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

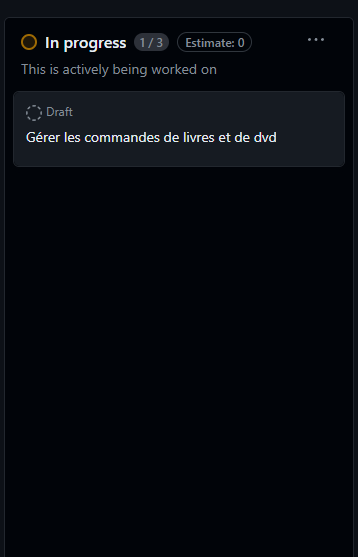
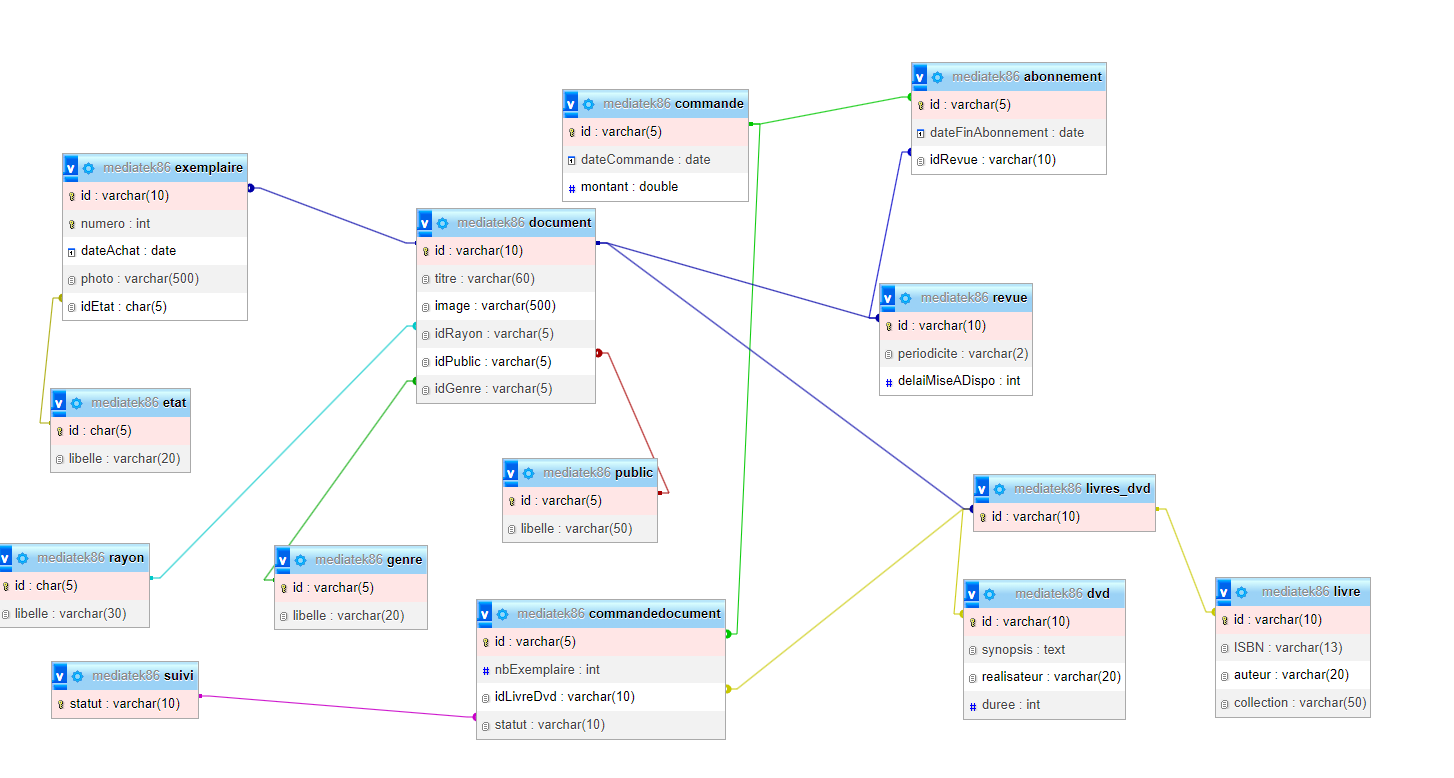


Diagramme UML de la tâche :

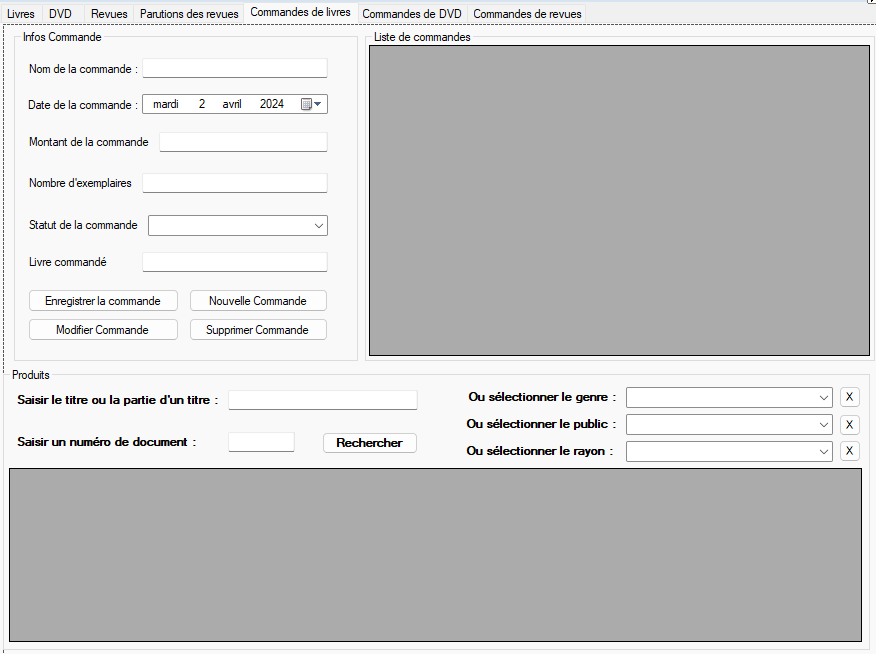
Une image contenant texte, diagramme, croquis, dessin

Description générée automatiquement

MCD de la tâche :



Maquette de la page :



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 à 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 :

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

L’ajout dans la vue :

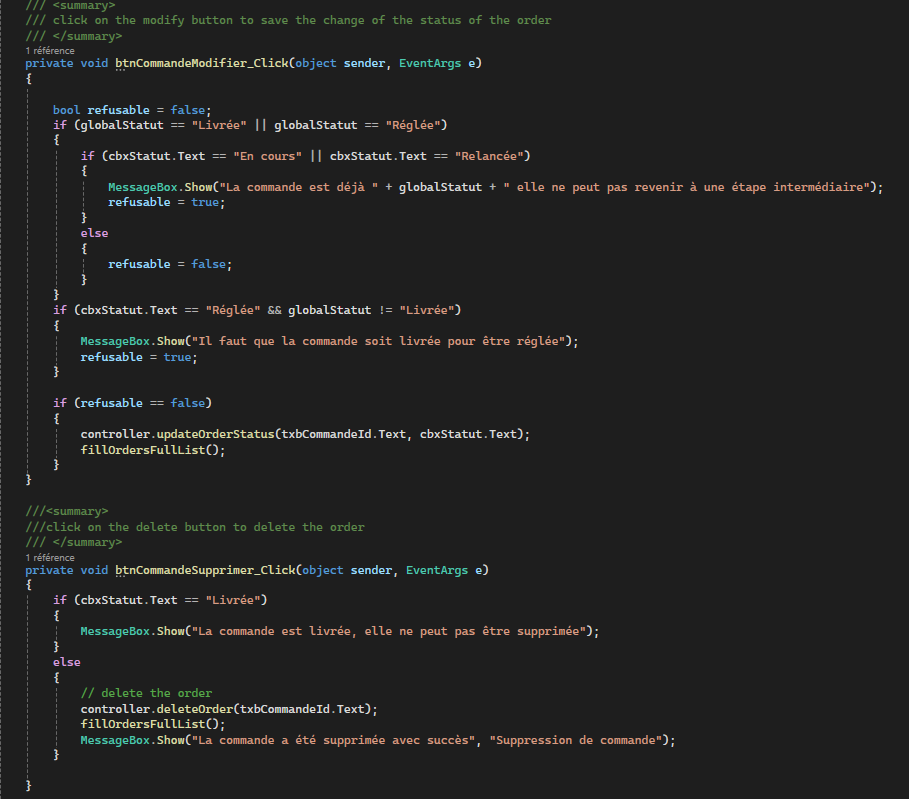
Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement

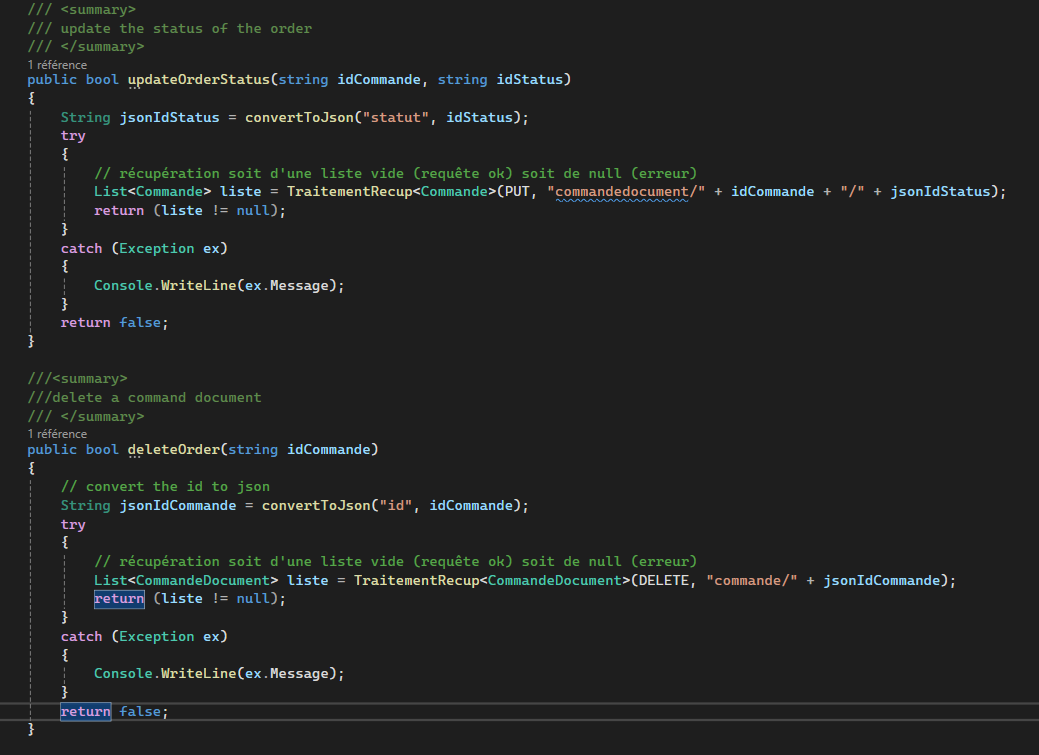
Pour cela l’api a été légèrement modifiée pour permettre l’ajout d’une commande (en premier) puis d’une commandedocument en une seule requête.

Une image contenant texte, capture d’écran

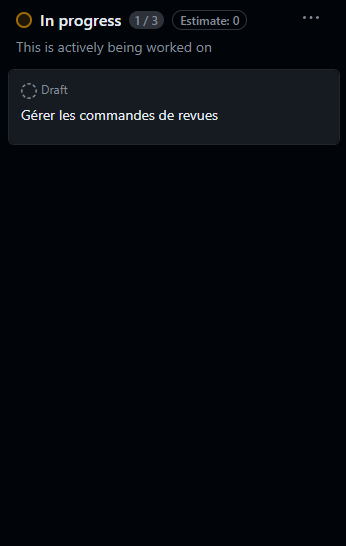
Description générée automatiquement

La modification et la supression dans la vue : 

Dans la classe access :



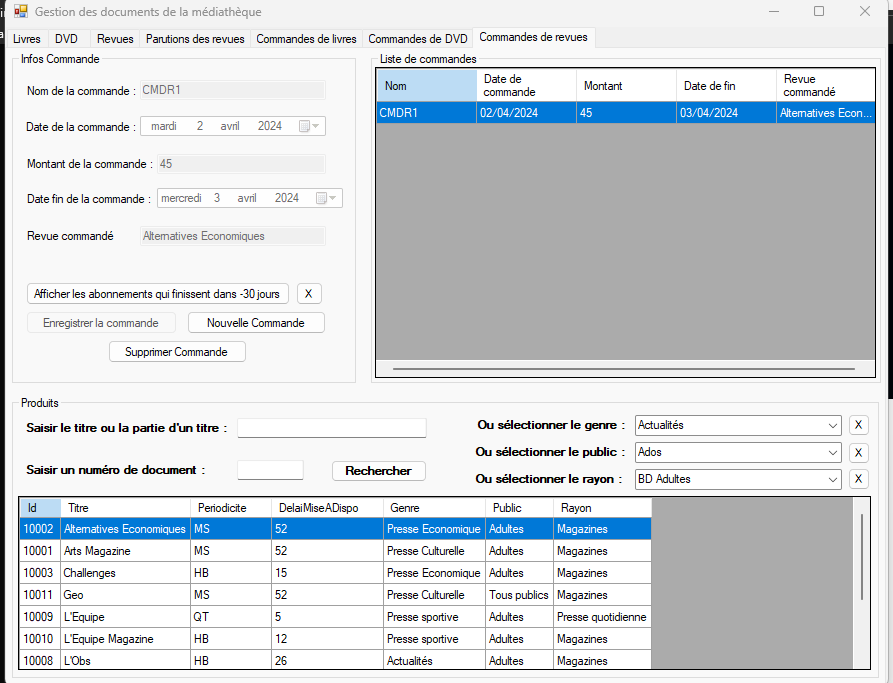
Le reste des méthodes sont des dérivés très proches des méthodes déjà définies.

1. Gérer les commandes de revues

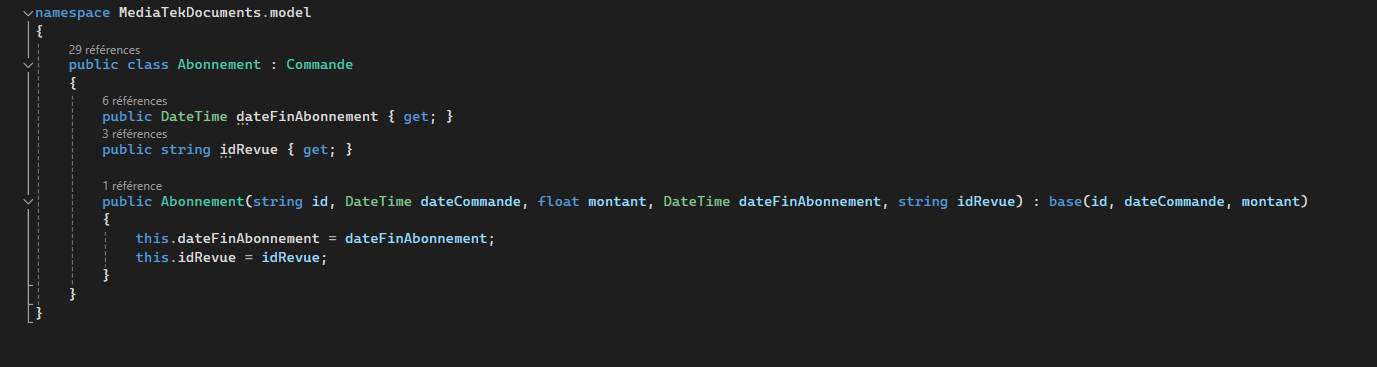
Temps estimé : 4h

Temps réel : 6h

Maquette de la page des revues :

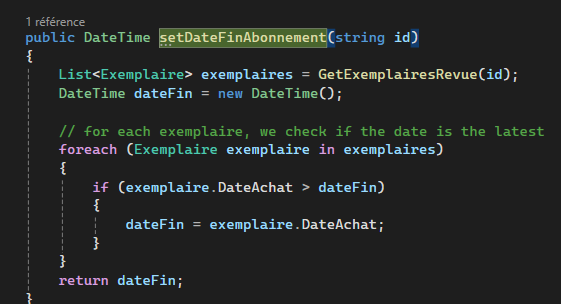


C’est à peu près le même principe que pour les commandes de livres et de dvd mais l’abonnement n’est pas modifiable et qu’il y a un bouton pour afficher les abonnements prenant fin dans – de 30 jours.

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

Methode d’ajout d’abonnement dans la vue :

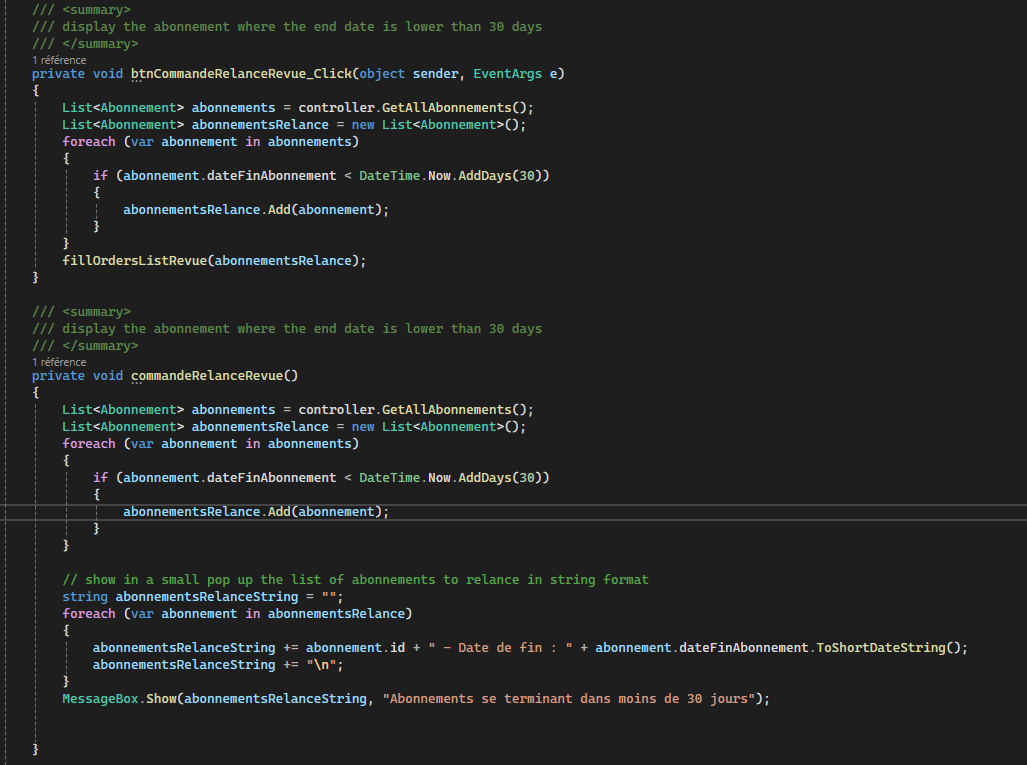


Methode setDateFinAbonnement qui permet de retrouver la date de la dernière parution de la revue : 

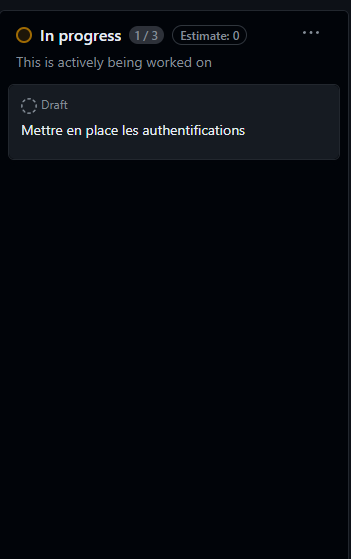
Ajout d’un abonnement dans access :



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



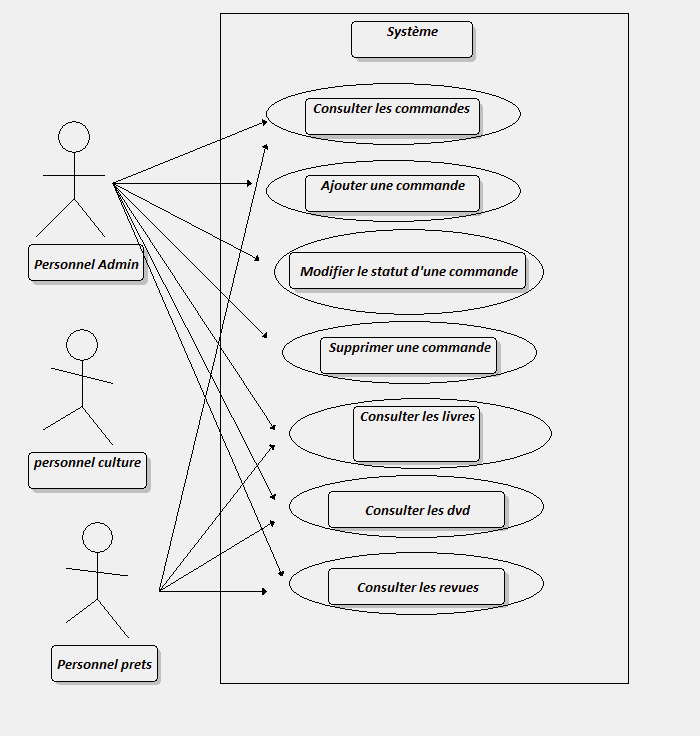
Mission 4 : Mettre en place des authentifications



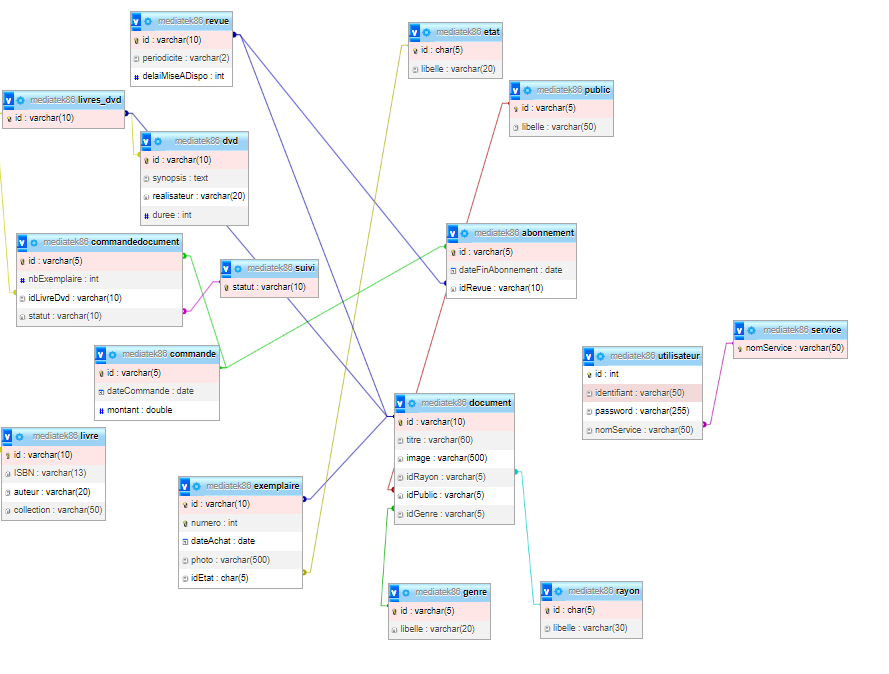
Temps estimé 4h

Temps réel 4h

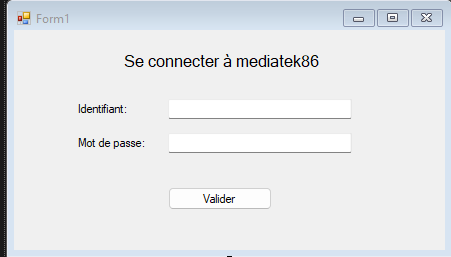
Diagramme UML de la tâche :



MCD (ajout de la table utilisateur et service) :



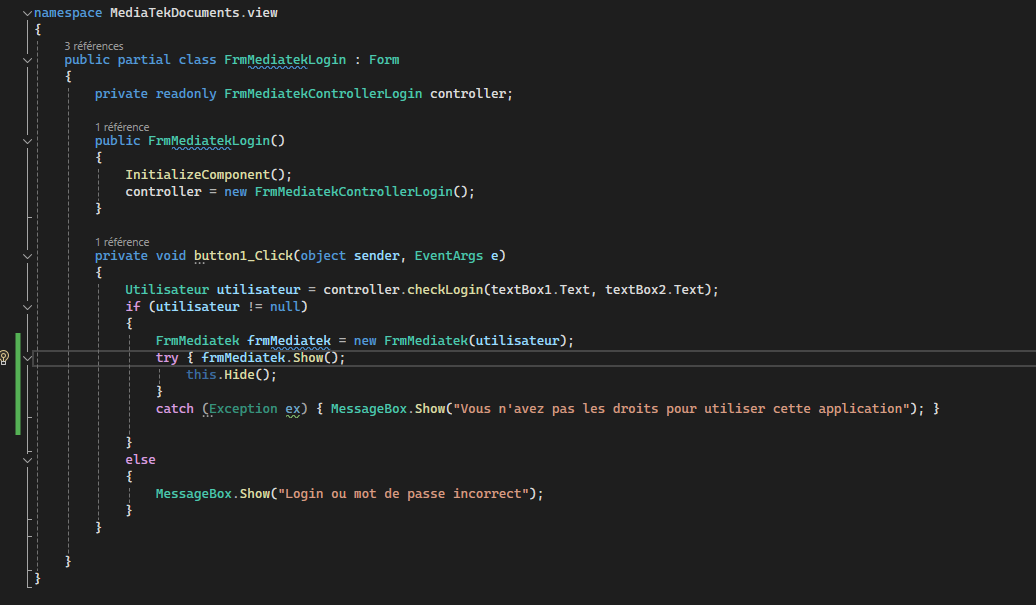
Maquette de la page d’authentification :



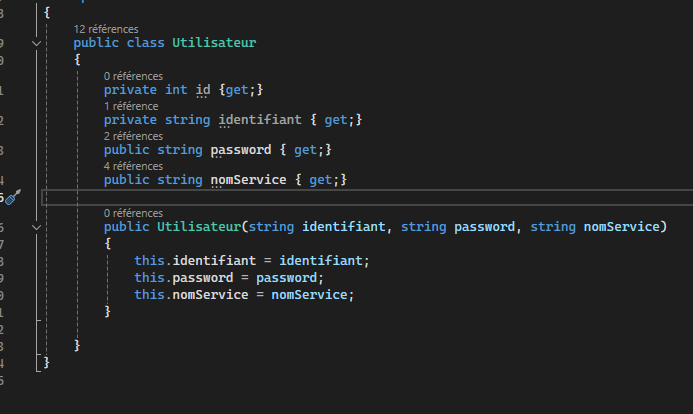
Fenêtre permettant d’accéder ou non à l’application

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

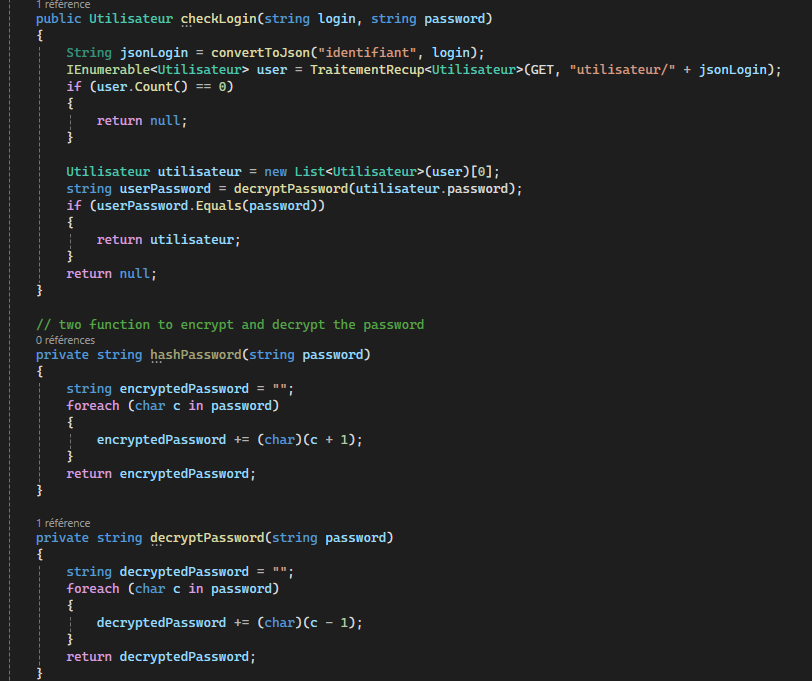
Vue admin :



Classe utilisateur :

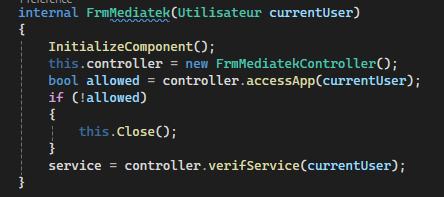


Méthodes ajoutées dans la classe access :

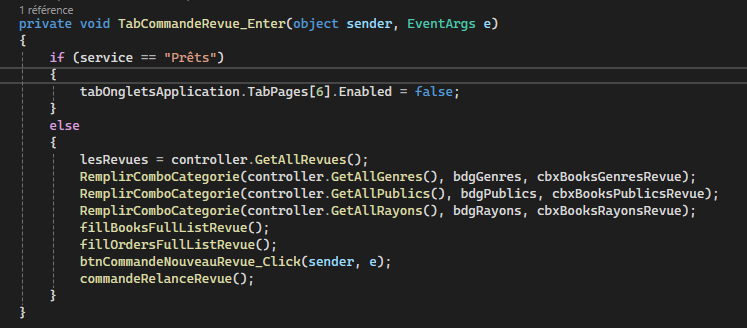


J’aurai pu utiliser un plug-in pour le hash mais j’ai décidé de l’encoder moi-même, ce n’est pas très sécure mais bon…

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).



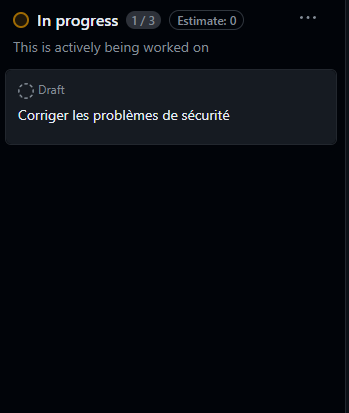
Un exemple de contrainte pour les commandes de revues concernant les membres du personnel prêts :



Ils n’ont donc pas accès aux commandes.

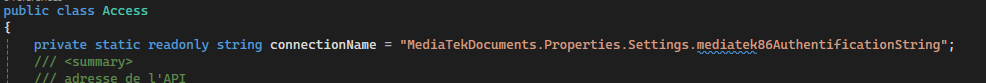
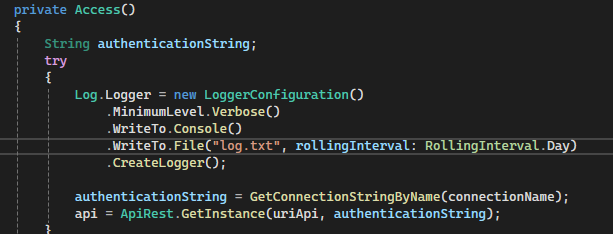
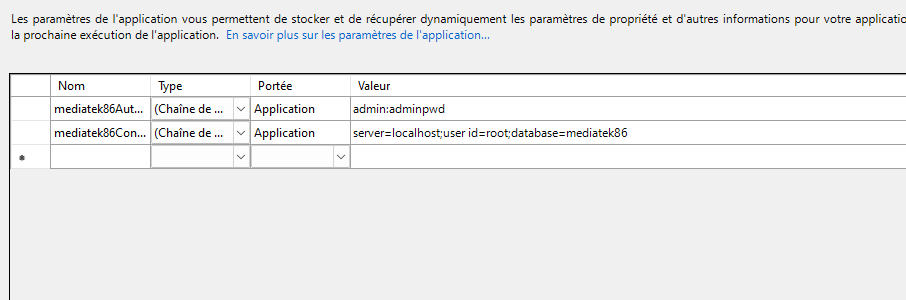
Mission 5 : Assurer la sécurité, la qualité et intégrer des logs

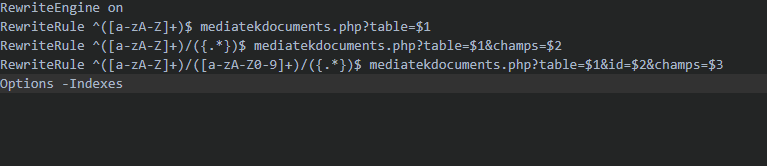
1. Corriger des problèmes de sécurité



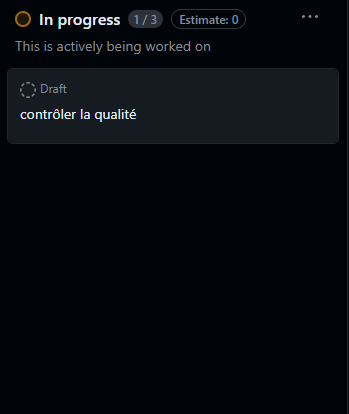
Temps estimé : 3h

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.   


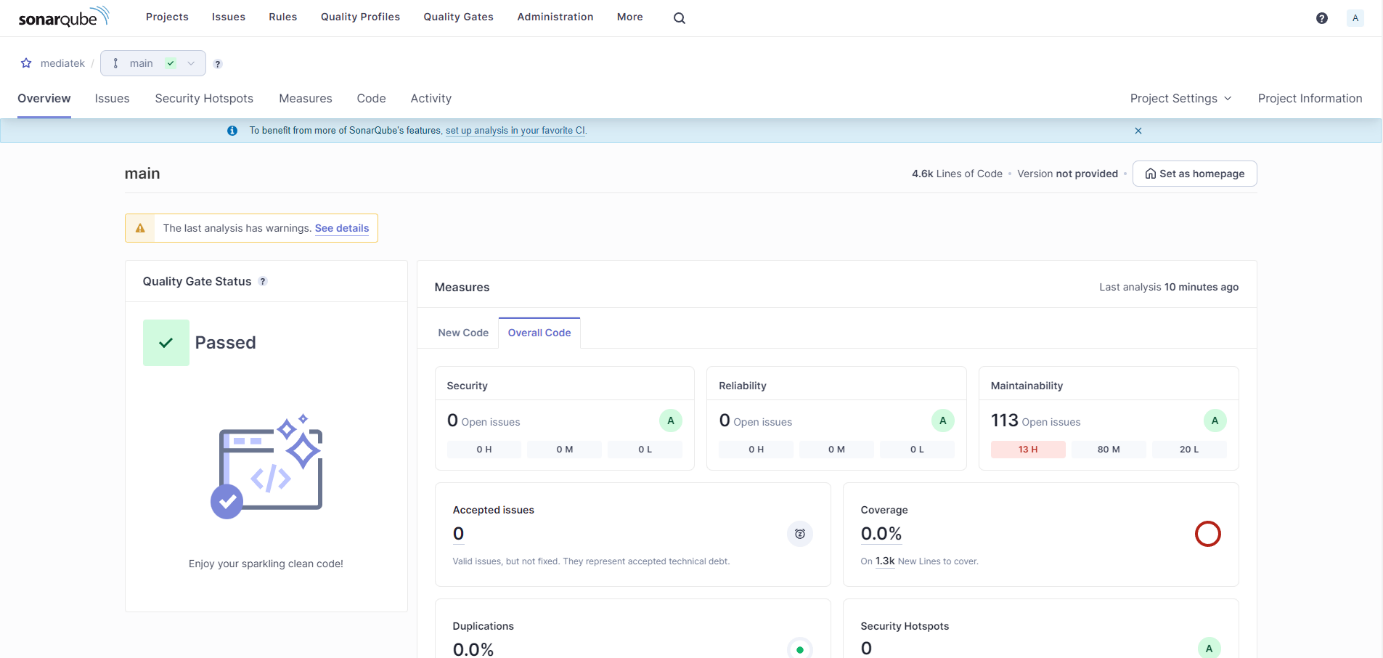
Pour la deuxième tâche il suffisait juste d’écrire « Options –Indexes » dans htaccess de l’api et les fichiers ne sont plus accessibles. 

1. Contrôler la qualité

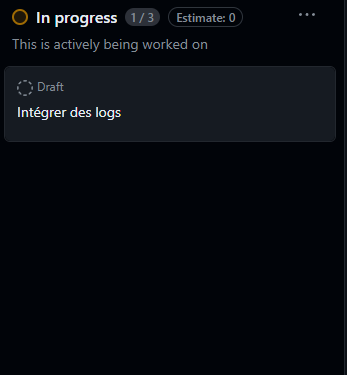


Temps estimé : 1 h

Temps réel : 1 h



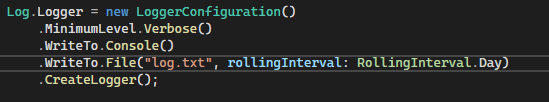
1. Intégrer des logs



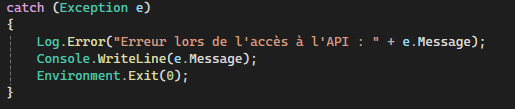
Temps estimé : 2h

Temps réel : 2h

Code pour générer des logs avec serilog :

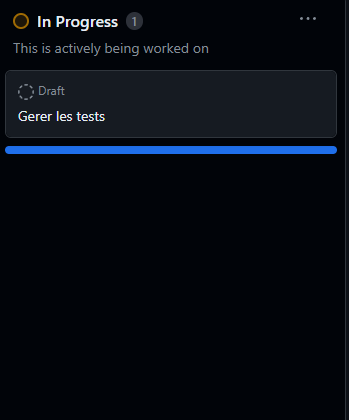


Exemple :



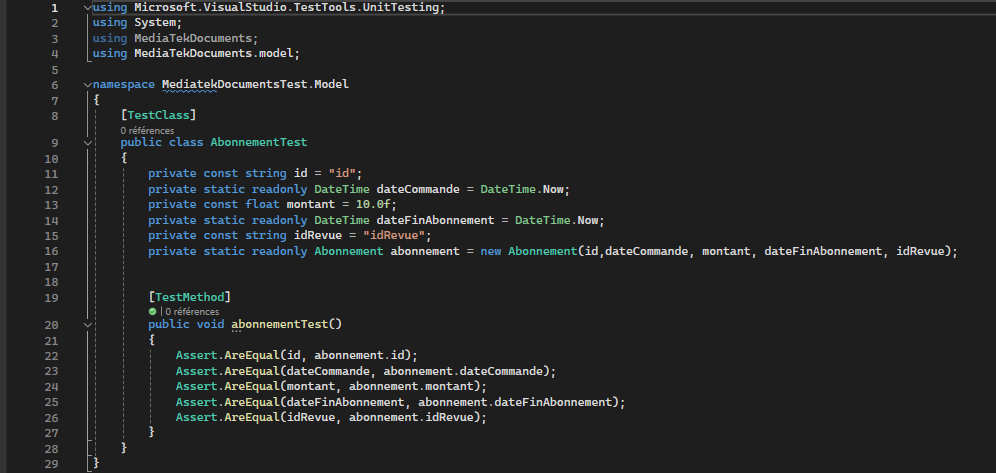
Mission 6 : Tester et documenter

1. Gérer les tests

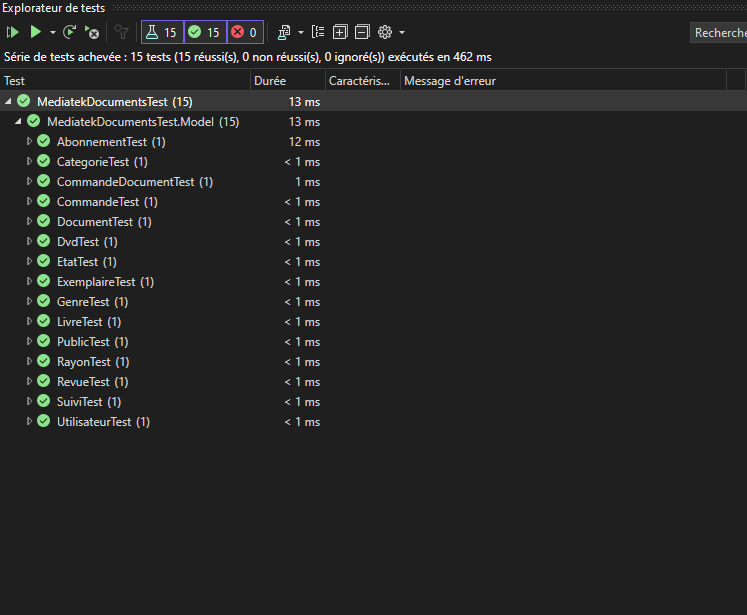


Plan de test complété :

Exemple de methode de test unitaire :

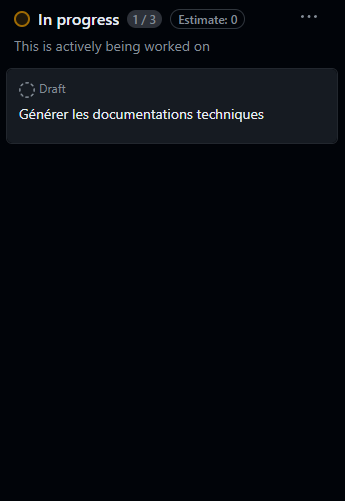


Résultats des tests unitaires :



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

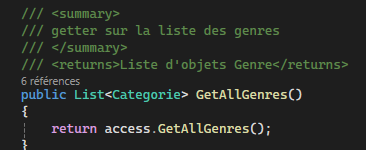
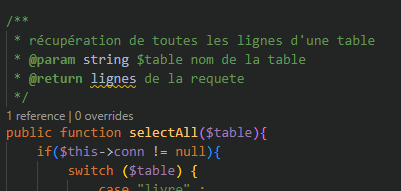
1. Créer les documentations techniques



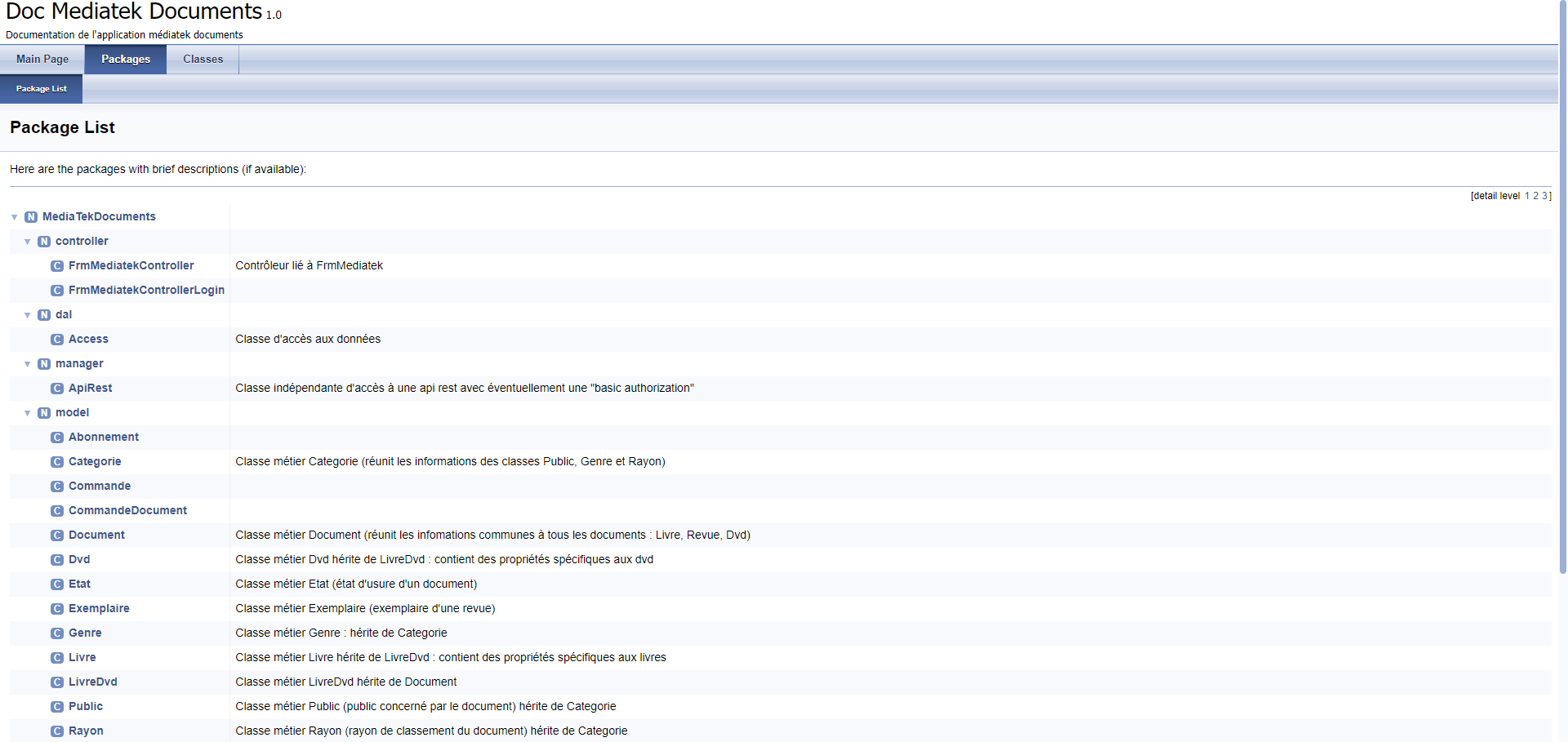
Temps estimé 1h

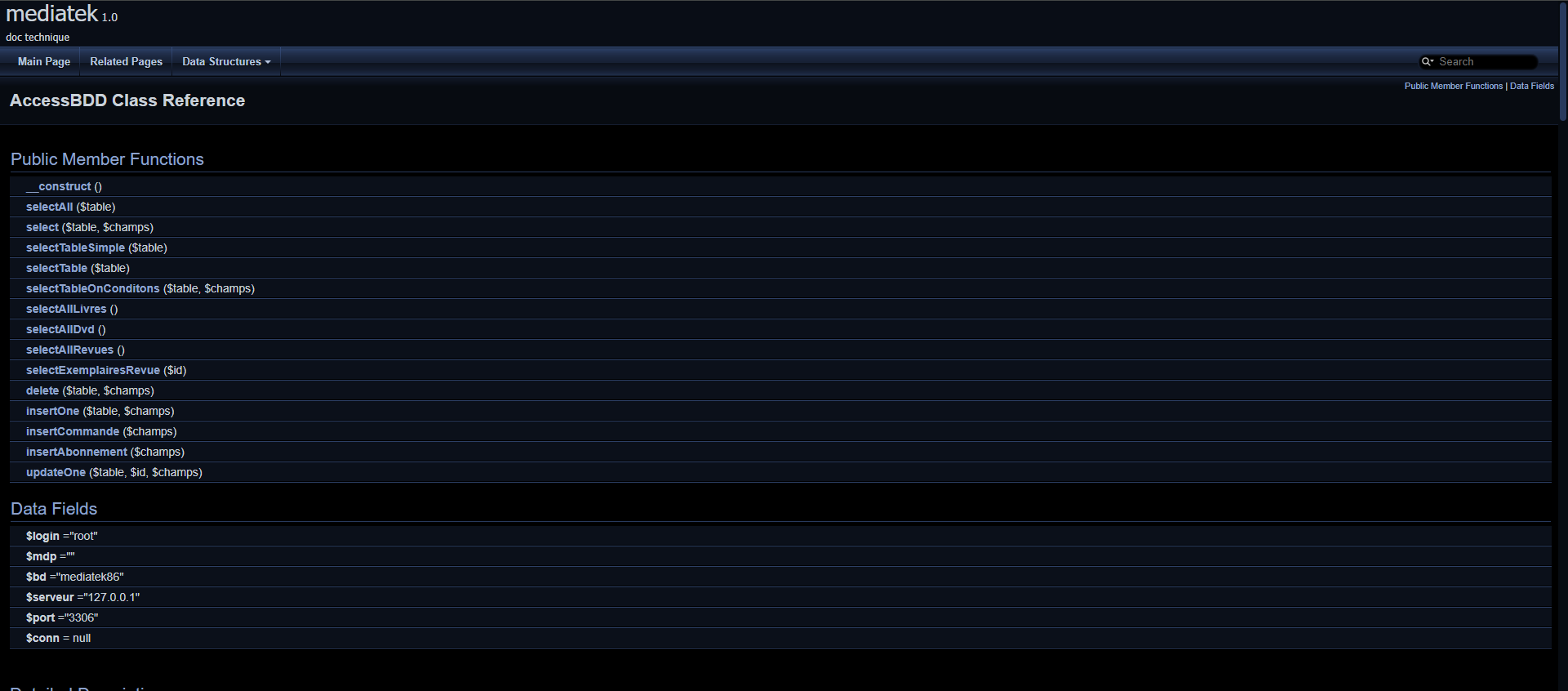
Temps réel 1h

Exemples de commentaires normalisés :

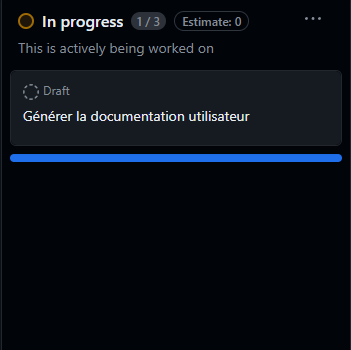


Documentation de l’application :



Documentation de l’api : 

1. Créer la documentation utilisateur



Temps estimé 2h

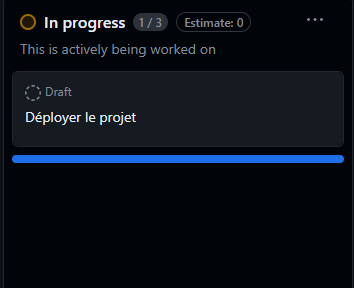
Temps réel 1h

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

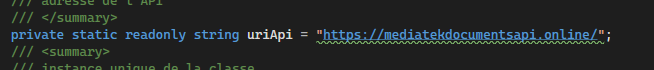
L’api a été déployée sur hostinger accessible depuis l’adresse mediatekdocumentsapi.online suivi des endpoints correspondants.

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

Une image contenant texte, capture d’écran, Police

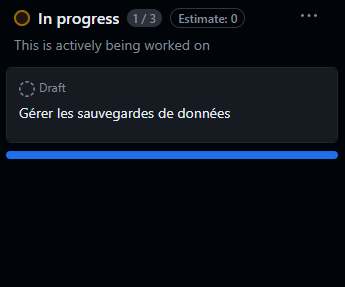
Description générée automatiquement

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

L’installeur à été générer depuis visual studio via clickOnce.

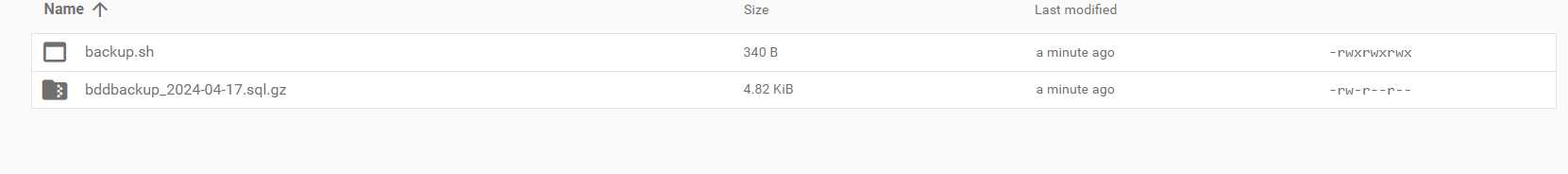
1. Gérer les sauvegardes de données



Temps estimé 1h

Temps réel 2h

La sauvegarde est en place



Toutes les minutes une sauvegarde de la bdd est faite, pour la restaurer il suffit de récupérer et de dézipper le fichier bddbackup.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.