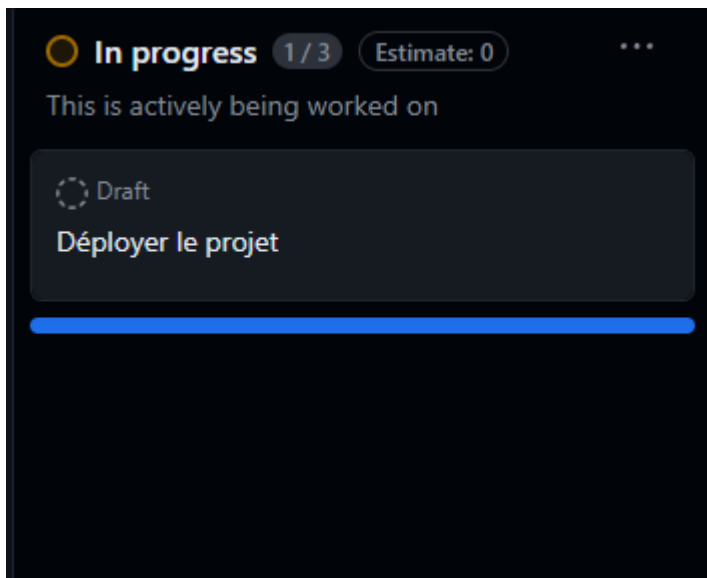
Outils utilisés :

Temps de la vidéo :



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

### 1) Déployer le projet



Temps estimé 3h

Temps réel 3h

L'api a été déployée sur hostinger accessible depuis l'adresse [mediatekdocumentsapi.online](https://mediatekdocumentsapi.online) suivi des endpoints correspondants.

Modifications apportées au code de l'application et de l'api :

```
public $bd="u482619961_mediatek86";  
1 reference  
public $serveur="localhost";  
1 reference  
public $port="3306";  
16 references  
public $conn = null;
```

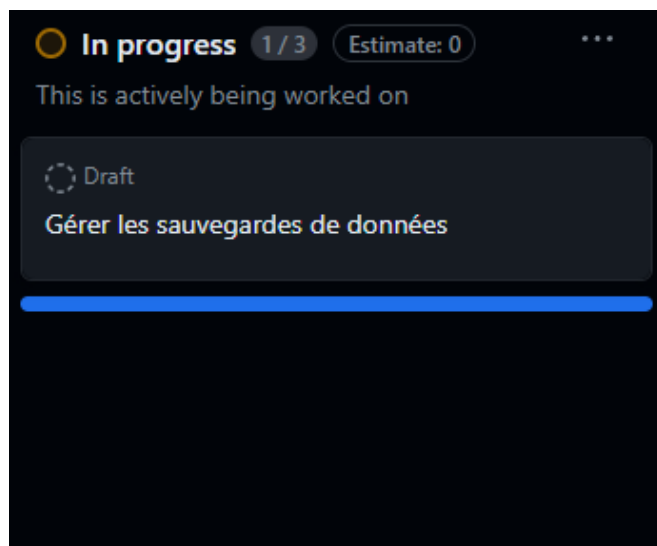
```

1 reference
public $login="u482619961_mateo";
/// adresse de l'API
/// </summary>
private static readonly string uriApi = "https://mediatekdocumentsapi.online/";
/// <summary>
/// instance unique de la classe

```

L'installateur a été généré depuis visual studio via clickOnce.

## 2) Gérer les sauvegardes de données



Temps estimé 1h

Temps réel 2h

La sauvegarde est en place

Name ↑	Size	Last modified	
backup.sh	340 B	a minute ago	-rw-rw-rw-
bddbbackup_2024-04-17.sql.gz	4.82 KiB	a minute ago	-rw-r--r--

Toutes les minutes une sauvegarde de la bdd est faite, pour la restaurer il suffit de récupérer et de dézipper le fichier bddbbackup.sql.gz, et de récupérer le fichier .sql afin de l'importer dans phpmyadmin.

### Bilan :

L'application est finie j'ai pu réaliser toutes les tâches (sauf celles facultatives) hormis la phase de test fonctionnel sur laquelle il était impossible de faire fonctionner specflow. Les tests unitaires non plus car je ne possédais aucune méthodes dans les classes du package model donc il n'y en a pas.