

## Mediatek Documents

[Mission 1 : gérer les document](#)

[Mission 2 : gérer les commandes](#)

[Mission 3 : mettre en place des authentifications](#)

[Mission 4 : mettre en place des authentifications :](#)

[Mission 5 : assurer la sécurité, la qualité et intégrer des logs](#)

[Mission 6 : tester et documenter](#)

[Mission 7 : déployer et gérer les sauvegardes de données](#)

[Bilan](#)

Présentation du projet : Contexte 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.

[Mission 1 : gérer les document](#)

### Mission 1 : gérer les documents #1

[Open](#) Codeuraxe/MediaTekDocuments [Private](#)

 Codeuraxe opened 1 minute ago

edited by Codeuraxe · Edits · ...

Dans les onglets actuels (Livres, Dvd, Revues), ajouter les fonctionnalités (boutons) qui permettent d'ajouter, de modifier ou de supprimer un document. Un document ne peut être supprimé que s'il n'a pas d'exemplaire rattaché, ni de commandes. La modification d'un document ne peut pas porter sur son id. Toutes les sécurités seront mises en place pour éviter des erreurs de manipulation. Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Document, idem pour LivresDvd.



 Codeuraxe added this to  @Codeuraxe's Mediatekdocuments 1 minute ago

[Edit](#)   ... 

Assignees

No one - [Assign yourself](#)

Labels

No labels

Projects

 @Codeuraxe's Mediatekdocuments

Status [In progress](#) ^

Création et nommage des boutons :

Les boutons sont créés et nommés selon une convention claire et cohérente, par exemple btnAjouterLivres, pour assurer une compréhension et une maintenabilité optimale du code.

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		Aventures	Ados	Beaux Livres
00007	Dans les coulisses d...	Kate Atkinson		Roman	Tous publics	Littérature étrangère
00003	Et je danse aussi	Anne Laure Bondou		Comédie	Tous publics	Littérature française
00019	Guide Vert - Iles Can...		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'armée furieuse	Fred Vargas	Commissaire Adamsb...	Policier	Adultes	Policiers français étra...

Informations détaillées

Numéro de document :

Code ISBN :

Titre :

Auteur(e) :

Collection :

Genre : 

Aventures

Public : 

Ados

Rayon : 

Beaux Livres

Chemin de l'image :

Image :

Ajouter

Modifier

Supprimer

Annuler

Valider

Logique événementiel

Logique événementielle :

La méthode enCoursModifLivres, appelée lors de l'ouverture de l'onglet avec le paramètre false, gère une grande partie de la logique événementielle de l'application.

```

/// <summary>
/// Ouverture de l'onglet Livres :
/// appel des méthodes pour remplir le datagrid des livres et des combos (genre, rayon, public)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void TabLivres_Enter(object sender, EventArgs e)
{
    lesLivres = controller.GetAllLivres();
    RemplirComboCategorie(controller.GetAllGenres(), bdgGenres, cbxLivresGenres);
    RemplirComboCategorie(controller.GetAllPublics(), bdgPublics, cbxLivresPublics);
    RemplirComboCategorie(controller.GetAllRayons(), bdgRayons, cbxLivresRayons);
    RemplirComboCategorie(controller.GetAllGenres(), bdgGenresInfo, cbxLivresGenresInfo);
    RemplirComboCategorie(controller.GetAllPublics(), bdgPublicsInfo, cbxLivresPublicInfo);
    RemplirComboCategorie(controller.GetAllRayons(), bdgRayonsInfo, cbxLivresRayonInfo);
    RemplirComboEtat(controller.GetAllEtats(), bdgEtats, cbxLivresExEtat);
}

```

Un booléen global a été ajouté pour différencier les ajouts des modifications. La variable `EnCoursModif` (livres, DVD, revues) est toujours initialisée à `false`, tandis que le bouton `btnAjouterLivres` la met à `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 = "";
}

```

Les boutons 'Annuler' et 'Valider' ne sont accessibles qu'en mode modification. Inversement, les trois autres boutons sont inaccessibles lorsque la modification n'est pas en cours.

## Communication avec la base de données.

Certaines requêtes à l'API (ex : `([a-zA-Z]+)`) ne fonctionnent pas avec `livres_dvd`. Les espaces et caractères spéciaux dans les champs JSON provoquent des erreurs 404. Pour y remédier, les `RewriteRule` ont été modifiées comme suit :

`([a-zA-Z0-9_])` permet de prendre en compte les caractères de l'alphabet, les chiffres ainsi que le souligné (`_`).

`[B L]` indique à Apache d'échapper les caractères non-alphanumériques avant d'appliquer la transformation. Le `L` (last) permet d'indiquer que la règle courante doit être appliquée immédiatement sans tenir compte des règles ultérieures. Ainsi, les espaces et autres caractères provoquant l'erreur 404 sont correctement traités par l'API.

Asynchronisme de l'API :

L'API est asynchrone, chaque requête est unique. En cas de requêtes multiples envoyées successivement, les premières peuvent ne pas être encore traitées, ce qui peut poser problème pour les requêtes interdépendantes. 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.

Par exemple, pour CreerLivre :

Dans l'application C#, des ComboBox sont mises en place pour faciliter l'interface utilisateur dans les informations détaillées. La logique est identique pour la gestion des DVD et des revues, donc elles ne sont pas présentées ici.

```
/// <param name="livre">Livre à créer.</param>
/// <returns>Vrai si l'opération est réussie.</returns>
1 référence
public bool CreerLivre(Livre livre)
{
    return access.CreerEntite("livre", JsonConvert.SerializeObject(livre));
}
```

Recherches

Saisir le titre ou la partie d'un titre :

Ou sélectionner le genre :  X

Saisir un numéro de document :

Ou sélectionner le public :  X

Ou sélectionner le rayon :  X

Id	Titre	Duree	Realisateur	Genre	Public	Rayon
20003	Jurassic Park	128	Steven Spielberg	Science Fiction	Tous publics	DVD films
20002	Le seigneur des anne...	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 ...	124	George Lucas	Science Fiction	Tous publics	DVD films

Informations détaillées

Número de document :

Durée :

Titre :

Réalisateur(trice) :

Synopsis :

Genre :

Public :

Rayon :

Chemin de l'image :

Image :

## Mission 2 : gérer les commandes :

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

Créer un onglet (ou une nouvelle fenêtre) pour gérer les commandes de livres.

La charte graphique doit correspondre à l'existant.

Dans toutes les listes, permettre le tri sur les colonnes.

Dans l'onglet (ou la fenêtre), permettre la sélection d'un livre par son numéro, afficher les informations du livre ainsi que la liste des commandes, triée par date (ordre inverse de la chronologie). La liste doit comporter les informations suivantes : date de la commande, montant, nombre d'exemplaires commandés et l'étape de suivi de la commande.

Créer un groupbox qui permet de saisir les informations d'une nouvelle commande et de l'enregistrer. Lors de l'enregistrement de la commande, l'étape de suivi doit être mise à "en cours".

Permettre de modifier l'étape de suivi d'une commande en respectant certaines règles (une commande livrée ou réglée ne peut pas revenir à une étape précédente (en cours ou relancée), une commande ne peut pas être réglée si elle n'est pas livrée).

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

Créer le trigger qui se déclenche si une commande passe à l'étape "livrée" et qui crée autant de tuples dans la table "Exemplaire" que nécessaires, en valorisant la date d'achat avec la date de commande et en mettant l'état de l'exemplaire à "neuf". Le numéro d'exemplaire doit être séquentiel par rapport au livre concerné.

Créer un onglet pour gérer les commandes de DVD en suivant la même logique que pour les commandes de livres.

Toutes les sécurités seront mises en place pour éviter des erreurs de manipulation.

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

Labels

No labels

Projects

@Codeuraxe's Mediatek Documents

Status

In Progress

Priority

Filter options

Size

Filter options

Estimate

Enter number...

Iteration

Choose an iteration

Start date

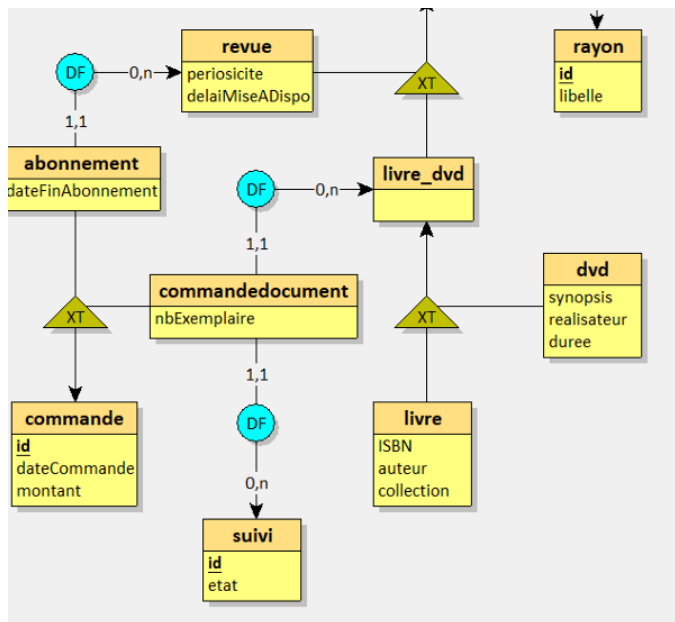
No date

End date

No date

Milestone

No milestone



## Modification de la base de données

Création de la table Suivi et ajout de la clé idSuivi dans la table commandeDocument.

## Création de l'interface graphique.

Ajout et configuration des nouveaux onglets en suivant le modèle des précédents. Les groupes peuvent être copiés pour configurer les onglets restants. Les livres sont affichés dans une DataGridView en haut, et les commandes associées en bas.

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

\*Rechercher  
Saisir le titre ou la partie d'un titre :  Ou sélectionner le genre :  X  
Saisir un numéro de document :  Rechercher le public :  X  
le rayon :  X

Id	Titre	Auteur	Collection	Genre	Public	Rayon
00017	Catastrophes au Brésil	Philippe Masson		Aventures	Ados	Beaux Livres
00007	Dans les coulisses du...	Kate Atkinson		Roman	Tous publics	Littérature étrangère
00003	Et je danse aussi	Anne Laure Bondou		Comédie	Tous publics	Littérature française
00019	Guide Vert - Îles Cana...		Guide Vert	Voyages	Tous publics	Voyages
00020	Guide Vert - Irlande		Guide Vert	Voyages	Tous publics	Voyages
00008	Histoire-dujuif-erant	Jean-d'Omesson		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 Adamab...	Policier	Adultes	Policiers français étra...

Catastrophes au Brésil de Philippe Masson

Id	DateCommande	NbExemplaire	Etat	Montant
00002	21/01/2024	2	réglée	2
00003	20/01/2024	2	réglée	55
00004	18/10/2023	15	livrée	15
00009	08/03/2023	27	livrée	4

Numéro de commande :   
Date de commande :   
Montant :   
Nombre d'exemplaires :   
Numéro du livre :   
Etat de la commande :   
Les commandes livrées ne peuvent pas être supprimées  
Ajouter Modifier Supprimer  
Annuler Valider

## Vue coté code.

La vue respecte la même logique que pour les livres, DVD et revues. La nouveauté est que la DataGridView dépend d'une autre pour l'affichage.

La méthode événementielle `SelectedIndexChanged()` de la DataGridView de recherche appelle le chargement de celle des commandes. La méthode `SelectedIndexChanged()` du groupe 'Commandes' charge les informations détaillées.

```

/// <summary>
/// Sur la sélection d'une ligne ou cellule dans le grid
/// affichage des commandes du livres
/// </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)bdgLivresComListe.List[bdgLivresComListe.Position];
            AfficheLivresCommandeInfos(livre);
            txbLivresComNumLivre.Text = livre.Id;
        }
        catch
        {
            VideLivresComZones();
        }
    }
    else
    {
        txbLivresComNumLivre.Text = "";
        VideLivresComInfos();
    }
}

```

```

/// <summary>
/// Récupère et affiche les commandes d'un livre
/// </summary>
/// <param name="livre"></param>
private void AfficheLivresCommandeInfos(Livre livre)
{
    string idLivre = livre.Id;
    VideLivresComInfos();
    lesCommandes = controller.GetCommandesLivres(idLivre);
    grpLivresCommandes.Text = livre.Titre + " de " + livre.Auteur;
    if (lesCommandes.Count == 0)
        VideLivresComInfos();
    RemplirLivresComListeCommandes(lesCommandes);
}

```

```

/// <summary>
/// Remplit le datagrid avec la liste reçue en paramètre
/// </summary>
/// <param name="livres">liste de livres</param>
private void RemplirLivresComListeCommandes(List<CommandeDocument> LesCommandes)
{
    if (LesCommandes.Count > 0)
    {
        bdgLivresComListeCommande.DataSource = LesCommandes;
        dgvLivresComListeCom.DataSource = bdgLivresComListeCommande;
        dgvLivresComListeCom.Columns["idLivreDvd"].Visible = false;
        dgvLivresComListeCom.Columns["idSuivi"].Visible = false;
        dgvLivresComListeCom.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
        dgvLivresComListeCom.Columns["id"].DisplayIndex = 0;
        dgvLivresComListeCom.Columns["dateCommande"].DisplayIndex = 1;
        dgvLivresComListeCom.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    }
    else
    {
        dgvLivresComListeCom.Columns.Clear();
    }
}

```

En cours de modifications ou d'ajout, il n'est plus possible de modifier l'affichage des DataGridView. Les commandes sont triées en fonction du titre de la colonne sélectionnée.

Si les commandes ne sont pas nulles et qu'un livre est sélectionné, elles sont triées. RemplirLivresComListeCommandes affiche les commandes dans la DataGridView.

```

private void dgvLivresComListe_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)
{
    VideLivresComZones();
    string titreColonne = dgvLivresComListe.Columns[e.ColumnIndex].HeaderText;
    List<Livre> sortedList = new List<Livre>();
    switch (titreColonne)
    {
        case "Id":
            sortedList = lesLivresCom.OrderBy(o => o.Id).ToList();
            break;
        case "Titre":
            sortedList = lesLivresCom.OrderBy(o => o.Titre).ToList();
            break;
        case "Collection":
            sortedList = lesLivresCom.OrderBy(o => o.Collection).ToList();
            break;
        case "Auteur":
            sortedList = lesLivresCom.OrderBy(o => o.Auteur).ToList();
            break;
        case "Genre":
            sortedList = lesLivresCom.OrderBy(o => o.Genre).ToList();
            break;
        case "Public":
            sortedList = lesLivresCom.OrderBy(o => o.Public).ToList();
            break;
        case "Rayon":
            sortedList = lesLivresCom.OrderBy(o => o.Rayon).ToList();
            break;
    }
    RemplirLivresComListe(sortedList);
}

```

```

private void RemplirLivresComListeCommandes(List<CommandeDocument> LesCommandes)
{
    if (LesCommandes.Count > 0)
    {
        bdgLivresComListeCommande.DataSource = LesCommandes;
        dgvLivresComListeCom.DataSource = bdgLivresComListeCommande;
        dgvLivresComListeCom.Columns["idLivreDvd"].Visible = false;
        dgvLivresComListeCom.Columns["idSuivi"].Visible = false;
        dgvLivresComListeCom.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
        dgvLivresComListeCom.Columns["id"].DisplayIndex = 0;
        dgvLivresComListeCom.Columns["dateCommande"].DisplayIndex = 1;
        dgvLivresComListeCom.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    }
    else
    {
        dgvLivresComListeCom.Columns.Clear();
    }
}

```



```

private void btnLivresComSupprimer_Click(object sender, EventArgs e)
{
    CommandeDocument commandeDocument = (CommandeDocument)bdgLivresComListeCommande[bdgLivresComListeCommande.Position];
    if (dgvLivresComListeCom.CurrentCell != null && txbLivresComNbCommande.Text != "")
    {
        if (commandeDocument.IdSuivi > 2)
            MessageBox.Show("Une commande livrée ou réglée ne peut être supprimée");
        else if (MessageBox.Show("Êtes vous sûr de vouloir supprimer la commande n°" + commandeDocument.Id +
            " concernant " + lesLivresCom.Find(o => o.Id == commandeDocument.IdLivreDvd).Titre + " ?",
            "Validation suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            if (controller.SupprimerLivreDvdCom(commandeDocument))
            {
                Thread.Sleep(50);
                try
                {
                    Livre livre = (Livre)bdgLivresComListe.List[bdgLivresComListe.Position];
                    AfficheLivresCommandeInfos(livre);
                    txbLivresComNumLivre.Text = livre.Id;
                }
                catch
                {
                    VideLivresComZones();
                }
            }
            else
            {
                MessageBox.Show("Erreur");
            }
        }
    }
}

```

Communication avec l'API.

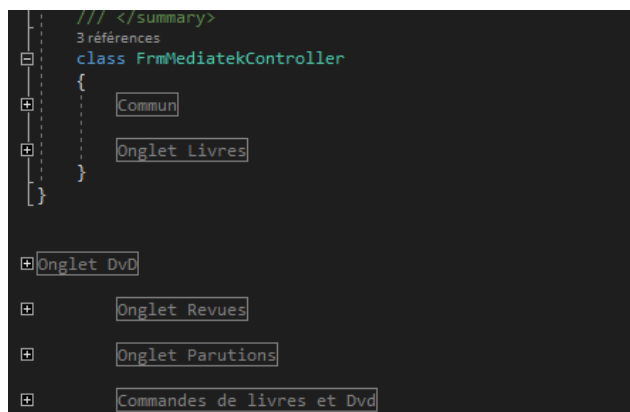
Des méthodes sont ajoutées pour effectuer les requêtes et récupérer les données. Les classes CommandeDocument et Abonnement étant des entités composées, chaque requête POST, PUT, DELETE doit être décomposée dans l'API.

```

private void AfficheDvdCommandeInfos(Dvd dvd)
{
    string idDvd = dvd.Id;
    VideDvdComInfos();
    lesCommandesDvd = controller.GetCommandesLivres(idDvd);
    grpDvdCommandes.Text = dvd.Titre + " de " + dvd.Realisateur;
    if (lesCommandes.Count == 0)
        VideDvdComInfos();
    RemplirDvdComListeCommandes(lesCommandesDvd);
}

```

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



Dans l'API.

```

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']);
            case "utilisateur" :

```

```

public function selectCommandesDocument($idLivreDvd){
    $param = array(
        "idLivreDvd" => $idLivreDvd
    );
    $req = "Select cd.id, c.dateCommande, c.montant, cd.nbExemplaire, cd.idLivreDvd, ";
    $req .= "cd.idsuivi, s.etat ";
    $req .= "from commandedocument cd join commande c on cd.id=c.id ";
    $req .= "join suivi s on cd.idsuivi=s.id ";
    $req .= "where cd.idLivreDvd = :idLivreDvd ";
    $req .= "order by c.dateCommande DESC";
    return $this->conn->query($req, $param);
}

```

```

public function selectAbonnementsRevue($idRevue){
    $param = array(
        "idRevue" => $idRevue
    );
    $req = "Select a.id, c.dateCommande, c.montant, a.dateFinAbonnement, a.idRevue ";
    $req .= "from abonnement a join commande c on a.id=c.id ";
    $req .= "where a.idRevue = :idRevue ";
    $req .= "order by c.dateCommande DESC";
    return $this->conn->query($req, $param);
}

```

Les classes C# CommandeDocument et Abonnement étant des entités composées dans la base de données. Chacune des requêtes POST, PUT, DELETE, les concernant doivent être décomposées dans l'API.

Exemple avec POST.

```

public function post($table, $champs){
    $result = null;
    if ($table == "livre"){
        $result = $this->accessBDD->insertLivre($champs);
    }elseif ($table == "dvd"){
        $result = $this->accessBDD->insertDvd($champs);
    }elseif ($table == "revue"){
        $result = $this->accessBDD->insertRevue($champs);
    }elseif ($table == "commandedocument"){
        $result = $this->accessBDD->insertCommande($champs);
    }elseif ($table == "abonnement"){
        $result = $this->accessBDD->insertAbonnement($champs);
    }else{
        $result = $this->accessBDD->insertOne($table, $champs);
    }
    if ($result == null || $result == false){
        $this->reponse(400, "requete invalide");
    }else{
        $this->reponse(200, "OK");
    }
}
}

```

```

public function insertCommande($champs)
{
    $champsCommande = [ "id" => $champs["Id"], "dateCommande" => $champs["DateCommande"],
        "montant" => $champs["Montant"] ];
    $champsCommandeDocument = [ "id" => $champs["Id"], "nbExemplaire" => $champs["NbExemplaire"],
        "idLivreDvd" => $champs["IdLivreDvd"], "idsuivi" => $champs["IdSuivi"] ];
    $result = $this->insertOne("commande", $champsCommande);
    if ($result == null || $result == false){
        return null;
    }
    return $this->insertOne("commandedocument", $champsCommandeDocument);
}

```

```

public function insertAbonnement($champs)
{
    $champsCommande = [ "id" => $champs["Id"], "dateCommande" => $champs["DateCommande"],
        "montant" => $champs["Montant"]];
    $champsAbonnement = [ "id" => $champs["Id"], "dateFinAbonnement" => $champs["DateFinAbonnement"],
        "idRevue" => $champs["IdRevue"]];
    $result = $this->insertOne("commande", $champsCommande);
    if ($result == null || $result == false){
        return null;
    }
    return $this->insertOne( "abonnement", $champsAbonnement);
}

```

selectMaxCommande permet d'obtenir le plus grand numéro de commande. Utilisée pour la création de CommandeDocuments et d'abonnements.

```

public function selectMaxCommande(){
    $req = "Select MAX(id) AS id FROM commande";
    return $this->conn->query($req);
}

```

Une commande ne peut être créée avec le suivi "en cours". Il n'est pas possible de statuer une commande comme "réglée" si elle n'a pas le statut "livré".

```

private void btnDvdComAjouter_Click(object sender, EventArgs e)
{
    EnCoursModifDvdCom(true);
    txbDvdComNumLivre.ReadOnly = true;
    ajouterBool = true;
    string id = PlusUnIdString(controller.GetNbCommandeMax());
    if (id == "1")
        id = "00001";
    VideDvdComInfos();
    cbxDvdComEtat.SelectedIndex = 0;
    txbDvdComNbCommande.Text = id;
    cbxDvdComEtat.Enabled = false;
}

/// <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 != "")
    {
        List<Suivi> lesSuivi = controller.GetAllSuivis().FindAll(o => o.Id >= ((Suivi)cbxDvdComEtat.SelectedItem).Id).ToList();
        if (lesSuivi.Count > 2)
            lesSuivi = lesSuivi.FindAll(o => o.Id < 4).ToList();
        EnCoursModifDvdCom(true);
        RemplirComboSuivi(lesSuivi, bdgDvdComEtat, cbxDvdComEtat);
        cbxDvdComEtat.SelectedIndex = 0;
    }
    else
    {
        MessageBox.Show("Aucune commande sélectionné");
    }
}
}

```

Spécificités Abonnement.

Gestion des documents de la médiathèque

DVD

Revue

Parutions des revues

Commandes de livres

Commandes de Dvd

Abonnements

\*Rechercher

Saisir le titre ou la partie d'un titre :

Revue dont un abonnement se termine dans moins de 30 jours

le genre :

X

Saisir un numéro de document :

Filtrer

le public :

X

Rechercher

le rayon :

X

Id	Titre	Periodicite	DelaiMiseADispo	Genre	Public	Rayon
10002	Alternatives Economi...	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	Beaux Livres
10013	heyyy 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
10008	L'Obs	HB	26	Actualités	Adultes	Magazines

Alternatives Economiques

Filtrer les abonnements se terminant dans moins de 30 jours :

Filtrer

Id	DateCommande	DateFinAbonnement	Montant
00023	20/11/2023	31/01/2024	21.2
00024	13/03/2023	09/02/2024	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

Ajouter

Modifier

Supprimer

Annuler

Valider

Création d'une alerte pour les abonnements se terminant dans moins de 30 jours.

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

Vérification qu'aucun abonnement n'est en cours avant suppression.

```

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

```

La méthode VerrifExemplaireAbo renvoie un booléen indiquant si un exemplaire de la revue a été acquis pendant la période de validité de l'abonnement. Si la méthode renvoie vrai, l'abonnement ne peut pas être supprimé.




```
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.CurrentCell != 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");
            }
        }
    }
}
}
```

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

### Mission 3 : gérer le suivi de l'état des exemplaires #4

Edit   ... 

 Open  Codeuraxe/MediathekDocuments  Private

 Codeuraxe opened now

edited by Codeuraxe · Edits ...

Agrandir la fenêtre en hauteur.

Dans l'onglet Livres, partie basse, ajouter la liste des exemplaires du livre sélectionné. Cette liste d'exemplaires doit contenir les colonnes suivantes : numéro d'exemplaire, date achat et libellé de l'état. La liste doit être triée par date d'achat, dans l'ordre inverse de la chronologie, et le clic sur une colonne doit permettre le tri sur la colonne. Sur la sélection d'un exemplaire, il doit être possible de changer son état.

Le principe est le même pour les DVD.

Dans l'onglet "Parutions des revues", liste des parutions, remplacer la colonne "Photo" par "Etat". Permettre aussi le changement d'état. Permettre de supprimer un exemplaire.



Assignees

No one - [Assign yourself](#)

Labels

No labels

Projects

 @Codeuraxe's untitled project

Status In Progress

Priority

Filter options



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

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		Aventures	Ados	Beaux Livres
00007	Dans les coulisses d...	Kate Atkinson		Roman	Tous publics	Littérature étrangère
00003	Et je danse aussi	Anne Laure Bondou		Comédie	Tous publics	Littérature française
00019	Guide Vert - Iles Can...		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'armée furieuse	Fred Vargas	Commissaire Adamsb...	Policier	Adultes	Policiers français étra...

Informations détaillées

Numéro de document :
00017
Code ISBN :

Titre :
Catastrophes au Brésil

Auteur(e) :
Philippe Masson

Collection :

Genre :
Aventures

Public :
Ados

Rayon :
Beaux Livres

Chemin de l'image :

Image :

Ajouter
Modifier
Supprimer
Annuler
Valider

Les méthodes pour consulter les exemplaires existent déjà dans l'application (classe côté C#, méthode `selectExemplairesRevue` dans l'API). Nous avons créé de nouvelles variables globales pour l'interface afin de gérer ces consultations plus efficacement.

```
#region Onglet Livres
private readonly BindingSource bdgLivresListe = new BindingSource();
private List<Livre> lesLivres = new List<Livre>();
private readonly BindingSource bdgGenresInfo = new BindingSource();
private readonly BindingSource bdgPublicsInfo = new BindingSource();
private readonly BindingSource bdgRayonsInfo = new BindingSource();
private readonly BindingSource bdgLivresListeEx = new BindingSource();
private List<Exemplaire> lesExemplairesLivres = new List<Exemplaire>();
```

Le `DataGridView` des exemplaires est automatiquement rempli ou vidé lorsqu'un livre est sélectionné dans le `DataGridView` affichant les livres. La méthode `AfficheLivresInfo` est appelée lorsque un livre est sélectionné, permettant une mise à jour dynamique et immédiate des informations.

```

private void AfficheLivresInfos(Livre livre)
{
    VideLivresEx();
    txbLivresAuteur.Text = livre.Auteur;
    txbLivresCollection.Text = livre.Collection;
    txbLivresImage.Text = livre.Image;
    txbLivresIsbn.Text = livre.Isbn;
    txbLivresNumero.Text = livre.Id;
    cbxLivresGenresInfo.SelectedIndex = cbxLivresGenresInfo.FindString(livre.Genre);
    cbxLivresPublicInfo.SelectedIndex = cbxLivresPublicInfo.FindString(livre.Public);
    cbxLivresRayonInfo.SelectedIndex = cbxLivresRayonInfo.FindString(livre.Rayon);
    txbLivresTitre.Text = livre.Titre;
    lesExemplairesLivres = controller.GetExemplairesRevue(livre.Id);
    gbxLivresEx.Text = "Les exemplaire de " + livre.Titre;
    RemplirLivresListeExemplaire(lesExemplairesLivres);
    string image = livre.Image;
    try
    {
        pcbLivresImage.Image = Image.FromFile(image);
    }
    catch
    {
        pcbLivresImage.Image = null;
    }
}

```

```

private void RemplirLivresListeExemplaire(List<Exemplaire> exemplaires)
{
    if (exemplaires.Count > 0)
    {
        bdgLivresListeEx.DataSource = exemplaires;
        dgvLivresListeEx.DataSource = bdgLivresListeEx;
        dgvLivresListeEx.Columns["photo"].Visible = false;
        dgvLivresListeEx.Columns["id"].Visible = false;
        dgvLivresListeEx.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
        dgvLivresListeEx.Columns["id"].DisplayIndex = 0;
        dgvLivresListeEx.Columns["numero"].DisplayIndex = 1;
        dgvLivresListeEx.Columns["dateAchat"].DefaultCellStyle.Format = "d/M/yyyy";
        dgvLivresListeEx.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
        //change les idEtats en la valeur de leur libelle
        for (int i = 0; i < dgvLivresListeEx.RowCount; i++)
        {
            if (int.TryParse(dgvLivresListeEx.Rows[i].Cells["idEtat"].Value.ToString(), out _))
            {
                dgvLivresListeEx.Rows[i].Cells["idEtat"].Value = lesEtatsEx.First(o => o.Id == dgvLivresListeEx.Rows[i].Cells["idEtat"].Value.ToString());
            }
        }
    }
    else
    {
        dgvLivresListeEx.Columns.Clear();
    }
}

```

Les détails d'une commande sont affichés lorsqu'une commande est sélectionnée dans le tableau des commandes. Ce fonctionnement est identique pour les onglets DVD et revues, assurant une cohérence et une facilité d'utilisation à travers les différentes sections de l'application.

Le GroupBox dans la vue côté formulaire a été copié et adapté des livres aux onglets DVD et revue, permettant une uniformité dans la présentation et l'interaction.

```

private void dgvLivresListeEx_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)
{
    VidLivresExemplaire();
    Livre livre = (Livre)bdgLivresListe.List[bdgLivresListe.Position];
    string titreColonne = dgvLivresListeEx.Columns[e.ColumnIndex].HeaderText;
    List<Exemplaire> sortedList = controller.GetExemplairesRevue(livre.Id);

    switch (titreColonne)
    {
        case "Id":
            sortedList = lesExemplairesLivres.OrderBy(o => o.Id).ToList();
            break;
        case "DateAchat":
            sortedList = lesExemplairesLivres.OrderBy(o => o.DateAchat).ToList();
            break;
        case "IdEtat":
            sortedList = lesExemplairesLivres.OrderBy(o => o.IdEtat).ToList();
            break;
    }
    RemplirLivresListeExemplaire(sortedList);
}

```

Deux nouveaux boutons ont été ajoutés pour permettre la modification et la suppression des exemplaires. Lorsqu'une modification est en cours, le libellé et la fonction des boutons changent, indiquant clairement l'état de l'action en cours.

Un booléen global, `modifEtat`, a été introduit pour alterner les fonctions des deux boutons. Lorsque le bouton de modification est activé, il valide d'abord si `modifEtat` est vrai. Sinon, la modification est lancée. Cette logique assure que les actions sont correctement suivies et mises en œuvre de manière cohérente.

La méthode activée par le clic du deuxième bouton suit une logique similaire, assurant que toutes les modifications et suppressions sont effectuées en respectant les mêmes règles de validation.

Ajouter Supprimer

Quand la modification est en cours le libellé et la fonction des boutons change. Un booléen déclaré en globale, `modifEtat` permet d'alterner les fonctions des deux boutons. En dessous, la méthode activée par le bouton « modifier » sert en premier lieu à valider, si `modifEtat` est vrai, sinon la modification est lancée.

```

private void BtnModifierLivres_Click(object sender, EventArgs e)
{
    EnCoursModifLivres(true);
}

/// <summary>
/// démarre la procédure de suppression de livre
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnSupprimerLivres_Click(object sender, EventArgs e)
{
    Livre leLivre = (Livre)bdgLivresListe.List[bdgLivresListe.Position];
    if (MessageBox.Show("Etes vous sur de vouloir supprimer" + leLivre.Titre + " de " + leLivre.Auteur + " ?",
        "Validation suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        // fonction a modifier pour prendre en charge le fait que l'on ne pourra pas supprimer un livre tant q
        if (controller.SupprimerLivre(leLivre))
        {
            Thread.Sleep(100);
            lesLivres = controller.GetAllLivres();
            RemplirLivresListeComplete();
        }
        else
        {
            MessageBox.Show("Erreur");
        }
    }
}
}

```

Deux nouveaux boutons ont été ajoutés pour permettre la modification et la suppression des exemplaires. Lorsqu'une modification est en cours, le libellé et la fonction des boutons changent, indiquant clairement l'état de l'action en cours.

Un booléen global, `modifEtat`, a été introduit pour alterner les fonctions des deux boutons. Lorsque le bouton de modification est activé, il valide d'abord si `modifEtat` est vrai. Sinon, la modification est lancée. Cette logique assure que les actions sont correctement suivies et mises en œuvre de manière cohérente.

La méthode activée par le clic du deuxième bouton suit une logique similaire, assurant que toutes les modifications et suppressions sont effectuées en respectant les mêmes règles de validation.

```

private void BtnSupprimerLivresEx_Click(object sender, EventArgs e)
{
    if (modifEtat)
    {
        modifEtat = false;
        btnSupprimerLivresEx.Text = "Supprimer";
        btnModifierLivresEx.Text = "Modifier";
        cbxLivresExEtat.Enabled = false;
        EnCoursModifLivres(false);
        grpLivresInfos.Enabled = true;
    }
    else
    {
        if (dgvLivresListeEx.CurrentRow != null)
        {
            if (MessageBox.Show("Etes vous de supprimer l'exemplaire " + txtLivresNbEx.Text + " ? ", "oui ?", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                int numero = int.Parse(txtLivresNbEx.Text);
                DateTime dateAchat = dtpLivresDateAchatEx.Value;
                string photo = txtLivresPhotoEx.Text;
                string idEtat = ((Etat)cbxLivresExEtat.SelectedItem).Id;
                string iDocument = txtLivresExId.Text;
                Exemplaire exemplaire = new Exemplaire(numero, dateAchat, photo, idEtat, iDocument);
                controller.SupprimerExemplaire(exemplaire);
                lesExemplairesLivres = controller.GetExemplairesRevue(iDocument);
                RemplirLivresListeExemplaire(lesExemplairesLivres);
                Thread.Sleep(20);
            }
        }
        else
        {
            MessageBox.Show("Aucun exemplaire sélectionné");
        }
    }
}

```

## Les États.

Une nouvelle classe Etat a été créée pour gérer les différents états des exemplaires. Une méthode dans Access récupère les états, et comme la classe Etat en C# est identique à celle de la base de données, aucune modification supplémentaire n'est nécessaire dans l'API.

Etat de l'exemplaire :

```

public List<Etat> GetAllEtats()
{
    IEnumerable<Etat> lesEtats = TraitementRecup<Etat>(GET, "etat");
    return new List<Etat>(lesEtats);
}

```

Pour les modifications d'exemplaires, étant donné qu'un exemplaire est composé de deux clés primaires, l'API a été modifiée pour que les requêtes PUT (modification) puissent prendre en compte ces deux clés.

```

public function updateOne($table, $id, $champs, $numero = null){
    if($this->conn != null && $champs != null){
        // construction de la requête
        $requete = "update $table set ";
        foreach ($champs as $key => $value){
            $requete .= "$key=: $key,";
        }
        // (enlève la dernière virgule)
        $requete = substr($requete, 0, strlen($requete)-1);
        $champs["id"] = $id;
        $requete .= " where id=:id;";
        if($numero != null)
        {
            $requete = substr($requete, 0, strlen($requete)-1);
            $champs["numero"] = $numero;
            $requete .= " and numero=:numero;";
        }
        return $this->conn->execute($requete, $champs);
    }else{
        return null;
    }
}

```

Des ajouts ont été faits dans FrmController pour intégrer ces nouvelles fonctionnalités et assurer que toutes les modifications et suppressions respectent les règles définies.

```

public bool UpdateExemplaire(Exemplaire exemplaire)
{
    return access.UpdateEntite("exemplaire", exemplaire.Id, JsonConvert.SerializeObject(exemplaire, new CustomDateTimeConverter()));
}

/// <summary>
/// Supprime un exemplaire de livre de la base de données.
/// </summary>
/// <param name="exemplaire">Exemplaire à supprimer.</param>
/// <returns>Vrai si l'opération est réussie.</returns>
public bool SupprimerExemplaire(Exemplaire exemplaire)
{
    return access.SupprimerEntite("exemplaire", JsonConvert.SerializeObject(exemplaire, new CustomDateTimeConverter()));
}

/// <summary>
/// Crée un nouveau livre dans la base de données.
/// </summary>
/// <param name="livre">Livre à créer.</param>
/// <returns>Vrai si l'opération est réussie.</returns>
public bool CreerLivre(Livre livre)
{
    return access.CreerEntite("livre", JsonConvert.SerializeObject(livre));
}

```

Mission 4 : mettre en place des authentifications :

Mission 4 : mettre en place des authentifications #5

Open

Codeuraxe/MediathekDocuments

Private

Codeuraxe opened 6 hours ago

edited by Codeuraxe · Edits · ...

Dans la base de données, ajouter une table Utilisateur et une table Service, sachant que chaque utilisateur ne fait partie que d'un service.  
Pour réaliser les tests, remplir les tables d'exemples.  
Ajouter une première fenêtre d'authentification. Faire en sorte que l'application démarre sur cette fenêtre.  
Suivant le type de personne authentifiée, empêcher certains accès en rendant invisibles ou inactifs certains onglets ou objets graphiques.  
Dans le cas du service Culture qui n'a accès à rien, afficher un message précisant que les droits ne sont pas suffisants pour accéder à cette application, puis fermer l'application.  
Faire en sorte que l'alerte de fin d'abonnement n'apparaisse que pour les personnes concernées (qui gèrent les commandes).

Assignees

No one - [Assign yourself](#)

Labels

No labels

Projects

@Codeuraxe's Mediatek Documents

Status In Progress

La classe Utilisateur a été créée sans nécessiter d'affichage, donc elle n'a pas besoin d'instancier un service ni de disposer d'une méthode toString().

```
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.IdService = idService;
    this.Service = service;
}
public string Id { get; }
public string Nom { get; set; }
public string Prenom { get; set; }
public string Mail { get; set; }
public string IdService { get; set; }
public string Service { get; set; }
```

L'application est désormais lancée par FrmLogin, l'interface de connexion. Les utilisateurs peuvent se connecter en utilisant soit leur nom, soit leur adresse mail.

Identifiant

Password

Connection

```

public function selectUtilisateur($champs)
{
    $param = array(
        "mail" => $champs["mail"],
        "password" => $champs["password"]
    );
    $req = "Select u.id, u.nom, u.prenom, u.mail, u.idservice, s.libelle as service ";
    $req .= "from utilisateur u join service s on u.idservice=s.id ";
    $req .= "where u.mail = :mail ";
    $req .= "and u.password = :password ";
    $req .= "or u.nom = :mail ";
    $req .= "and u.password = :password";
    return $this->conn->query($req, $param);
}

```

FrmLogin instancie FrmLoginController, le contrôleur de connexion. Si les informations d'identification de l'utilisateur et le mot de passe sont valides, la fenêtre d'authentification se ferme, et l'utilisateur est dirigé vers la vue principale.

```

private FrmLoginController controller;
public FrmLogin()
{
    InitializeComponent();
    Init();
}

/// <summary>
/// Initialisations :
/// Création du contrôleur
/// </summary>
private void Init()
{
    txtLogin.Text = "";
    txtPwd.Text = "";
    controller = new FrmLoginController();
}

```



```

private void BtnConnec_Click(object sender, EventArgs e)
{
    if (controller.GetLogin(txbLogin.Text, txbPwd.Text))
        this.Visible = false;
    else
    {
        MessageBox.Show("Mauvais mot de passe ou login utilisateur");
        txbLogin.Text = "";
        txbPwd.Text = "";
    }
}

private void txbLogin_TextChanged(object sender, EventArgs e)
{
}

private void FrmLogin_Load(object sender, EventArgs e)
{

```

```

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

/// <summary>
/// Authentifie l'utilisateur en utilisant son email et mot de passe.
/// </summary>
/// <param name="mail">Email de l'utilisateur.</param>
/// <param name="password">Mot de passe de l'utilisateur.</param>
/// <returns>Vrai si l'authentification est réussie.</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;
}

```

La sécurité est renforcée avec l'importation de System.Security.Cryptography pour utiliser les fonctions d'encodage. Lors d'une tentative de connexion, si le hash et le salt du mot de passe fourni par l'utilisateur correspondent à ceux stockés dans la base de données, l'application instancie la vue principale avec l'utilisateur en paramètre.

```
private static string GetHash(HashAlgorithm hashAlgorithm, string input)
{
    // Convertit le texte en tableau de bytes et calcule le hash.
    byte[] data = hashAlgorithm.ComputeHash(Encoding.UTF8.GetBytes(input));

    // Utilise un StringBuilder pour assembler les bytes du hash en texte.
    var sBuilder = new StringBuilder();

    // Parcourt chaque byte du hash et le convertit en chaîne hexadécimale.
    for (int i = 0; i < data.Length; i++)
    {
        sBuilder.Append(data[i].ToString("x2"));
    }

    // Retourne la chaîne hexadécimale.
    return sBuilder.ToString();
}
```

Une fonction a été créée pour encrypter le mot de passe et le retourner encodé en base64 au format string.

Les fonctions de hash étant validées, une méthode GetLogin a été ajoutée dans Access, appelée depuis FrmLoginController pour vérifier les informations de connexion.

```
public Utilisateur GetLogin(string mail, string hash)
{
    Dictionary<string, string> login = new Dictionary<string, string>
    {
        { "mail", mail },
        { "password", hash }
    };
    String mailHash = JsonConvert.SerializeObject(login);
    List<Utilisateur> utilisateurs = TraitementRecup<Utilisateur>(GET, "utilisateur/" + mailHash);
    if (utilisateurs.Count > 0)
        return utilisateurs[0];
    Log.Error("Access.GetLogin catch user fail connection :" + mail);
    return null;
}
```

FrmMediatek est instancié avec l'utilisateur connecté. Des méthodes sont implémentées pour restreindre l'accès en fonction des droits de l'utilisateur. Cela se fait soit à l'ouverture du programme, soit dans les méthodes TabOngletEnter().

```

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 modifEtat = false;
    private bool firstLoad = true;

    /// <summary>
    /// Constructeur : création du contrôleur lié à ce formulaire
    /// </summary>
    internal FrmMediatek(Utilisateur lutilisateur)
    {
        InitializeComponent();
        this.controller = new FrmMediatekController();
        this.utilisateur = lutilisateur;
        VerifDroitAccueil(lutilisateur);
    }
}

```

Si un utilisateur (son service) n'est pas autorisé à utiliser l'application, celle-ci se ferme automatiquement pour des raisons de sécurité.

```

private void VerifDroitAccueil(Utilisateur lutilisateur)
{
    if (!controller.VerifDroitAccueil(lutilisateur))
    {
        MessageBox.Show("Droits insuffisant");
        Application.Exit();
    }
}

```

Exemple avec TabLivres\_Enter : si l'utilisateur n'a pas les droits de modification, il ne pourra que consulter les informations. Sinon, il sera averti au premier lancement de l'application des abonnements arrivant à expiration.

```

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);
    RemplirComboCategorie(controller.GetAllGenres(), bdgGenresInfo, cbxLivresGenresInfo);
    RemplirComboCategorie(controller.GetAllPublics(), bdgPublicsInfo, cbxLivresPublicInfo);
    RemplirComboCategorie(controller.GetAllRayons(), bdgRayonsInfo, cbxLivresRayonInfo);
    RemplirComboEtat(controller.GetAllEtats(), bdgEtats, cbxLivresExEtat);
    lesEtatsEx = controller.GetAllEtats();
    modifEtat = false;
    RemplirLivresListeComplete();
    if (controller.VerifDroitModif(utilisateur))
    {
        EnCoursModifLivres(false);
        if (firstLoad)
        {
            Thread.Sleep(150);
            AfficherAlerteAbo();
            firstLoad = false;
        }
    }
    else
    {
        ConsultationLivres();
    }
}

```

Dans le contrôleur, les droits des différents services sont spécifiés en variables statiques. Les méthodes de vérification appelées depuis la vue fonctionnent toutes selon le même modèle. Pour les onglets permettant des commandes, si l'utilisateur a des droits insuffisants, il sera averti puis redirigé vers le premier onglet.

```

public bool VerifDroitAccueil(Utilisateur utilisateur)
{
    List<string> services = new List<string> { "compta", "biblio", "accueil" };
    return services.Contains(utilisateur.Service);
}

/// <summary>
/// Vérifie si le service de l'utilisateur permet la modification de données.
/// </summary>
/// <param name="utilisateur">Utilisateur à vérifier.</param>
/// <returns>Vrai si la modification est autorisée.</returns>
public bool VerifDroitModif(Utilisateur utilisateur)
{
    List<string> services = new List<string> { "biblio", "accueil" };
    return services.Contains(utilisateur.Service);
}

/// <summary>
/// Vérifie si l'utilisateur peut passer des commandes.
/// </summary>
/// <param name="utilisateur">Utilisateur à vérifier.</param>
/// <returns>Vrai si les commandes sont autorisées.</returns>
public bool VerifCommande(Utilisateur utilisateur)
{
    List<string> services = new List<string> { "biblio" };
    return services.Contains(utilisateur.Service);
}

```

```

private void TabLivresCom_Enter(object sender, EventArgs e)
{
    if (!controller.VerifCommande(utilisateur))
    {
        MessageBox.Show("Droits insuffisant");
        tabControl.SelectedIndex = 0;
    }
    else
    {
        lesLivresCom = controller.GetAllLivres();
        RemplirComboCategorie(controller.GetAllGenres(), bdgGenres, cbxLivresComGenres);
        RemplirComboCategorie(controller.GetAllPublics(), bdgPublics, cbxLivresComPublics);
        RemplirComboCategorie(controller.GetAllRayons(), bdgRayons, cbxLivresComRayons);
        RemplirComboSuivi(controller.GetAllSuivis(), bdgLivresComEtat, cbxLivresComEtat);
        RemplirComboEtat(controller.GetAllEtats(), bdgEtats, cbxLivresExEtat);
        EnCoursModifLivresCom(false);
        RemplirLivresComListeComplete();
    }
}

```

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

### Mission 5 : Tâche 1 : corriger des problèmes de sécurité #6

Edit   ... 

Open

Codeuraxe/MediatekDocuments

Private



Codeuraxe opened 3 minutes ago

edited by Codeuraxe · Edits · ...

Pour connaître l'intitulé des problèmes, voir le document [missions.pdf](#). Pour chaque problème, créer une "issue" dans le dépôt GitHub correspondant, affecter l'issue à un des développeurs (si vous êtes 2, vous ou votre collègue, sinon vous-même), créer une branche pour proposer une correction de code en expliquant la correction, faire un pull request, accepter le pull request si le résultat correspond bien aux attentes.

Aide pour le problème n°1

Récupérer le code de l'application Habilitations <https://github.com/CNED-SLAM/Habilitations> et observer le contenu de App.config (dans le dossier Habilitations) et de Access.php (dans Habilitations/dal) pour comprendre comment les informations de connexions ont été sécurisées en étant placées dans App.config puis récupérées dans Access.php. S'inspirer de cette technique pour mémoriser le couple "login:pwd" dans App.config au lieu de le laisser en dur directement dans Access.php.

Aide pour le problème n°2

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

Assignees

No one - [Assign yourself](#)

Labels

No labels

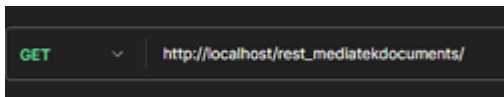
Projects

@Codeuraxe's Mediatek Documents

Status **In Progress**

Priority

Filter options



Pour éviter que les utilisateurs puissent accéder à l'ensemble des fichiers, une redirection de la racine vers une page d'erreur "404" a été mise en place dans le fichier htaccess d'Apache :

RewriteRule ^\$ mediatekdocuments.php?error=404

Du côté de l'application C#, les identifiants de connexion ont été déplacés dans le fichier app.config, lequel est ignoré par Git (gitignore) lors des changements de mot de passe à la mise en production. Ces identifiants sont appelés par la méthode GetAuthentificationString de la classe Access. Ainsi, lors de l'instanciation de Access, la variable de classe api est valorisée comme précédemment.

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

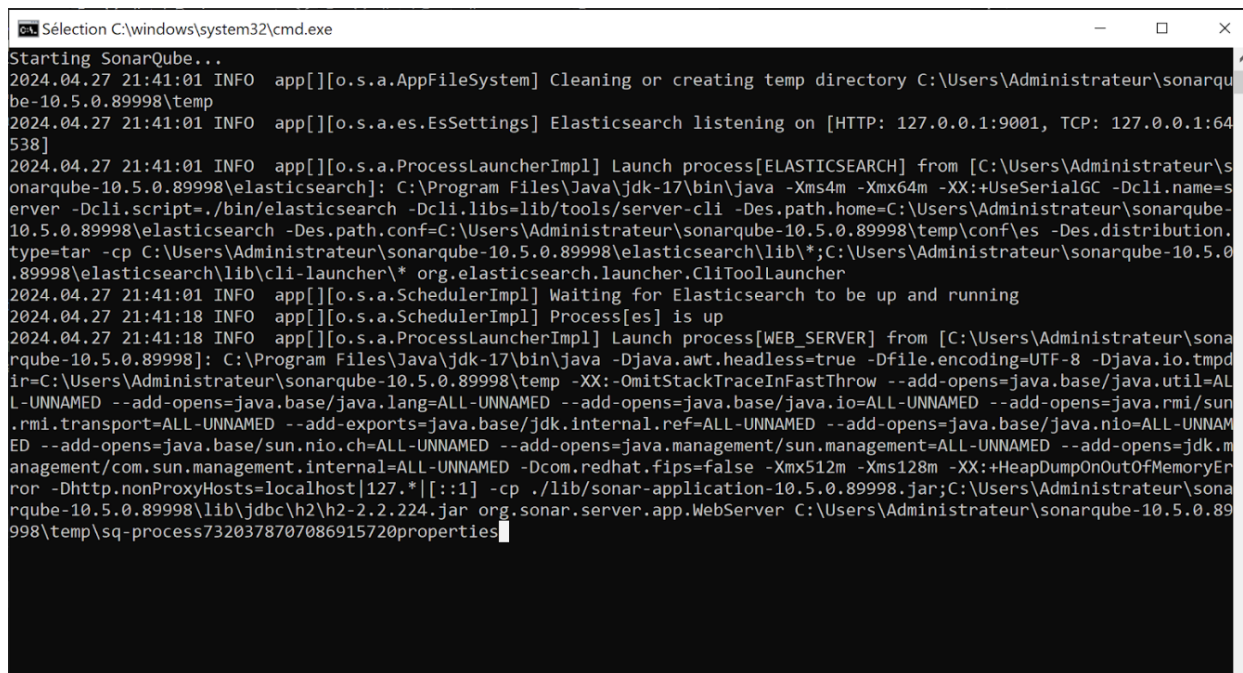
```

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);
    }
}

```

Contrôler la qualité.



```

C:\windows\system32\cmd.exe
Starting SonarQube...
2024.04.27 21:41:01 INFO app[[o.s.a.AppFileSystem] Cleaning or creating temp directory C:\Users\Administrateur\sonarqube-10.5.0.89998\temp
2024.04.27 21:41:01 INFO app[[o.s.a.es.EsSettings] Elasticsearch listening on [HTTP: 127.0.0.1:9001, TCP: 127.0.0.1:64538]
2024.04.27 21:41:01 INFO app[[o.s.a.ProcessLauncherImpl] Launch process[ELASTICSEARCH] from [C:\Users\Administrateur\sonarqube-10.5.0.89998\elasticsearch]: C:\Program Files\Java\jdk-17\bin\java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=./bin/elasticsearch -Dcli.libs=lib/tools/server-cli -Des.path.home=C:\Users\Administrateur\sonarqube-10.5.0.89998\elasticsearch -Des.path.conf=C:\Users\Administrateur\sonarqube-10.5.0.89998\temp\conf\es -Des.distribution.type=tar -cp C:\Users\Administrateur\sonarqube-10.5.0.89998\elasticsearch\lib\*;C:\Users\Administrateur\sonarqube-10.5.0.89998\elasticsearch\lib\cli-launcher\* org.elasticsearch.launcher.CliToolLauncher
2024.04.27 21:41:01 INFO app[[o.s.a.SchedulerImpl] Waiting for Elasticsearch to be up and running
2024.04.27 21:41:18 INFO app[[o.s.a.SchedulerImpl] Process[es] is up
2024.04.27 21:41:18 INFO app[[o.s.a.ProcessLauncherImpl] Launch process[WEB_SERVER] from [C:\Users\Administrateur\sonarqube-10.5.0.89998]: C:\Program Files\Java\jdk-17\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=C:\Users\Administrateur\sonarqube-10.5.0.89998\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/sun.management=ALL-UNNAMED --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED -Dcom.redhat.fips=false -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost[127.*][:1] -cp ./lib/sonar-application-10.5.0.89998.jar;C:\Users\Administrateur\sonarqube-10.5.0.89998\lib\jdbc\h2\h2-2.2.224.jar org.sonar.server.app.WebServer C:\Users\Administrateur\sonarqube-10.5.0.89998\temp\sq-process7320378707086915720properties

```

L'analyse du code est effectuée avec SonarQube. Un serveur local de SonarQube est lancé pour cette analyse. Grâce à une extension, les recommandations sont directement affichées dans Visual Studio 2019, permettant des ajustements en temps réel.

Mise en place des logs :

Un plan de tests, incluant les tests API, est disponible au téléchargement sur la page du projet. Des tests unitaires sont réalisés pour toutes les classes modèles, couvrant à la fois l'instanciation et les méthodes.

Par exemple, pour un livre, les tests unitaires valident l'instanciation et le bon fonctionnement des méthodes associées.

```
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 de Access. Les logs sont capturés lorsque le programme rencontre des erreurs, offrant une traçabilité et une capacité de diagnostic améliorée.

```
else
{
    Console.WriteLine("code erreur = " + code + " message = " + (String)retour["message"]);
    Log.Error("Access.TraitementRecup 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



Documentation technique.

Les documentations techniques détaillées sont disponibles sur la page de présentation du projet. Elles couvrent toutes les fonctionnalités et les modules de l'application, fournissant des instructions claires pour les développeurs et les utilisateurs finaux.

## Documentation

### Packages

#### Application

### Interfaces, Classes and Traits

#### AccessBDD

Classe de construction des requêtes SQL à envoyer à la BDD

#### ConnexionPDO

Classe de connexion et d'exécution des requêtes dans une BDD MySQL

#### Controle

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

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

Déploiement de l'API :

L'API est hébergée à l'adresse suivante : <https://mediatek-documents.site/>. Elle est hébergée sur un service web fiable, garantissant une disponibilité continue et une performance optimale.

Des configurations de sécurité supplémentaires ont été mises en place pour protéger l'API contre les accès non autorisés et les attaques potentielles.

Gestion des sauvegardes de données :

Les sauvegardes de la base de données sont configurées pour se faire de manière incrémentale chaque jour et de manière intégrale chaque semaine.

Les sauvegardes incrémentales permettent de sauvegarder uniquement les modifications apportées depuis la dernière sauvegarde, optimisant ainsi l'espace de stockage et la rapidité du processus.

Les sauvegardes intégrales, qui sont des copies complètes de la base de données, sont conservées pendant une période de 4 semaines, assurant une rétention suffisante pour une restauration en cas de besoin.

Une politique de rotation des sauvegardes a été mise en place pour garantir que les anciennes sauvegardes sont supprimées de manière sécurisée après la période de rétention.

## Bilan

Le temps investi dans le projet m'a permis de progresser et d'acquérir une nouvelle perspective sur la résolution des problèmes. J'ai pu aborder certains problèmes sous un nouvel angle et découvrir des approches plus efficaces pour les résoudre.

Les leçons tirées de ce projet m'ont permis d'améliorer mes compétences en développement, en gestion de projet et en résolution de problèmes. Je suis convaincu que ces expériences me seront bénéfiques pour les projets futurs, et je suis enthousiaste à l'idée de continuer à apprendre et à évoluer dans ce domaine.