



Compte rendu d'activité

Evolution de l'application bureau de gestion documentaire
de MediaTek86

03/04/2022

Table des matières

Compte rendu d'activité.....	1
Contexte.....	4
InfoTech Services 86.....	4
Contexte InfoTech Services 86.....	4
MediaTek86.....	4
Contexte MediaTek86.....	4
Projet DigimediaTek86.....	5
Contexte projet DigimediaTek86.....	5
Application bureau MediaTek86 Gestion documentaire.....	5
Contexte et objectifs de l'application MediaTek86 Gestion documentaire.....	5
Rappel de l'existant.....	6
La base de données.....	6
Les fonctionnalités existantes.....	7
Les packages existants.....	8
Package bdd.....	8
Package contrôleur.....	8
Package modele.....	9
Package metier.....	9
Package vue.....	9
Program.cs.....	9
Expressions des besoins.....	9
Remarques générales.....	9
Fonctionnalité gestion des documents.....	9
Fonctionnalité de gestion des commandes.....	9
Fonctionnalité de gestion du suivi de l'état des documents.....	10
Fonctionnalité d'authentification des utilisateurs.....	10
Qualité, logs et tests unitaires.....	10
Informations techniques et outils utilisés.....	11
Spécificités IDE.....	11
Spécificités outils et plugins.....	11
Tests unitaires.....	12
Documentation technique.....	12
Suivi de projet.....	12
Réalisation.....	12
Préparation de l'environnement de travail.....	12
Établir le plan de suivi sur la plateforme Trello.....	12
Mise en place de l'environnement de développement.....	18
Importation de la base de donnée.....	18
Mise en place de l'IDE Visual Studio.....	19
Récupération du code source.....	20
Lier le dépôt Github avec Visual Studio.....	20
Mise en place de sonarQube.....	21
Gestion des documents (<i>optionnel, non implémenté pour l'instant</i>).....	23
Suivi de projet.....	23
Modifications dans le contrôleur.....	23
Modifications dans la vue.....	23
Dépôt Github.....	23

Gestion des commandes.....	24
Suivi de projet.....	24
Gestion de la base de données.....	26
Création de la table suivi.....	26
Modification dans la table Commande.....	27
Ajout de données tests dans la table CommandeDocuments.....	28
Ajout de données tests dans la table Commande.....	28
Création du trigger après une update dans commande lorsqu'une commande est livrée.....	29
Mise en place des gestion des commandes de Livres.....	30
Modifications dans la vue.....	30
Création des méthodes pour l'affichage.....	32
BtnLivresCmdNumRecherche_Click().	33
Création de Classes dans modele.....	35
Ajout dans la classe Dao.....	37
Modification dans le controleur.....	38
Mise en place des gestion des commandes de DVD.....	39
Mise en place des commandes de revues.....	40
Création Classe Abonnement.....	40
Modifications dans le controleur.....	41
Test unitaire DateDeParution.....	42
Création du trigger de contrôle de partition sur héritage de commande.....	44
Ajout de la fonctionnalité Alerte de fin d'abonnement.....	45
Procédure stockée : la liste des revues dont l'abonnement se termine dans moins de 30 jours.....	45
Ajout dans le controleur.....	45
Ajout dans Dao.....	46
Ajout dans la vue.....	47
Dépot github.....	48
Mission 3 : gérer le suivi de l'état des documents.....	48
Implémentation en cours.....	48
Mission 4 : mettre en place des authentifications.....	48
Objectifs.....	48
Modifications dans la base de données.....	49
Ajout de la table service.....	49
Ajout table utilisateur.....	49
Mise en place fenêtre d'authentification.....	50
Modification du package vue.....	50
Qualité du code.....	50
Suivi de projet.....	50
Tests unitaires.....	51
Classe FrmMediatek.....	51
Classe AbonnementTest.....	51
Contrôler la qualité du code avec SonarLint et SonarQube.....	52
SonarLint.....	52
Corrections de SonarQube.....	52
Doctequine.....	54
Déploiement.....	54

Création de l'installateur.....	54
Setup Wizard.....	54
Configurer Setup_MediatekGest_Documentaire.....	55
Mise en ligne de la base de données.....	57
Heroku.....	57
JawsDB MySQL.....	58
Bilan sur les objectifs atteints.....	60
Finalités.....	60
Liste des compétences couvertes (B1, B2, B3).....	61
B1 Gérer le patrimoine informatique.....	61
B1 Répondre aux incidents et aux demandes d'assistance et d'évolution.....	61
B1 Travailler en mode projet.....	61
B1 Mettre à disposition des utilisateurs un service informatique.....	61
B1 Organiser son développement professionnel.....	61
B2 Concevoir et développer une solution applicative.....	62
B2 Assurer la maintenance corrective ou évolutive d'une solution applicative.....	62
B2 Gérer les données.....	62
B3 Assurer la cybersécurité d'une solution applicative et de son développement.....	62

Contexte

InfoTech Services 86

Contexte InfoTech Services 86

InfoTech Services 86 (**ITS 86**) est une Entreprise de Services Numériques (ESN) spécialisée dans le développement informatique d'applications bureau, web ou mobile, ainsi que dans l'hébergement, l'infogérance, la gestion de parc informatique et l'ingénierie système et réseau. Elle répond régulièrement à des appels d'offres en tant que société d'infogérance et prestataire de services informatiques grâce à son pôle Développement qui permet de proposer des solutions d'hébergements ainsi qu'une expertise d'intégration de services, de développement de logiciel ou encore de gestion de bases de données.

MediaTek86

Contexte MediaTek86

Parmi les clients d'**ITS 86** se trouve le réseau de médiathèques de la Vienne, MediaTek86 dont la gestion du parc informatique et la numérisation des activités internes du réseau dépendent des services d'**ITS 86**. Dans l'optique d'accroître l'attractivité de son réseau de médiathèques, **MediaTek86** veut développer des outils numériques pour des usages en interne ainsi que des services en ligne pour ses usagers. Ces projets

numériques sont pilotés par un chef de projet numérique chargé de la maîtrise d'ouvrage (MOA) auprès des ESN auxquelles **MediaTek86** a fait appel.

Projet DigimediaTek86

Contexte projet DigimediaTek86

Le projet DigimediaTek86 fait parti de cette liste de projet à mettre en place. L'objectif de ce projet s'étend sur deux axes, l'un interne puisqu'il implique la mise à disposition de services numériques pour ses employés, le second orienté vers ses usagers. Ici nous nous intéresseront à cette première partie puisque ce rapport détaille de la mise en place des nouvelles fonctionnalités dans l'application bureau de MediaTek86 dont la mise en place s'inscrit dans leur volonté à se numériser.

Il est à noter qu'un document détaillant davantage les différents éléments du présent contexte est accessible à ce lien sous le nom de [Contexte_MediaTek86.pdf](#)

Application bureau MediaTek86 Gestion documentaire

Contexte et objectifs de l'application MediaTek86 Gestion documentaire

L'application bureau de MediaTek86 est écrite en C#. Actuellement, elle permet de faire des recherches(tris et filtres) et d'afficher les informations sur les documents de la bibliothèque, qu'il s'agisse de livre, DVD ou revue. De plus, elle permet de gérer la réception des nouveaux numéro de revues. Cette application est actuellement utilisée sur plusieurs postes de travail au sein de la médiathèque et accède à la même base de donnée. Enfin, il n'y a actuellement qu'une seule fenêtre divisée en plusieurs onglets.

Dans un premier temps, l'objectif de cette évolution est de faire évoluer la gestion des documents en ajoutant des opération de type CRUD (ajouter, modifier et supprimer) en ajoutant une fonction de contrôle par un trigger. Dans un second temps, des interfaces de gestion de suivi des commandes des exemplaires physiques, puis des revues doit être mise en place. Puis, un système de suivi de l'état physique des documents doit être élaboré. Enfin, il est nécessaire de mettre en place un système d'authentification sécurisé

contre les injections SQL et que l'application n'offre pas les mêmes fonctionnalités en fonction du service de l'utilisateur de l'application.

Rappel de l'existant

La base de données

Pour accéder à la base de données, il faut récupérer le script de création de la base de données puis l'importer dans un logiciel de gestion de base de données sous **mySQL** permettant ainsi une gestion en local avec **phpMyAdmin**. Cette base de données, nommée **mediatek86** contient actuellement 13 tables qui sont les suivantes :

- abonnement
- commande
- commandedocument
- document
- dvd
- etat
- exemplaire
- genre
- livre
- livre_dvd
- public
- rayon
- revue

Aussi, pour davantage de clarté, le script de création a été récupéré afin de construire le **schéma UML** de cette base via **WinDesign** en utilisant sa fonction de **reverse engineering**, ce qui nous permet d'avoir la figure suivante :

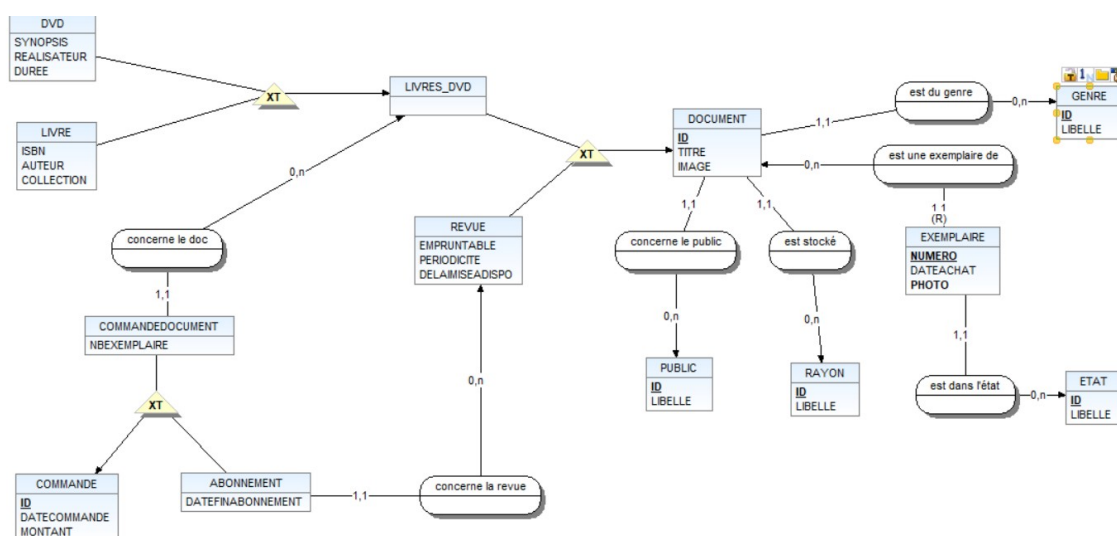


Figure 1: schéma UML de la base de donnée mediatek86 généré par WinDesign

Le script de création de cette base de donnée ainsi que son schéma UML sont accessibles via les liens suivant sous le nom [script_sql_mediatek86formations.zip](#) et [SchemaUML_bd_mediatek86.png](#)

Les fonctionnalités existantes

A l'ouverture, l'application ouvre une fenêtre contenant plusieurs onglets : Livres, DVD, Revues et Parutions de Revues. Dans chacun de ces onglets, un ensemble de objets graphique permet d'effectuer des recherches selon différents critères et le type d'objet recherché (Public, rayon, genre, titre, numero). Au clic sur un onglet, une liste détaillée affiche l'ensemble des données contenues dans la base selon le type d'onglet ouvert, par exemple si l'on est sur l'onglet revues, l'ensemble des revues contenues dans la base seront listés. Il en va de même pour les autres onglets. Ainsi la fonction de recherche permet de n'afficher que les résultats cohérents par rapport aux critères de la recherche dans l'onglet concerné. Enfin, sur le clic d'une ligne de la liste, l'ensemble des informations détaillées de l'objet sélectionné sont affichés dans un groupbox d'affichage.

Gestion Médiathèque

Onglets: Livres | DVD | **Revue** | Parutions des revues

Recherches

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

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

Ou sélectionner le rayon : X

Id	Titre	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
10006	Le Monde	QT	5	Actualités	Adultes	Presse quotidienne

Informations détaillées

Numéro de document : Empruntable : ☒

Titre :

Périodicité :

Délai mise à dispo :

Genre :

Public :

Rayon :

Chemin de l'image :

Figure 2: Exemple d'affichage de l'application à l'état initial

Enfin, les différents documents (contextes détaillés, script SQL, documentations) sont mis à dispositions à l'adresse suivante : [documentations](#). De même, le code source de l'application d'origine est accessible ici : [code source](#).

Les packages existants

Le code source fourni contient 5 packages qui sont les suivants : **bdd**, **controleur**, **metier**, **modele** et **vue**.

Package bdd

Ce package contient la Classe **DbbMySql** qui permet les exécutions et envoies des requêtes vers la base de données. Le code source initial de cette classe est accessible depuis les liens suivants : [BddMySql](#)

Package contrôleur

Ce package contient la Classe **Controle** permettant de relier les différentes Méthodes et Classes de manière contrôlée, notamment entre le package vue et le package modele. Le code source initial de cette classe est accessible depuis les liens suivants : [Controle](#)

Package modele

Ce package contient la Classe **Dao** qui permet de rassembler l'ensemble des méthodes nécessitant la création de requêtes SQL vers la base de données mediatek86. Elle utilise les méthodes du package bdd et de sa Classe **DbbMySql** pour leurs exécutions. Le code source initial de cette classe est accessible depuis les liens suivants : [Dao](#)

Package metier

Ce package contient l'ensemble des Classes types métiers permettant la création d'Objets personnalisés nécessaires à l'application, tels que **Utilisateur**, **Abonnement**, **Commande**, **Documents**, etc. Les différents codes sources initiaux de ces classes sont accessibles depuis les liens suivants : [metiers](#)

Package vue

Ce package contient la Classe **FrmMedithek** de type Form qui permet l'affichage des objets graphiques ainsi que leurs interactions. Le code source initial de cette classe est accessible depuis les liens suivants : [vue](#)

Program.cs

Ce fichier permet l'initialisation de l'application ainsi que sa mise en route. Le code source initial de cette classe est accessible depuis les liens suivants : [program](#)

Expressions des besoins

Remarques générales

L'évolution de l'application doit respecter la structure des couches ainsi que la logique du code actuelle. Aussi, il faut restreindre les possibilités de manipulations des utilisateurs, en mode lecture seule pour certains objets, le contrôle les injections SQL et éviter les saisies utilisateur par une liste de valeurs prédéfinies.

Fonctionnalité gestion des documents

L'application doit pouvoir ajouter, modifier ou supprimer des documents de types livres, DVD et revues.

dans les onglets actuels (Livres, Dvd, Revues),

- Ajouter les fonctionnalités qui permettent d'ajouter, de modifier ou de supprimer un document
- Sécurités pour éviter des erreurs de manipulation
- Créer le trigger qui contrôle la contrainte de partition de l'héritage sur Document
- Idem pour LivresDvd.

(non implémenté)

Fonctionnalité de gestion des commandes

Pour les commandes des livres ou des DVD, il faut que l'application soit capable de gérer et suivre l'évolution d'une commande qui passe par différents stade : (en cours, livrée, réglée, relancée). Cette fonctionnalité n'étant pas initialement gérée ni dans l'application, ni dans la base, il faudra mettre en place ce système. Une commande peut concerner un ou plusieurs exemplaires d'un objet. Il est aussi nécessaire de mettre en place une interface de visualisation du suivi et de gestion de ces commandes. Une fois au stade « livré », le ou les articles commandés doivent être automatiquement généré dans la BDD avec un numéro séquentiel par rapport au document concerné. Enfin, si une commande n'est pas au stade « livré », elle doit pouvoir être supprimé.

Concernant la commande de revues, une commande est un abonnement, ou un renouvellement d'abonnement. Il est donc nécessaire de mettre en place une interface de visualisation et de gestion sur laquelle il sera possible de consulter l'ensemble des abonnements de revues, ceux qui sont expirés. Enfin, une commande de revue est supprimable lorsqu'un aucun exemplaire lié à cette commande n'est enregistré.

Fonctionnalité de gestion du suivi de l'état des documents

Cette fonctionnalité permet le suivi de l'état des documents physiques : livres, DVD et revue. A la création d'un nouveau document, il est marqué automatiquement comme « neuf », puis l'application doit pouvoir changer l'état vers : usagé, détérioré, inutilisable. Enfin, il doit être possible de supprimer un exemplaire.

(non implémenté)

Fonctionnalité d'authentification des utilisateurs

L'application doit démarrer sur une demande d'authentification permettant de déterminer si l'employé est reconnu et son service. Dans le cas d'un employé du service Culture, un message doit l'informer que l'application n'est pas accessible pour leur service.

Une restriction des droits d'accès aux différentes fonctionnalités doivent être mises en place selon les services des différents employés.

Service Administratif :

- Tout accès

Service Prêts

- Consultation du catalogue disponible (livre, DVD, Revue)

Service Culture

- Aucuns accès

Administrateur

- Tout accès

Qualité, logs et tests unitaires

Afin d'assurer la propreté du code de l'application, l'extension SonarLint sera mise en place afin de trouver les erreurs de code ou de mauvaises mise en forme. Aussi des logs de journalisation doivent être ajoutés dans les catches qui contiennent les messages consoles afin de les enregistrer dans un fichier. De plus, des tests unitaires seront mis en place afin de vérifier la non régression de l'application actuelle lors du développement sur la certaines des fonctionnalités clés.

Informations techniques et outils utilisés

Spécificités IDE

L'évolution de l'application récupérée a été effectué avec la version 16.11.10 de l'IDE Microsoft Visual Studio 2019.

Spécificités outils et plugins

La revue du code produit a été effectuée par l'intermédiaire de **SonarLint** qui est un plugin permettant d'analyser du code dans l'IDE. Ainsi, il permet de mettre en évidence les éventuelles vulnérabilités et bogues lors de l'écriture du code comme le montre la *figure 1*.

Liste d'erreurs						
Solution complète		0 Erreurs	3 Avertissements	0 Messages	IntelliSense uniquement	
	Code	Description	Fichier	Ligne	Projet	État de...
▶	S125	Remove this commented out code.	Program.cs	21	Mediatek86	Actif
▶	S1848	Either remove this useless object instantiation of class 'Controle' or use it.	Program.cs	22	Mediatek86	Actif
▶	S1848	Either remove this useless object instantiation of class 'Controle' or use it.	Program.cs	22	Mediatek86	Actif

Figure 3: Exemple de contre rendu d'analyse de code avec SonarLint

De même, le logiciel [SonarQube](#) a été intégré au projet afin de mesurer la qualité du code. Il est capable de recenser plusieurs niveaux de bogues ou d'erreurs tels que le niveau de documentation ou couverture de test, de repérer les duplications de code et le non respect de règles de programmation ainsi que la complexité du code produit. Enfin, il permet d'évaluer l'application sous plusieurs notes comme le montre la figure ci-dessous :

Mediatek86_Documentaire		Passed		Last analysis: 2 minutes ago		
Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
1 C	0 A	- A	2 A	0.0% R	0.0% G	1.5k S C#

Figure 4: Exemple de résultat d'évaluation par SonarQube

Documentation technique

La documentation technique des diverses classes, propriétés et méthodes établis lors de la poursuite du développement de l'application bureau a été établi avec [Doxygen](#) qui est un logiciel sous libre licence permettant de générer une documentation technique standard. Ce dernier a été choisit pour ses fonctionnalités de personnalisations. Cette documentation est accessible en ligne à l'adresse suivante : [Documentation technique de l'application bureau](#).

Suivi de projet

Le suivit de projet a été établi avec la plateforme [Trello](#) qui permet d'établir un plan d'évolution et de suivit de projet. Enfin, l'intégralité des codes sources produits ont été stocké sur la plateforme [Github](#) au nom de projet [Mediatek86_Documentaire](#) enfin le Portfolio présentant l'ensemble de ce projet est accessible [ici](#).

Réalisation

Préparation de l'environnement de travail

Établir le plan de suivi sur la plateforme Trello

La mise en place du suivi de projet a commencé par la création d'un tableau public sur Trello du nom d'[gestionmediatek86](#). Ainsi, il est possible de constater qu'il comporte 6 listes dont les titres vont de la mission dite 0 à la mission 7 dite Mission bilan. Chacune de ces listes contiennent des cartes qui correspondent à différentes étapes des missions à effectuer. Une fonction d'étiquette de couleur permet d'avoir un suivi visuel de ce qui est « **Fini** » en violet, « **en cours** » en jaune et « **à venir** » en rouge.

La [figure 5](#) ci-contre illustre un exemple de contenu d'une de ces listes et la [figure 6](#) montre les différents menus de création.

Enfin, un intervalle de temps en jours a été établi afin d'estimer le temps de réalisation de chacune des tâches. De même, des check-lists par tâche ont été mise en place afin d'obtenir un meilleur suivi.

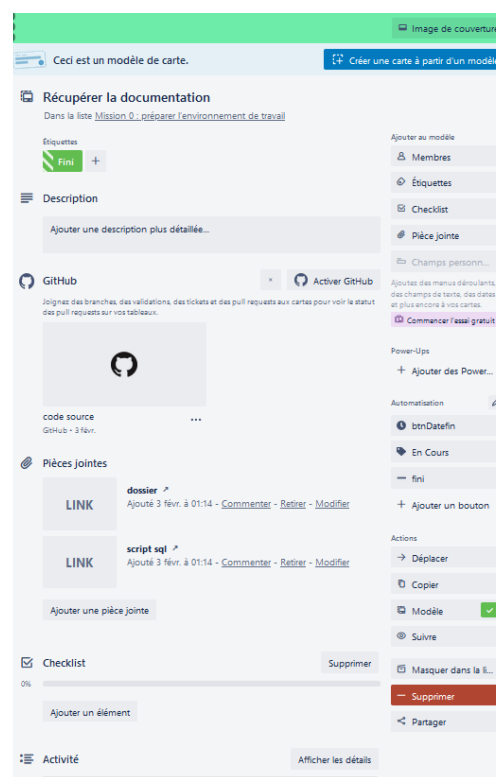


Figure 5: Exemples de carte de suivi avec étiquette, dates de fin et check-list

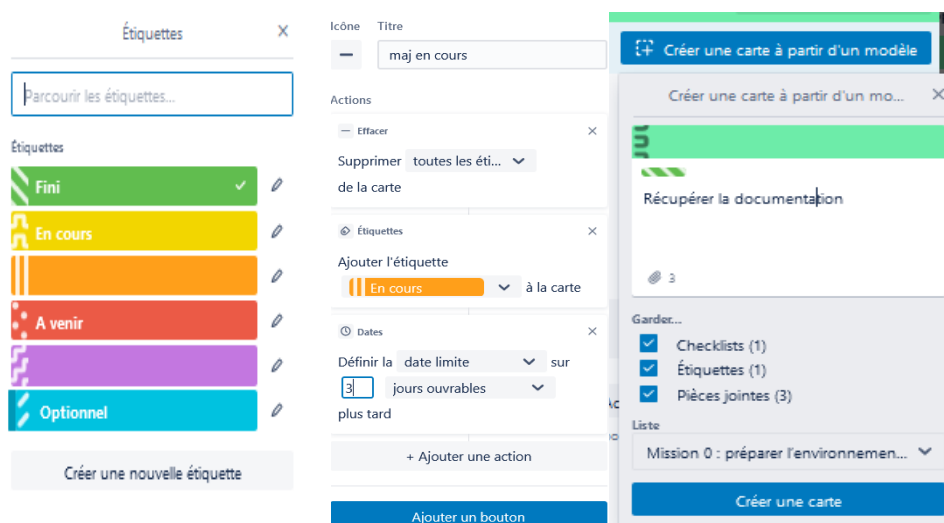


Figure 6: Menus de création d'étiquettes et des boutons d'automatisation

Aussi, afin d'accélérer la mise à jour des étiquettes, un bouton d'automatisation, « fini » à été mis en place, comme le montre la figure , ainsi en appuyant sur le bouton de la carte, lorsque l'étape est terminée l'étiquette passe de « **En cours** » à « **Fini** ». De plus, à la création d'une carte il est possible de la marquer comme carte modèle, ce qui permet de créer plus rapidement de nouvelles cartes.

Enfin, le dépôt sur GitHub a été liée à ce suivi de projet permettant d'obtenir en tant que PowerUp, qui sont des plugins permettant d'ajouter des fonctionnalités supplémentaires au tableau.



Figure 7: Powerup Github : association repository github de l'application

Concernant ce lien entre le tableau et le dépôt GitHub, il permet d'abord d'établir un lien constant vers le projet, mais aussi de faire apparaître le message du dernier commit, renseignant ainsi de l'avancé du projet. Pour finir, ce tableau Trello étant publique, il est consultable à [l'adresse suivante](#).

Il est ainsi possible d'énumérer les différentes missions :

- **Mission 0 : Préparer l'environnement de travail – temps estimé 2h**
 - **Mise en place du tableau sur Trello**
 - **Récupération de la documentation**
 - Script SQL de la base de données
 - Code source de l'application initial
 - Dossier de l'existant et des besoins
 - **Mise en place de l'environnement de développement**
 - Installation de WampServer
 - Création de la base de donnée distante dans PhpMyAdmin
 - Installation de l'IDE Visual Studio 2019
 - Tester le code source sur l'IDE
 - Création du dépôt sur Github

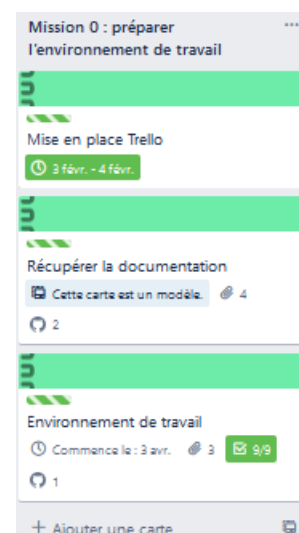


Figure 8: Cartes de la mission 0 préparer l'environnement de travail

- Mise en place d'analyse SonarQube
- **Mission 1 : Gérer les opération CRUD sur les documents – temps estimé 7h (Optionnel)**
 - 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.
- **Mission 2 : Coder les fonctionnalités de gestion des commandes – temps estimé 15h**
 - **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.
 - **Dans l'application**
 - **Onglet pour gérer les commandes de livres.**
 - Dans l'onglet, 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.
 - 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.

■ **Onglet pour gérer les commandes de DVD**

- Créer un onglet pour gérer les commandes de revues
- une commande représente un nouvel abonnement ou le renouvellement d'un abonnement. Dans le cas d'un nouvel abonnement, la revue sera préalablement créée dans l'onglet Revues. Donc, dans l'onglet des commandes de revues, il n'y a pas de distinction entre un nouvel abonnement et un renouvellement.
- Permettre la sélection d'une revue par son numéro, afficher les informations de la revue ainsi que la liste des commandes (abonnements), triée par date (ordre inverse de la chronologie). La liste doit comporter les informations suivantes : date de la commande, montant et date de fin d'abonnement.
- Créer un groupbox qui permet de saisir les informations d'une nouvelle commande (nouvel abonnement ou renouvellement, le principe est identique) et de l'enregistrer. Une commande de revue peut être supprimée, si aucun exemplaire n'est rattaché (donc, en vérifiant la date de l'exemplaire, comprise entre la date de la commande et la date de fin d'abonnement).
- Créer et utiliser la méthode 'ParutionDansAbonnement' qui reçoit en paramètre 3 dates (date commande, date fin abonnement, date parution) et qui retourne vrai si la date de parution est entre les 2 autres dates.
- **Créer le test unitaire sur cette méthode.**
- La présentation de chaque onglet de gestion des commandes doit être similaire à l'onglet "Parutions des revues".
- Dans toutes les listes, permettre le tri sur les colonnes.
- 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.
- Créer une procédure stockée qui permet d'obtenir la liste des revues dont l'abonnement se termine dans moins de 30 jours. Dès l'ouverture de l'application, ouvrir une petite fenêtre d'alerte rappelant la liste de ces revues (titre et date de fin abonnement) triée sur la date dans l'ordre chronologique.
- **Mission 3: Gérer le suivi des documents– temps estimé 6h**
 - Agrandir la fenêtre en hauteur
 - Onglet Livres, partie basse,
 - Ajouter la liste des exemplaires du livre sélectionné.

- Contient les colonnes : 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 ;
 - Onglet "Parutions des revues" :
 - Liste des parutions,
 - remplacer la colonne "Photo" par "Etat".
 - Permettre aussi le changement d'état ;
 - Permettre de supprimer un exemplaire.
-
- **Mission 4: Mettre en place les authentification – temps estimé 6h**
 - 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 ;
 - Ajout d'une première fenêtre d'authentification que le contrôleur ouvre en première ;
 - Selon le service de la 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).
 - **Mission 5: Qualité, tests et documentation technique – temps estimé 6h**
 - Contrôler la qualité du code avec SonarLint
 - Créer des tests fonctionnels (avec SpecFlow ou directement avec les NUnit)
 - Ajouter des logs dans les catch qui contiennent des affichages consoles et enregistrer ces logs dans un fichier texte ;
 - Création de la documentation technique de l'application
 - **Mission 6: Création d'une vidéo de démonstration via OBS – temps estimé 1h**
(+ 24h pour son acceptation par la modération de la plateforme d'hébergement)
 - **Mission Bilan**
 - Déployer l'application et la mettre à disposition sur la plateforme de dépôt.
 - Mise en ligne de la base de donnée distante
 - Rédaction du contre rendu d'activité.
 - Mise en place du portfolio de l'activité.

La **figure 9** suivante illustre ce tableau de suivi à la fin de son élaboration, soit la Mission 0.

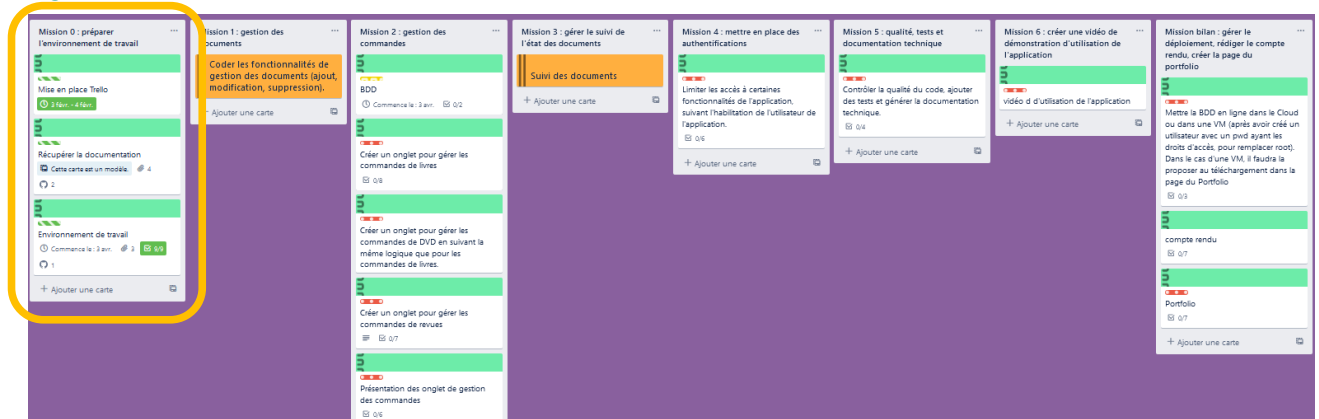


Figure 9: Tableau de suivi de projet Trello, début de la mission 0

Mise en place de l'environnement de développement

Importation de la base de donnée

Après ouverture de WampServer, dans la barre des tâches de Windows, il est nécessaire de cliquer sur **phpMyAdmin**. A cet instant, une nouvelle page du navigateur s'ouvre pour afficher un nouveau formulaire, cf **figure 10**. Il s'agit de la page de connexion à phpMyAdmin, il faut alors remplir en tant qu'utilisateur : « root » et laisser vide pour le mot de passe, à moins que ce dernier ait été configuré.

Figure 10: Fenêtre de connexion à phpMyAdmin

En arrivant dans le menu de phpMyAdmin, il faut alors aller dans l'onglet « importer » qui affiche alors la *figure 12*.

Importation dans le serveur courant

Fichier à importer :

Le fichier peut être compressé (gzip, bzip2, zip) ou non.
Le nom du fichier compressé doit se terminer par `.[format].[compression]`. Exemple : `.sql.zip`

Parcourir les fichiers : Script_BD_mediatek86.sql (Taille maximale : 128Mio)

Il est également possible de glisser-déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier :

Figure 11: Menu d'importation de script SQL de phpMyAdmin

Il convient ainsi de renseigner le script de création de la base de données, soit **Script_BD_mediatek86.sql**, puis en bas de page cliquer sur « Executer ». La page finit par s'actualiser pour afficher plusieurs messages indiquant que l'importation a réussi, comme l'illustre la *figure 12*.

✓ L'importation a réussi, 57 requêtes exécutées. (Script_BD_mediatek86.sql)

Figure 12: Exemples de messages confirmant la bonne importation du script de création de la base de données

Il est alors possible de voir la base de donnée **mediatek86** apparaître dans la liste des bases disponibles, puis en cliquant dessus l'ensemble de ses 13 tables apparaissent.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> abonnement	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> commande	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> commandedocument	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> document	Parcourir Structure Rechercher Insérer Vider Supprimer	41	InnoDB	utf8mb4_general_ci	64,0 kio	-
<input type="checkbox"/> dvd	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> etat	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> exemplaire	Parcourir Structure Rechercher Insérer Vider Supprimer	14	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> genre	Parcourir Structure Rechercher Insérer Vider Supprimer	19	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> livre	Parcourir Structure Rechercher Insérer Vider Supprimer	26	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> livres_dvd	Parcourir Structure Rechercher Insérer Vider Supprimer	30	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> public	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> rayon	Parcourir Structure Rechercher Insérer Vider Supprimer	15	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> revue	Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8mb4_general_ci	16,0 kio	-
13 tables	Somme	168	MyISAM	utf8_unicode_ci	304,0 kio	0 o

Figure 13: Structure des tables de la base de donnée

Mise en place de l'IDE Visual Studio

Visual Studio est un IDE permettant le développement d'application C#. L'IDE est téléchargeable à [ce lien](#), ici il s'agit de la version 2019. Après l'installation de cet IDE avec les options par défaut, il convient de le démarrer en mode administrateur afin de simplifier les différents éléments de démarrage. De plus, ici le logiciel a été configuré pour s'ouvrir automatiquement dans ce mode. Pour cela, il convient d'effectuer un clic droit sur l'icône de lancement de Visual Studio, d'aller dans « Propriétés », puis à l'ouverture de la fenêtre des propriétés, de cliquer sur « Avancé » puis de cocher « exécuter en tant qu'administrateur » comme le montre la [figure 14](#).

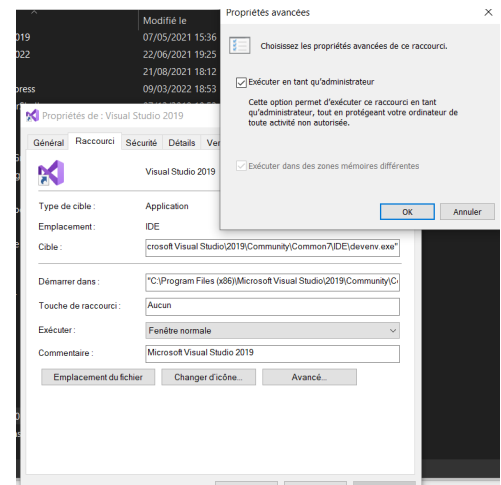


Figure 14: Fenêtre des propriétés de l'icône de lancement de l'IDE

Récupération du code source

La récupération du code source se fait sur la plateforme GitHub à cette [adresse](#) en cliquant sur le bouton vert à droite avec inscrit « Code » puis de sélectionner « **Download zip** ». Après téléchargement, avoir dézipper le dossier et avoir placé le projet dans le dossier de traitement, ici dans : `%user%\Programmation\Git\Mediatek86_Documentaire`

Pour ouvrir le projet, il convient de se rendre dans le répertoire de dépôt local puis de cliquer sur le lien du projet pour Visual Studio : **Mediatek86.sln**. Enfin, il suffit de cliquer sur l'icône de lecture verte à côté de démarrer afin de lancer l'application.

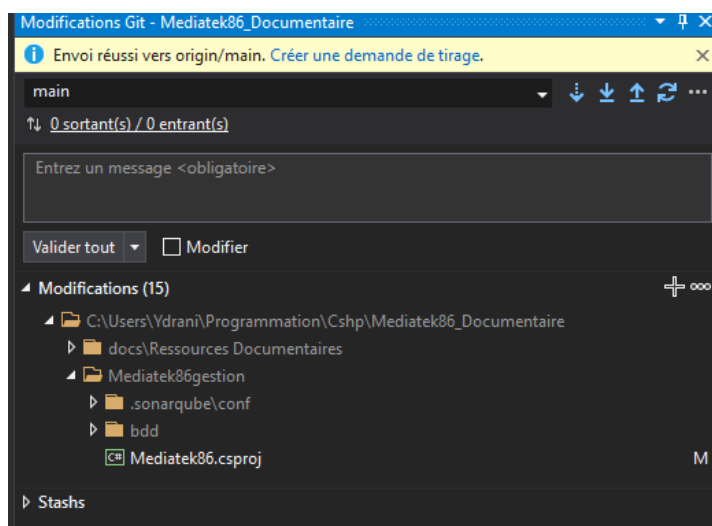


Figure 15: Fenêtre de contrôle de version Github sur VisualStudio

Lier le dépôt Github avec Visual Studio

Afin de pouvoir directement gérer le dépôt GitHub depuis Visual Studio, il convient d'aller dans l'onglet « **Modification Git** » puis de saisir un message de commit dans la fenêtre qui

apparaît. A cet instant, les modifications non intégrés au dépôt s'affiche en dans l'onglet **Modification**. Dans le cas présent, le dépôt étant préalablement créé et le compte de dépôt étant déjà lié à Visual Studio, il suffit de commiter puis de cliquer sur « **valider tout et pousser** ». A la réussite de l'envoi, une pop-up s'affiche.

Mise en place de sonarQube

Pour utiliser cette plateforme, il convient de télécharger l'application puis de l'installer dans le dossier racine **C :** du poste de travail. Ensuite, il suffit de lancer le fichier **StartSonar.bat**. Plusieurs fenêtres d'invite de commande s'exécute afin de lancer le serveur SonarQube local, de se rendre à l'adresse **localhost:9000** dans un navigateur, puis de se connecter avec login et mot de passe admin.

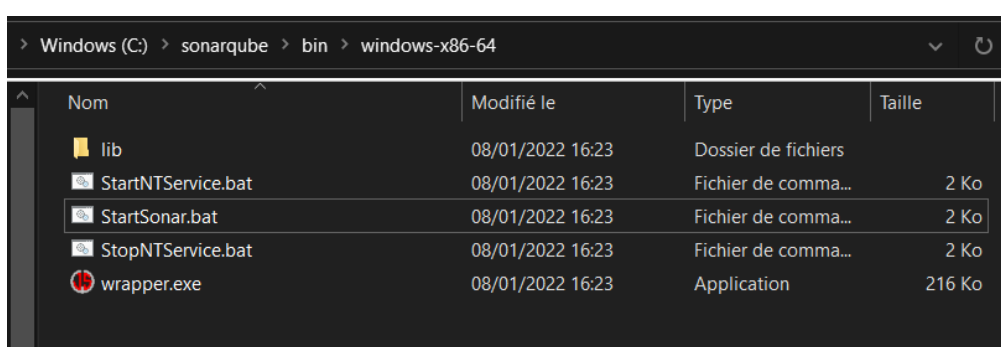


Figure 16: Localisation du launcher du server de SonarQube

Aussi dans le dossier racine du poste de travail, dans le dossier de **sonarscanner**, il faut modifier le fichier **SonarQube.Analysis.xml** afin de remplacer la valeur de **sonar.login** par celle d'un token généré pour sonarQube et éventuellement celui de **sonar.host.url** par celui de la valeur de **url** du serveur **sonarQube**. Puis, dans le dossier du projet de l'application, il faut ajouter la ligne de code suivante dans le fichier **Mediatek86.csproj**

```
<PropertyGroup>
<!-- Project is not a test project -->
  <SonarQubeTestProject>false</SonarQubeTestProject>
</PropertyGroup>
```

Une fois dans le menu de **sonarQube**, il convient de créer un nouveau projet et de le créer avec les options en local, puis générer un token, sélectionner le type de projet, ici gradle puis d'exécuter le script affiché dans la console de l'IDE.

Localement, puis en générant le token ou en récupérant un déjà existant. Puis, en choisissant le build **.NET**, puis **.NETFramework**. A la sélection de ce dernier plusieurs lignes de commandes apparaissent qu'il convient d'exécuter dans une invite de commande dans le répertoire de l'application.

```
SonarScanner.MSBuild.exe begin /k:"Mediatek86_Documentaire" /d:sonar.host.url=
"http://localhost:9000" /d:sonar.login="xxxxx"
MsBuild.exe /t:Rebuild
SonarScanner.MSBuild.exe end /d:sonar.login="xxxxx"
```

Où, « **xxxxx** » est le token généré à la création du projet. Ainsi, une fois la dernière ligne exécuté et après avoir refresh la page il est possible d'obtenir l'affichage suivant :

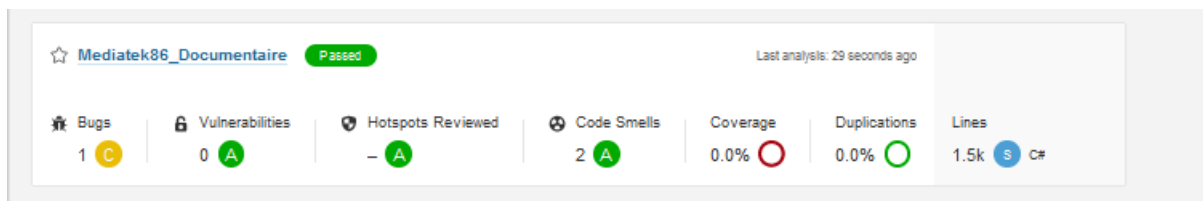


Figure 17: Analyse de l'application à l'Etat initial dans SonarQube

Une fois de retour dans **Visual Studio**, il est possible d'obtenir directement les informations de **sonarQube**. Pour cela il est nécessaire d'avoir installé les extensions nécessaires depuis l'onglet Extension présent dans l'IDE puis de choisir les extensions pour **SonarLint** et **SonarQube**. Dans la fenêtre **Team explorer**, en cliquant sur l'onglet de SonarQube, puis sur **connect** une demande d'authentification est demandé. Il convient alors de renseigner les informations de connexion au serveur de **SonarQube**. Puis la liste des projets existant sur le serveur apparaissent. Il suffit de double cliquer sur celui correspondant afin de le lier à Visual Studio.

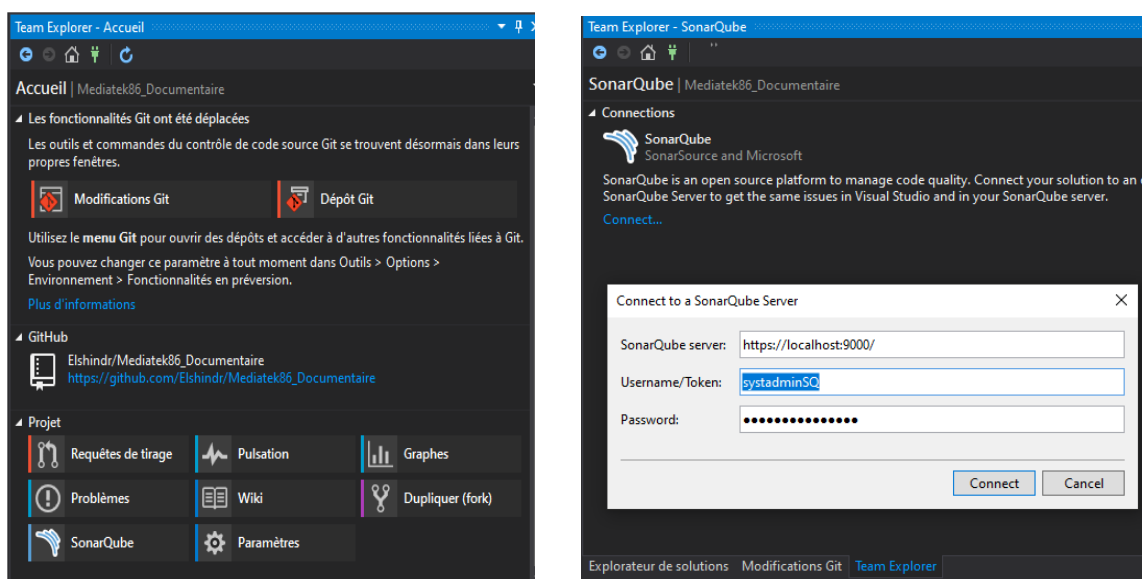



Figure 18: Fenêtre de Team explorer dans VisualStudio

Mise à jour par push

1 Create GitHub Secrets

In your GitHub repository, go to **Settings > Secrets** and create two new secrets:

1 Click on **New repository secret**


2 In the **Name** field, enter `SONAR_TOKEN` 


3 In the **Value** field, enter an existing token, or a newly generated one:

[Generate a token](#)

4 Click on **Add secret**

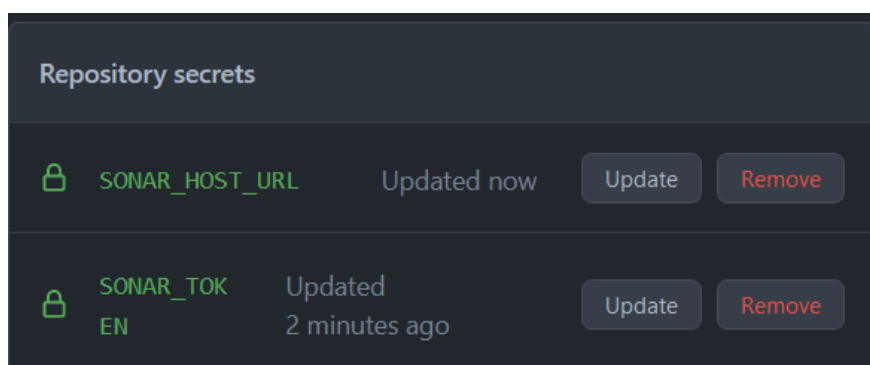
1 Click on **New repository secret**

2 In the **Name** field, enter `SONAR_HOST_URL` 

3 In the **Value** field, enter `http://localhost:9000` 

4 Click on **Add secret**

[Continue](#)



Gestion des documents *(optionnelle, non implémenté pour l'instant)*

Suivi de projet

Selon le tableau de suivi établi dans Trello illustré par la *figure*, nous nous situons dans la mission 1 qui consiste à gérer les fonctionnalités de type CRUD sur les documents. Il est possible de constater que les check-list de la mission 0 ont bien été rempli et les étiquettes sont passés de jaune à violet, indiquant ainsi leur fin de traitement.

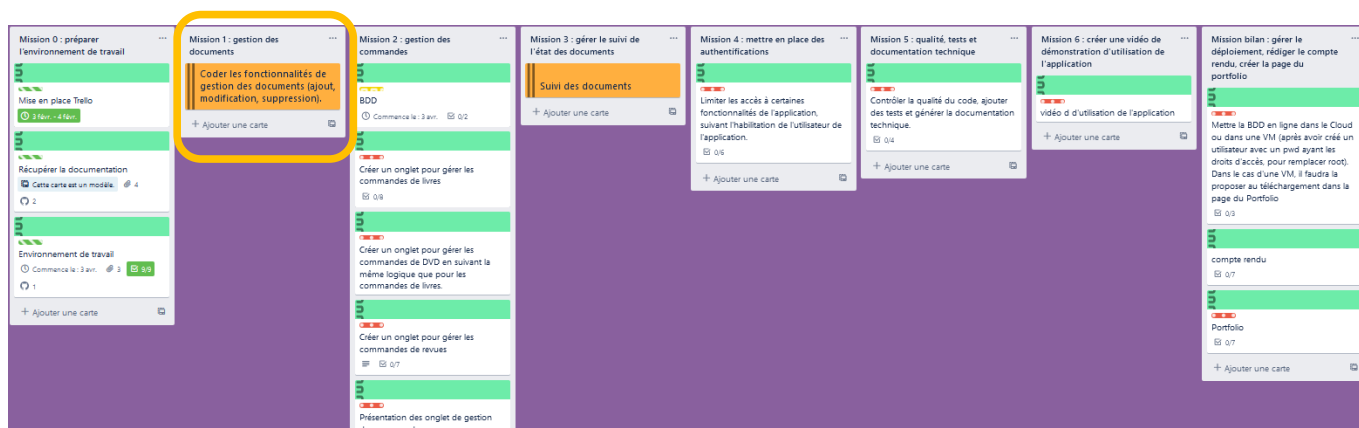


Figure 19: Tableau de suivi début de mission 1

Modifications dans le contrôleur

Modifications dans la vue

Dépôt Github

A l'issue de ce code, un commit a été effectué sur la plateforme GitHub dans le dépôt du projet.

Gestion des commandes

Suivi de projet

Selon le suivi de projet, nous nous situons au début de la mission 2 : gestion des commandes cf [figure 20](#). Dans cette intention, la liste est composée de 5 cartes chacune correspondant à une étape différente, à savoir :

- Modifier la base de donnée en conséquence
- Création d'un onglet pour gérer les commandes des livres
- Création d'un onglet pour gérer les commandes des DVD
- Création d'un onglet pour gérer les commandes des revues
- Présentation des onglets de gestion des commandes et création de procédures dans la base de données

Les étiquettes des cartes de la **Mission 0** sont passées de jaune à vert indiquant ainsi la fin de ces étapes. De même, les différentes check-lists ont été mises à jours.

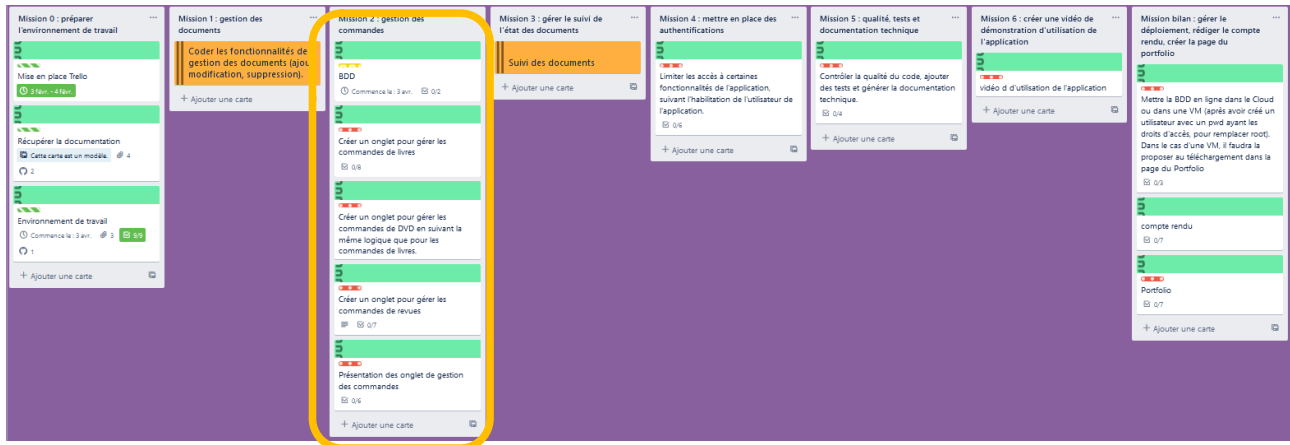


Figure 20: Suivi de mission Début de l'étape 2 gérer la fonctionnalité des commandes

Concernant à nouveau la mission 2, chaque carte comportent une check-list avec ses différentes étapes.

- **Gestion de 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 pour gérer les commandes de livres**
 - Dans l'onglet, 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.



Figure 21: Liste des étapes de la mission 2

- 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**
 - Doit suivre la même logique que pour les commandes de livres.
- **Créer un onglet pour gérer les commandes de revues**
 - dans le cas d'un nouvel abonnement, la revue sera préalablement créée dans l'onglet Revues. Donc, dans l'onglet des commandes de revues, il n'y a pas de distinction entre un nouvel abonnement et un renouvellement.
 - Permettre la sélection d'une revue par son numéro, afficher les informations de la revue ainsi que la liste des commandes (abonnements), triée par date (ordre inverse de la chronologie).
 - La liste doit comporter les informations suivantes : date de la commande, montant et date de fin d'abonnement.
 - Créer un groupbox qui permet de saisir les informations d'une nouvelle commande (nouvel abonnement ou renouvellement, le principe est identique) et de l'enregistrer.
 - Une commande de revue peut être supprimée, si aucun exemplaire n'est rattaché (donc, en vérifiant la date de l'exemplaire, comprise entre la date de la commande et la date de fin d'abonnement)
 - Pour cela, créer et utiliser la méthode 'ParutionDansAbonnement' qui reçoit en paramètre 3 dates (date commande, date fin abonnement, date parution) et qui retourne vrai si la date de parution est entre les 2 autres dates.
 - Créer le test unitaire sur cette méthode.
- **Présentation des onglets de gestion des commandes de revue**
 - La présentation de chaque onglet de gestion des commandes doit être similaire à l'onglet "Parutions des revues".
 - Dans toutes les listes, permettre le tri sur les colonnes.
 - 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
- Créer une procédure stockée qui permet d'obtenir la liste des revues dont l'abonnement se termine dans moins de 30 jours.
- Dès l'ouverture de l'application, ouvrir une petite fenêtre d'alerte rappelant la liste de ces revues (titre et date de fin abonnement) triée sur la date dans l'ordre chronologique.

Gestion de la base de données

Création de la table suivi

Pour cela, il faut se connecter à **phpMyAdmin**, se rendre dans la base de données mediatek86 puis dans l'onglet Concepteur afin de récupérer le schéma suivant :

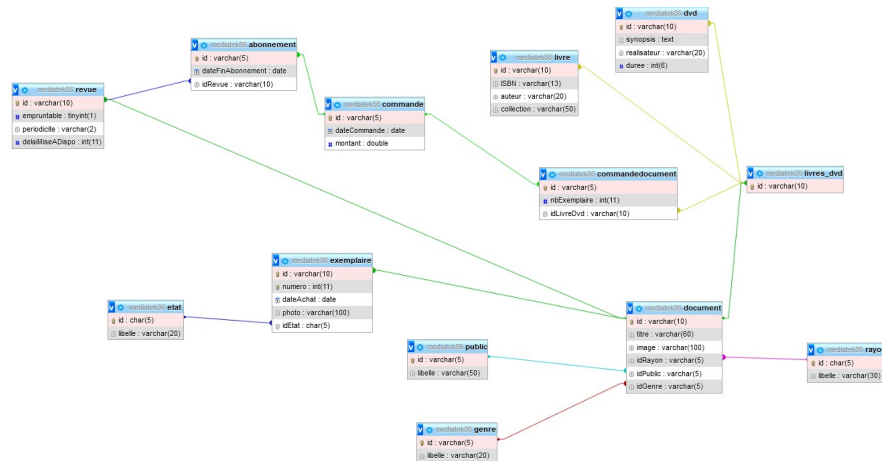


Figure 22: Schéma de la base mediatek86 du concepteur de phpMyAdmin

Puis dans l'onglet de SQL, la table suivi est créée de la manière suivante :

```
CREATE TABLE suivi (
  idSuivi INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
  label varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Concernant les informations nécessaires à cette table, elle concerne les étapes de suivi d'une commande qui, selon l'analyse des besoins, doit avoir les statuts suivant : *en cours*, *livrée*, *réglée* et *relancé*. Ainsi, nous avons un id de type integer en tant que clé primaire de la table, puis un label de type (15)varchar, Enfin, dans ce même onglet SQL, la requête SQL exécutée afin de remplir la table est la suivante :

```
INSERT INTO suivi (label)
VALUES
  ('en cours'),
  ('livrée'),
  ('réglée'),
  ('relancé');
```

Modification dans la table Commande

Afin de pouvoir récupérer les données de la table suivi, il est nécessaire de modifier la table commande en lui ajoutant une colonne supplémentaire avec les requêtes suivante s :

```
ALTER TABLE commande ADD idSuivi INTEGER NOT NULL
```

```
ALTER TABLE commande
```

```
ADD CONSTRAINT commande_ibfk_1 FOREIGN KEY (idSuivi) REFERENCES suivi(idSuivi);
```

La première permet d'ajouter la colonne, la seconde d'ajouter idSuivi de la table suivi en tant que clé étrangère dans la table commande. Ainsi dans le concepteur, on peut alors obtenir la *figure 23* suivante :

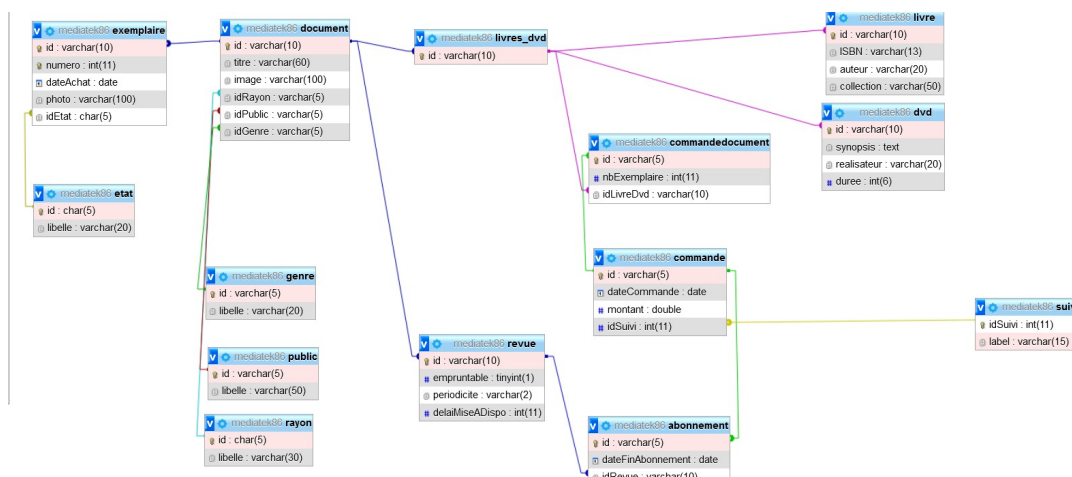


Figure 23: Schéma de la base mediatek86 avec les modifications

Requête d'ajout de données tests dans la table CommandeDocuments

```
INSERT INTO commandedocument(id, nbExemplaire, idLivreDvd)
VALUES
('00001', 3, '00017'),
('00002', 4, '00017'),
('00003', 12, '00017'),
('00004', 1, '00017'),
('00005', 5, '00004'),
('00006', 5, '00004');
```

```
('00007', 67, '00004');
```

Requête d'ajout de données tests dans la table Commande

```
Insert INTO `commande` (id, dateCommande, montant, idSuivi)
VALUES
('00001', '2021-01-13', 1.99, 3),
('00002', '2021-12-23', 123.43, 4),
('00003', '2021-11-08', 123.43, 1),
('00004', '2021-11-23', 55.10, 2),
('00005', '2021-12-24', 23.3, 3),
('00006', '2021-01-24', 1.03, 1),
('00007', '2022-11-24', 1.43, 2);
```

Création du trigger :Après une update dans commande si une commande est livrée

A la mise à jour d'un suivi de commande sur « livré », un trigger se déclenche après la prise en compte de la validation du nouveau suivi. Il permet de créer autant de tuple dans la table exemplaire qu'il y a eu de nombre d'exemplaire du document commandé. Cette requête SQL a directement été intégré dans la SGBD de phpMyAdmin par l'onglet SQL.

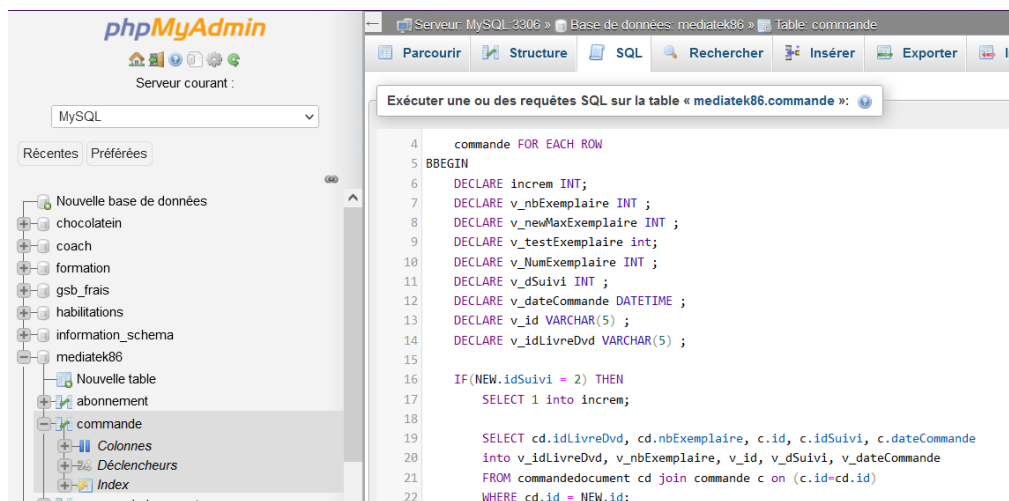


Figure 24: Exemple d'exécution de requête SQL dans phpMyAdmin

```
CREATE TRIGGER `trigInsertExemplrs` AFTER UPDATE ON `commande`
FOR EACH ROW BEGIN
  DECLARE increm INT;
  DECLARE v_nbExemplaire INT ;
  DECLARE v_newMaxExemplaire INT ;
  DECLARE v_testExemplaire int;
  DECLARE v_NumExemplaire INT ;
```

```

DECLARE v_dSuivi INT ;
DECLARE v_dateCommande DATETIME ;
DECLARE v_id VARCHAR(5) ;
DECLARE v_idLivreDvd VARCHAR(5) ;

IF(NEW.idSuivi = 2) THEN
    SELECT 1 into increm;

    SELECT cd.idLivreDvd, cd.nbExemplaire, c.id, c.idSuivi, c.dateCommande
    into v_idLivreDvd, v_nbExemplaire, v_id, v_dSuivi, v_dateCommande
    FROM commandedocument cd join commande c on (c.id=cd.id)
    WHERE cd.id = NEW.id;

    SELECT COUNT(numero) into v_newMaxExemplaire
    FROM exemplaire
    WHERE id = v_idLivreDvd;

    REPEAT
        INSERT INTO exemplaire(id, numero, dateAchat, photo, idEtat)
        VALUES ( v_idLivreDvd, v_newMaxExemplaire+increm, v_dateCommande,
        "", '00001');
        SELECT (increm+1) into increm ;

    UNTIL (increm > v_nbExemplaire) END REPEAT ;

END IF ;
END

```

Mise en place des gestion des commandes de Livres

Modifications dans la vue

Dans l'IDE, pour ajouter un nouvel onglet il faut se rendre dans le package vue, puis dans le designer **FrmMediatek.Designer.cs**. Une fois après avoir sélectionné un onglet déjà présent, il est possible d'ajouter d'autres onglets depuis la fenêtre des propriétés de **tabOngletsApplication** qui est l'objet graphique représentant les onglets. Afin de modifier son texte ainsi que de déterminer son

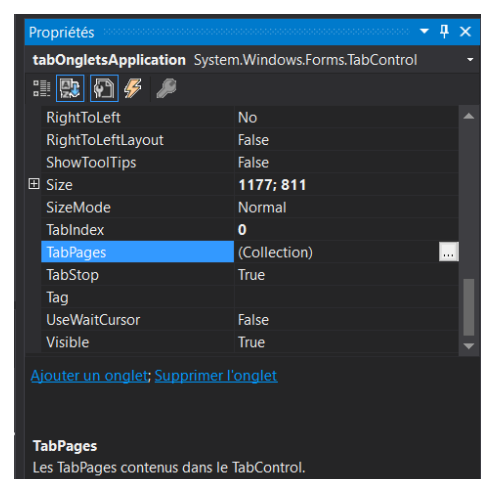


Figure 25: Fenêtre des propriétés de l'objet tabOngletsApplication

identifiant de code, il faut se rendre dans l'attribut **TabPage** qui ouvre une fenêtre permettant de gérer une collection d'onglets.

Une fois dans cette fenêtre, il est possible d'ajouter ou de supprimer d'autres onglets, ainsi que de saisir les attributs de **Text** et de **Name** qui correspondent réciproquement au titre de l'onglet et à son nom identifiant dans le code.

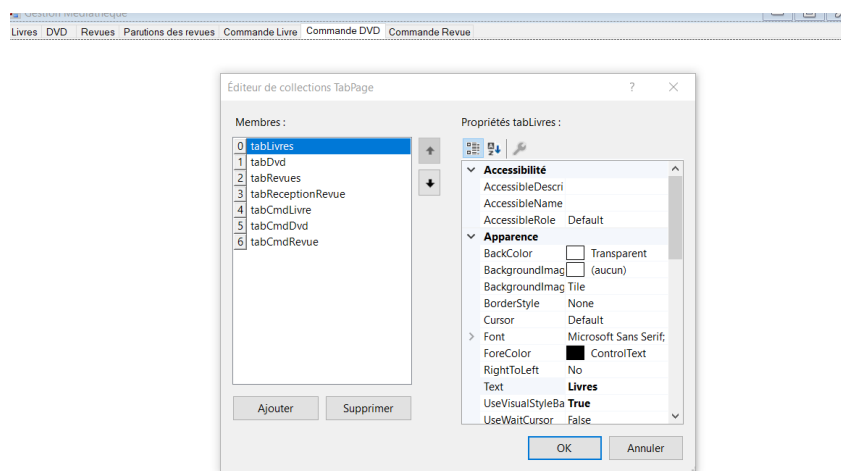


Figure 26: Propriétés de Collection de tabOngletsApplication et ses nouveaux onglets

Étant donné que la mission actuelle demande la création des onglets pour le suivi de livre, de dvd et de revue, ces trois onglets ont été directement créés à cette étape. Il est à noter que pour l'instant les autres propriétés de cet objet graphique n'ont pas à être modifiées afin de conserver la structure et l'aspect d'origine de l'application.

Afin de mettre en place la saisie du numéro du livre, un **label**, suivi d'une **TextBox** puis d'un **bouton** de Recherche ont été mis en place. Concernant l'affichage des informations d'un livre, l'ensemble du **groupbox** de l'onglet livre a été récupéré afin de préserver la structure. Seuls les propriétés **Name** des différents objets graphiques ont été modifiées afin de s'adapter à l'onglet **Commande Livre**. Par exemple, si le nom **txbLivresTitre** correspond à l'onglet Livre son équivalent dans l'onglet **Commande de Livres** est **txbCmdLivresTitre**.

Commandes de livres

Numéro de livre : 00017

Code ISBN : 3,21457E+12

Titre : Catastrophes au Brésil

Auteur(e) : Philippe Masson

Collection :

Genre : Policier

Public : Ados

Rayon : Jeunesse romans

Chemin de l'image :

Commandes :

DateCommande	Montant	NbExemplaire	Label	Id	IdLivreDvd
13/04/2022	2,5	4	livrée	00003	00017
13/04/2022	2,5	7	livrée	00005	00017
12/04/2022	6,5	7	livrée	00004	00017
12/04/2022	2	2	livrée	00002	00017
11/04/2022	1	4	livrée	00001	00017

Nouvelle commande du livre

Nombre d'exemplaires : 0

Montant : 0

Date Commande : 12/04/2022

Modification de commande du livre

IdCommande :

Date Commande :

IdLivre :

Nb Exemplaires :

Suivi : livrée

Figure 27: Vue de l'onglet Commande de livre après modifications

La mise en place d'un **DataGridView** permet d'afficher la liste des commandes d'un livre, ici nommé **dgvLivresCmdListe**. L'ensemble de ses paramètres sont laissés par défauts et identiques à ceux des autres DataGridView afin de respecter les normes de l'application

Création des méthodes pour l'affichage

A l'effigie du mode opératoire du code déjà présent, une nouvelle région a été ajoutée dans le fichier **FrmMediatek.cs**. Cette région est nommée **LivresCommandes**.

```
#region LivresCommandes

//-----
// ONGLET "COMMANDES D'UN LIVRE"
//-----
```

Figure 28: Exemple d'ajout de région

BtnLivresCmdNumRecherche_Click()

La méthode de gestion de l'événement sur le clique du bouton de recherche permet de vérifier

Son code est le suivant :

```
/// <summary>
/// Recherche et affichage du livre dont on a saisi le numéro.
/// Si non trouvé, affichage d'un MessageBox.
/// </summary>
```

```
/// <param name="sender"></param>
/// <param name="e"></param>
private void BtnLivresCmdNumRecherche_Click(object sender, EventArgs e)
{
    if (!txbLivresCmdRecherche.Text.Equals(""))
    {
        Livre livre = lesLivres.Find(x =>
x.Id.Equals(txbLivresCmdRecherche.Text));

        if (livre != null)
        {
            AfficheLivresCmdInfos(livre);
        }
        else
        {
            MessageBox.Show("numéro introuvable");
            txbLivresCmdRecherche.Text = "";
        }
    }
}
```

Cette méthode appelle la méthode qui permet l'affichage des informations du livre recherché **AfficheLivresCmdInfo()** qui est la suivante :

```
/// <summary>
/// Affichage des informations du livre sélectionné dans l'onglet des
commandes
/// </summary>
/// <param name="livre"></param>
private void AfficheLivresCmdInfos(Livre livre)
{
    txbLivresCmdAuteur.Text = livre.Auteur;
    txbLivresCmdCollection.Text = livre.Collection;
    pcbLivresCmdImage.Text = livre.Image;
    txbLivresCmdIsbn.Text = livre.Isbn;
    txbLivresCmdNumero.Text = livre.Id;
    txbLivresCmdGenre.Text = livre.Genre;
    txbLivresCmdPublic.Text = livre.Public;
    txbLivresCmdRayon.Text = livre.Rayon;
    txbLivresCmdTitre.Text = livre.Titre;
    string image = livre.Image;
    try
    {

```



```

        pcbLivresCmdImage.Image = Image.FromFile(image);
    }
    catch
    {
        pcbLivresCmdImage.Image = null;
    }
    string idDocument = txbLivresCmdNumero.Text;
    lesCmdLivre = controle.GetAllCommandesLivre(idDocument);
    RemplirLivresCmdListe(lesCmdLivre);
}

```

En fin de méthode, un appel vers le contrôleur est effectué afin de récupérer les informations depuis la base de données par l'appel `GetAllCommandesLivre()` où l'identifiant du livre, `idDocument`, est injecté afin de servir de filtre dans la requête SQL de sélection. Puis, un appel vers la méthode **RemplirLivresCmdListe()** permet l'affichage des informations du livre recherché dans le `DataGridView` de l'onglet de commande de livre qui est la suivante :

```

/// <summary>
/// Remplit le datagrid des commandes de livres avec la liste reçue en paramètre
/// </summary>
/// <param name="commandes">Liste des commandes du document</param>
private void RemplirLivresCmdListe(List<Commande> commandes)
{
    bdgLivresCmdListe.DataSource = commandes;
    dgvLivresCmdListe.DataSource = bdgLivresCmdListe;
    dgvLivresCmdListe.Columns["id"].Visible = false;
    dgvLivresCmdListe.Columns["idSuivi"].Visible = false;
    dgvLivresCmdListe.Columns["idLivreDvd"].Visible = false;

    dgvLivresCmdListe.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
    dgvLivresCmdListe.Columns["dateCommande"].DisplayIndex = 0;
    dgvLivresCmdListe.Columns["montant"].DisplayIndex = 1;
    dgvLivresCmdListe.Columns["nbExemplaire"].DisplayIndex = 2;
    dgvLivresCmdListe.Columns["label"].DisplayIndex = 3;
}

```

Cette liste est alimentée par la propriété **DataSource** qui ici a la valeur de la liste des commandes du livre concerné. Afin de lier la liste des commandes nommée `commandes` et celle de l'objet graphique `DataGridView` nommé **dgvLivresCmdListe**, il est nécessaire de mettre en place un objet **BindingSource**, ici nommée **bdgLivresCmdListe**, comme le montre le code suivant placé dans les déclarations de variables globales.

```
private readonly BindingSource bdgLivresCmdListe = new BindingSource();  
private List<Commande> lesCmdLivre = new List<Commande>();
```

Création de Classes dans modele

Création des classes **Suivi**, **Commande** qui hérite de la nouvelle classe **CommandesDocuments**.

Dans **Commande.cs**

```
public class Commande : CommandeDocument  
{  
  
    private readonly DateTime dateCommande;  
    private readonly double montant;  
    private readonly string idSuivi;  
    private readonly string label;  
  
    public Commande(string idCommande, string idLivreDvd, int  
nbExemplaire, DateTime dateCommande, double montant, string idSuivi,  
string label)  
    : base(idCommande, idLivreDvd, nbExemplaire)  
    {  
        this.DateCommande = dateCommande;  
        this.Montant = montant;  
        this.IdSuivi = idSuivi;  
        this.Label = label;  
    }  
  
    public DateTime DateCommande { get; set; }  
    public double Montant { get; set; }  
    public string IdSuivi { get; set; }  
    public string Label { get; set; }  
}
```

S

Dans **CommandeDocument.cs**

```
public class CommandeDocument  
{  
    private string id;
```

```
private string idLivreDvd;  
private int nbExemplaire;  
public CommandeDocument(string id, string idLivreDvd, int  
nbExemplaire)  
{  
    this.Id = id;  
    this.IdLivreDvd = idLivreDvd;  
    this.NbExemplaire = nbExemplaire;  
}  
  
public string Id { get => id; set => id = value; }  
public string IdLivreDvd { get => idLivreDvd; set => idLivreDvd =  
value; }  
public int NbExemplaire { get => nbExemplaire; set => nbExemplaire =  
value; }  
}
```

Suivi.cs

```
public abstract class Suivi  
{  
    private readonly string id;  
    private readonly string libelle;  
  
    protected Suivi(string id, string libelle)  
    {  
        this.id = id;  
        this.libelle = libelle;  
    }  
  
    public string Id { get => id; }  
    public string Libelle { get => libelle; }  
  
    /// <summary>  
    /// Récupération du libellé pour l'affichage dans les combos  
    /// </summary>  
    /// <returns></returns>  
    public override string ToString()  
    {  
        return this.libelle;  
    }  
}
```

Ajout dans la classe Dao

```
/// <summary>
/// Retourne toutes les commandes d'un livre à partir de la BDD
/// </summary>
/// <returns>Liste d'objets Livre</returns>
public static List<Commande> GetAllCommandesLivre(string idDocument)
{
    List<Commande> lesCmdLivres = new List<Commande>();
    string req = "Select c.id, c.dateCommande, c.montant, c.idSuivi,
s.label, cd.nbExemplaire, cd.idLivreDvd";
    req += " from commande c join suivi s on c.idSuivi = s.idSuivi";
    req += " join commandedocument cd on c.id = cd.id";
    req += " where cd.idLivreDvd = @iddoc";
    req += " order by c.dateCommande DESC";

    Dictionary<string, object> parameters = new Dictionary<string, object>
    {
        { "@iddoc", idDocument }
    };

    BddMySQL curs = BddMySQL.GetInstance(connectionString);
    curs.Select(req, parameters);

    while (curs.Read())
    {
        string idCommande = (string)curs.Field("id");
        DateTime dateCommande = (DateTime)curs.Field("dateCommande");
        double montant = (double)curs.Field("montant");

        int idSuivi = (int)curs.Field("idSuivi");
        string label = (string)curs.Field("label");

        string idLivreDvd = (string)curs.Field("idLivreDvd");
        int nbExemplaire = (int)curs.Field("nbExemplaire");

        Commande commande = new Commande(idCommande, idLivreDvd,
nbExemplaire, dateCommande, montant, idSuivi.ToString(), label);
    }
}
```

```
        lesCmdLivres.Add(commande);  
    }  
    curs.Close();  
  
    return lesCmdLivres;  
}
```

Modification dans le controleur

```
/// <summary>  
/// getter sur la liste des commandes d'un livre  
/// </summary>  
/// <returns>Collection d'objets Commande </returns>  
public List<Commande> GetAllCommandesLivre(string idDocumentlivreDvd)  
{  
    return Dao.GetAllCommandesLivre(idDocumentlivreDvd);  
}
```

Mise en place des gestion des commandes de DVD

La procédure est identique que pour la gestion des commandes de livre puisqu'ils dépendent tout deux de la table et de la classe commandedocument. Il suffit de créer la vue adaptée similaire à celle des commandes de livres à la différence que les informations détaillés sont celles des dvd. L'ensemble des fonctionnalités en terme de recherche, modifications de suivi, d'ajout de commande et de suppression sont identiques, à la différence que les noms des object graphiques sont différents.

Gestion Médiathèque

[Livres](#)
[DVD](#)
[Revue](#)
[Parutions des revues](#)
[Commande Livre](#)
[Commande DVD](#)
[Commande Revue](#)

Commandes de DVD

Numéro de DVD:

Titre :

Réalisateur(trice) :

Synopsis :

Genre :

Public :

Rayon :

Durée :

Chemin de l'image :

Commandes :

DateCommande	Montant	NbExemplaire	Label
12/04/2022	0,5	1	relancée
12/04/2022	2	4	en cours
12/04/2022	1,5	4	réglée
12/04/2022	1	1	en cours

Nouvelle commande du DVD

 Nombre d'exemplaires :

 Montant :

 Date Commande :

Modification de commande du DVD

 IdCommande :

 IdDvd :

 Date Commande :

 Nb Exemplaires :

 Suivi :

Mise en place des commandes de revues

Création Classe Abonnement

```

/// <summary>
/// Classe de gestion des Abonnements d'une revue
/// Herite de la Classe Commande
/// </summary>
public class Abonnement : Commande
{
    /// <summary>
    /// Variable privée de type string de l'identifiant de la revue
    /// </summary>
    private string idRevue;

    /// <summary>
    /// Variable privée de type DateTime de la date de fin d'abonnement
    /// </summary>
    private DateTime dateFinAbonnement;

    /// <summary>
    /// Constructeur de la classe Abonnement
    /// </summary>

```

```

    /// <param name="id"></param>
    /// <param name="idRevue"></param>
    /// <param name="dateFinAbonnement"></param>
    public Abonnement(string idCommande, DateTime dateCommande, double
montant, string idSuivi, string label, string idRevue, DateTime
dateFinAbonnement) :
        base(idCommande, dateCommande, montant, idSuivi, label)
    {
        this.IdRevue = idRevue;
        this.DateFinAbonnement = dateFinAbonnement;
    }

    /// <summary>
    /// Getter et Setter de la propriété idRevue autogénérés
    /// </summary>
    public string IdRevue { get => idRevue; set => idRevue = value; }

    /// <summary>
    /// Getter et Setter de la propriété dateFinAbonnement autogénérés
    /// </summary>
    public DateTime DateFinAbonnement { get => dateFinAbonnement; set =>
dateFinAbonnement = value; }
}

```

Modifications dans le controleur

```

    /// <summary>
    /// Methode du controleur accédant à la méthode GetAllCommandesRevue
    /// </summary>
    /// <param name="idRevue">Identifiant d'une revue</param>
    /// <returns>La liste des abonnements d'une revue</returns>
    public List<Abonnement> GetAllAbonnementRevue(string idRevue)
    {
        return Dao.GetAllAbonnementRevue(idRevue);
    }

    /// <summary>
    /// Methode de controle de création d'une commande d'abonnement
    /// </summary>
    /// <param name="abo">Un identifiant d'abonnement d'une revue</param>
    /// <returns>Vrai si CreerAbonnement retourne Vrai</returns>
    public bool CreerAbonnement(Abonnement abo)
    {
        return Dao.CreerAbonnement(abo);
    }

```

```

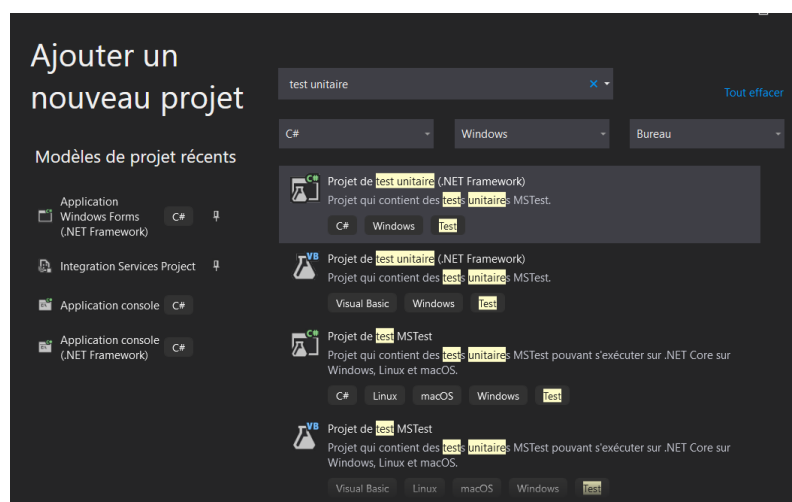
}

/// <summary>
/// Methode du controleur accedant à la methode SupprimerAboRevue
/// </summary>
/// <param name="idCommande">Identifiant d'une commande</param>
/// <returns>Vrai si SupprimerAboRevue retourne Vrai</returns>
public bool SupprimerAboRevue(string idCommande)
{
    return Dao.SupprimerAboRevue(idCommande);
}

/// <summary>
/// Methode du controleur accedant à la methode GetDateParution
/// </summary>
/// <param name="idRevue">Identifiant d'une revue</param>
/// <returns>Date de Parution d'une revue</returns>
public DateTime GetDateParution(string idRevue)
{
    return Dao.GetDateParution(idRevue);
}

```

Test unitaire DateDeParution



Configurer votre nouveau projet

Projet de test unitaire (.NET Framework) C# Windows Test

Nom du projet

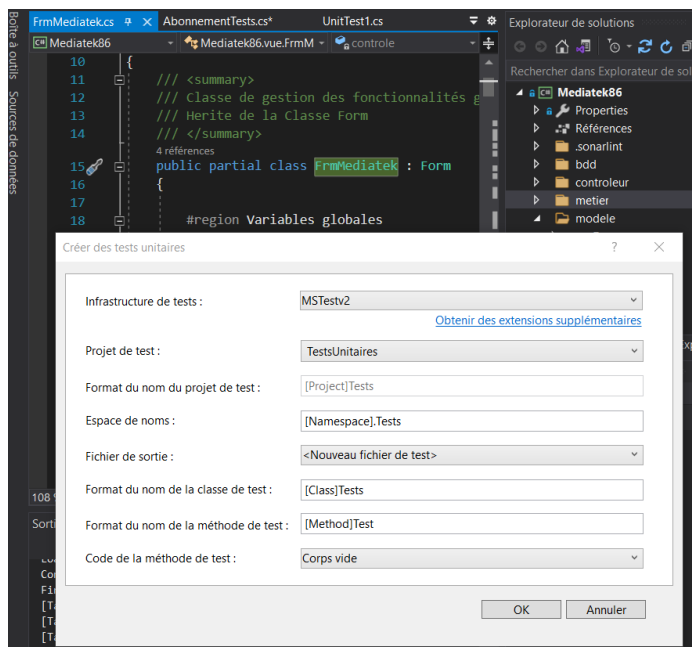
TestsUnitaires

Emplacement

C:\Users\Ydrani\Programmation\Cshp\Mediatek86_Documentaire

Framework

.NET Framework 4.7.2



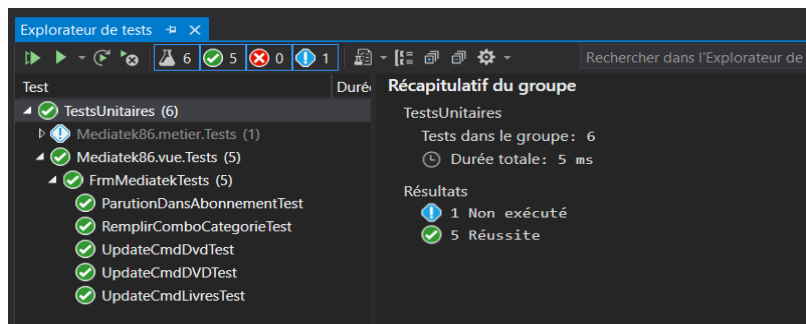
```
namespace Mediatek86.vue.Tests
{
    [TestClass()]
    public class FrmMediatekTests
    {
        private DateTime dateFin = DateTime.Parse("14/03/2008");
        private DateTime dateParu = DateTime.Parse("12/03/2008");
        private DateTime dateCommande = DateTime.Parse("10/03/2008");
    }
}
```

```

    private bool ParutionDansAbonnement(DateTime dateCommande, DateTime
dateFin, DateTime dateParution)
    {
        if ((dateCommande < dateParution && dateParution < dateFin) ||
dateParution == DateTime.MinValue)
        {
            return true;
        }
        return false;
    }

    [TestMethod()]
    public void ParutionDansAbonnementTest()
    {
        Assert.AreEqual(true, ParutionDansAbonnement(dateCommande,
dateFin, dateParu));
        Assert.AreEqual(false, ParutionDansAbonnement(dateParu, dateFin,
dateCommande));
        Assert.AreEqual(false, ParutionDansAbonnement(dateCommande,
dateParu, dateFin));
    }

```



Création du trigger de contrôle de partition sur héritage de commande

Une partition sur héritage de commande implique qu'une commande est soit une commandedocument soit un abonnement.

```
CREATE TRIGGER `trigInsertCommandes`  
BEFORE INSERT ON `commande` FOR EACH ROW  
BEGIN  
    DECLARE idCmdDoc INT;  
    DECLARE idCmdRevue INT;  
  
    SELECT Count(id) into idCmdDoc FROM commandedocument WHERE id =  
NEW.id;  
    SELECT Count(id) into idCmdRevue FROM abonnement WHERE id = NEW.id;  
  
    IF(idCmdDoc > 0 || idCmdRevue > 0) THEN  
        SIGNAL SQLSTATE "45000"  
        SET MESSAGE_TEXT = "opération impossible doublon idCmd" ;  
    END IF;  
  
END ;
```

Ajout de la fonctionnalité Alerte de fin d'abonnement

Procédure stockée : la liste des revues dont l'abonnement se termine dans moins de 30 jours.

```
BEGIN  
    DECLARE idRevues VARCHAR(5);  
    DECLARE grpStr VARCHAR(255);  
  
    SELECT DISTINCT idRevue,  
    GROUP_CONCAT( DISTINCT idRevue)  
    INTO idRevues, grpStr  
    FROM abonnement  
    WHERE dateFinAbonnement <= DATE_ADD(NOW(), INTERVAL 1 MONTH);  
  
    IF(grpStr = null) THEN  
        SET grpStr = "";  
    END IF;  
  
    RETURN grpStr;  
END
```

Ajout dans le controleur

```
/// <summary>
/// Methode du controleur d'appel de la methode GetAllRevues
/// Getter sur la chaine lesFinAbo
/// </summary>
/// <returns>Chaine de liste d'idRevue</returns>
0 référence
public String GetEndingAbonnement()
{
    return lesFinAbo;
}

/// <summary>
/// Methode du controleur accédant à la méthode GetEndingTitleDate
/// Permet la récupération de la liste des revues en fin d'abonnements
/// </summary>
/// <returns>La chaine d'alerte à afficher</returns>
1 référence
public String GetEndingTitleDate()
{
    Dictionary<String, String> dictFinAbo = Dao.GetEndingTitleDate(lesFinAbo);
    string strList = "";

    foreach (var item in dictFinAbo)
    {
        strList += item.Key + " termine le : " + item.Value + ".\n";
    }
    return strList;
}
```

Ajout dans Dao

```

/// <summary>
/// Methode permettant de lancer une requete SQL SELECT d'appel de fonction stockée
/// </summary>
/// <returns>Chaine d'idRevues des revues en fin d'abonnement </returns>
1 référence
public static String GetEndingAbonnement()
{
    string strFinAbo = "";

    string req = "SELECT fctEndingAbo() as id;";

    BddMySQL curs = BddMySQL.GetInstance(connectionString);
    curs.RegSelect(req, null);

    while (curs.Read())
    {
        strFinAbo += (string)curs.Field("id");
    }
    curs.Close();

    return strFinAbo;
}

```

Ajout dans la vue

```

/// <summary>
/// Constructeur de la Classe
/// Récupère l'instance du controleur de la Classe Controle
/// </summary>
/// <param name="controle"></param>
1 référence
internal FrmMediatek(Controlle controle)
{
    InitializeComponent();

    this.controle = controle;
    MessageBox.Show(controle.GetEndingTitleDate(), "Information: Fins Abonnements");
}

```

```
public static Dictionary<String, String> GetEndingTitleDate(String strIdRevues)
{
    string[] lstlesRevues = strIdRevues.Split(',');
    DateTime dateFinAbonnement ;
    Dictionary<String, String> dictFinAbo = new Dictionary<String, String>();

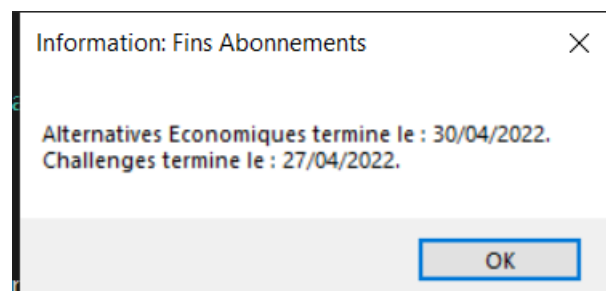
    foreach (var item in lstlesRevues)
    {
        string req = "SELECT Distinct titre, dateFinAbonnement FROM abonnement a JOIN document d ON (d.id = a.idRevue) ";
        req += " WHERE idRevue = @liste GROUP BY titre ";

        Dictionary<string, object> parameters = new Dictionary<string, object>
        {
            { "@liste", item}
        };

        BddMySQL curs = BddMySQL.GetInstance(connectionString);
        curs.ReqSelect(req, parameters);

        while (curs.Read())
        {
            string titre = (string)curs.Field("titre");
            dateFinAbonnement = (DateTime)curs.Field("dateFinAbonnement");

            dictFinAbo.Add(titre, dateFinAbonnement.ToShortDateString());
        }
        curs.Close();
    }
    return dictFinAbo;
}
```



Dépôt github

A l'issue de ce code, un [commit](#) a été effectué sur la plateforme GitHub dans le dépôt du projet.

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

Implémentation en cours

Mission 4 : mettre en place des authentifications

Objectifs

Cette mission a pour objectif la mise en place des restrictions d'accès à certaines fonctionnalités de l'application, suivant l'habilitation de l'utilisateur de l'application.

- **Dans la base de donnée**
 - Ajouts des table Utilisateur et une table Service sachant que chaque utilisateur ne fait partie que d'un service.
 - Remplir les tables d'exemples pour contrôler des tests
- **Dans la vue**
 - Fenêtre d'authentification que le contrôleur ouvre en première
 - 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
- **Dans le contrôleur**
 - suivant le type de personne authentifiée, empêcher certains accès en rendant invisibles ou inactifs certains onglets ou objets graphiques
 - L'alerte de fin d'abonnement n'apparaisse que pour les personnes concernées (qui gèrent les commandes).

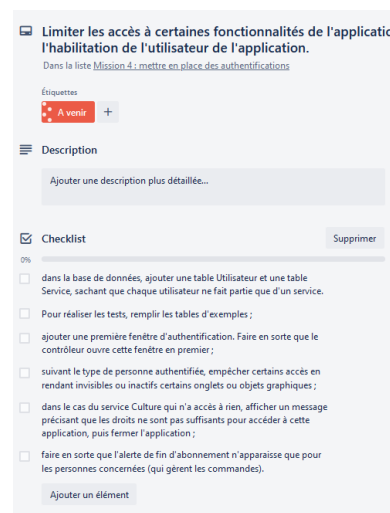


Figure 29: Suivi Trello Mission 4

Modifications dans la base de données

Ajout de la table service

```
DROP TABLE IF EXISTS service;
CREATE TABLE service (
  idService int PRIMARY KEY NOT NULL AUTO_INCREMENT,
  label varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
INSERT INTO service(label)
VALUES
('administratif'),
('prêts'),
('culture'),
('administrateur');
```

Ajout table utilisateur

```
DROP TABLE IF EXISTS `utilisateur`;
CREATE TABLE IF NOT EXISTS `utilisateur` (
  `idUser` int(11) NOT NULL AUTO_INCREMENT,
  `idService` int(3) NOT NULL,
  `pwd` varchar(255) NOT NULL,
  `login` varchar(255) NOT NULL,
  PRIMARY KEY (`idUser`),
  KEY `commande_ibfk_1` (`idService`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Sa clé étrangère vers idService

```
ALTER TABLE `utilisateur`
  ADD CONSTRAINT `commande_ibfk_1` FOREIGN KEY (`idService`) REFERENCES
`service` (`idService`);
COMMIT;
```

Exemple d'entrées test

```
INSERT INTO utilisateur(idService, pwd, login)
VALUES (1,'ppfdsfhtrts3','adflog1'),
(2,'rts3fdsfhtrt','prtlog1'),
(3,'pZpfgsfhtrts3','cltlog1'),
(4,'ppfdsfhtrts3','adminlog1'),
(1,'3Eh63Gfe4htC','adflog2'),
(2,'5gsfhdemGT5G','prtlog2'),
(3,'r4gfpfdsfht','cltlog2'),
(4,'slogr4gfdgdA','adminlog2');
```

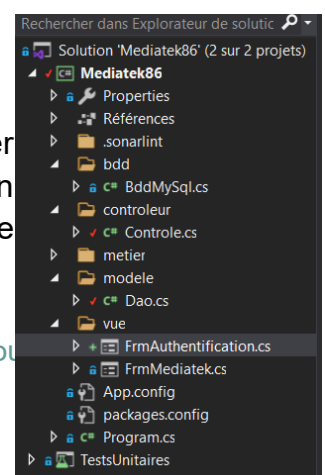
Authentifications

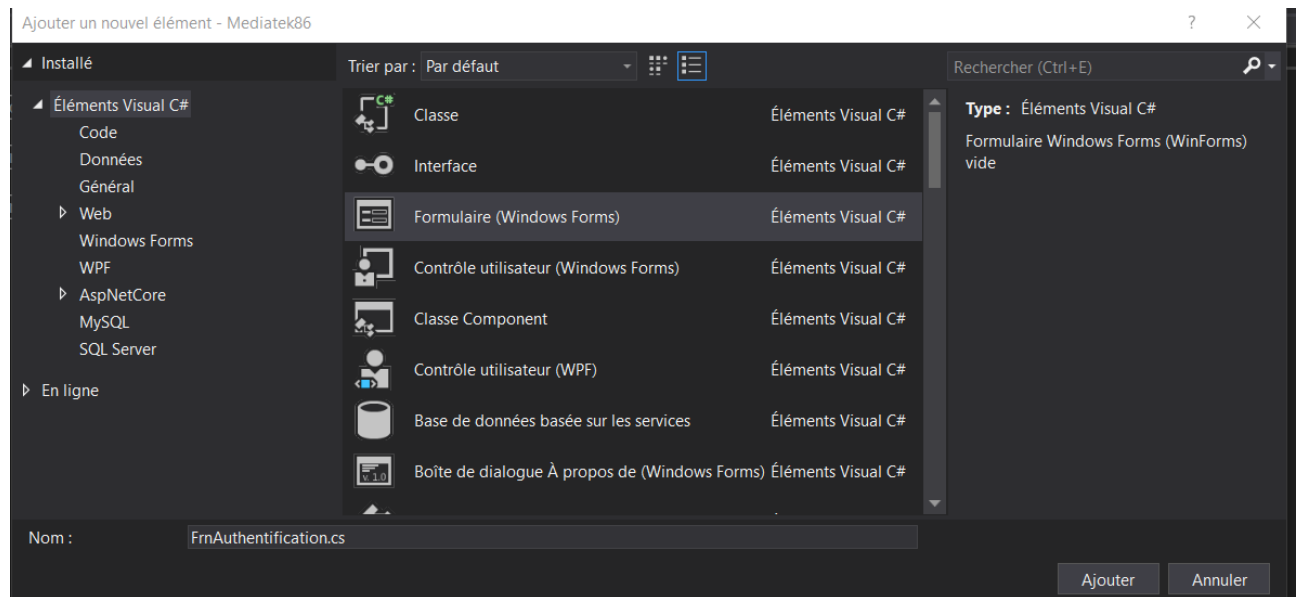
```
admf_pwd    || admf_log || Administratif
prt_pwd     || prt_log  || Prets
clt_pwd     || clt_log  || Culture
admin_pwd   || admin_log || Administrateur
```

Mise en place fenêtre d'authentification

Modification du package vue

Dans l'IDE, dans le package vue, sur un clique droit sélectionner ajouter un nouvel element, choisir formulaire Windows Forms. Un nouveau formulaire, nommée FrmAuthentification est créé dans le package vue.





Qualité du code

Suivi de projet

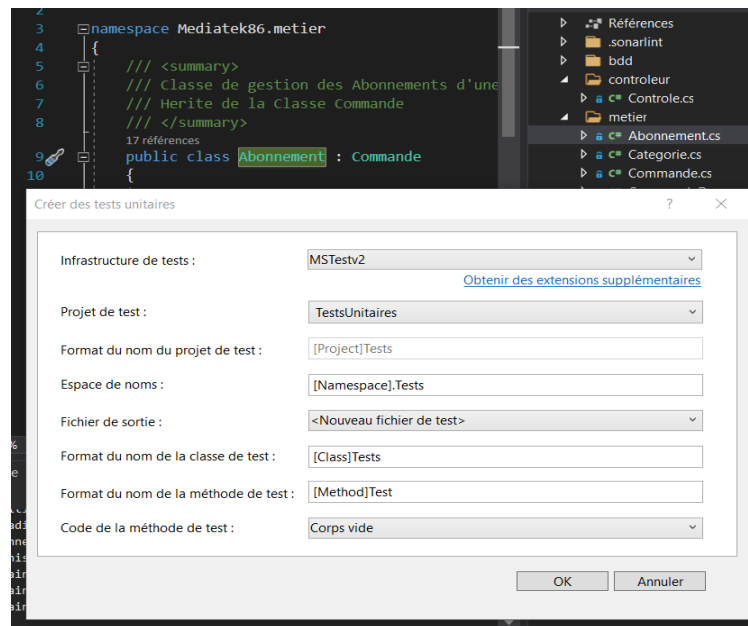
La mission 4 terminée, ses étiquettes dans ce tableau modifiées pour être mises en violet pour signifier leurs clôtures, nous obtenons alors le tableau de suivi visible. La mission suivante a pour objectif de mettre en place la qualité du code produit, ainsi une liste de 4 quatre ont été établis. Ainsi les différentes étapes établies sont les suivantes :

- Contrôler la qualité du code avec SonarLint
- Création des tests unitaires
- Création de la documentation technique de l'application

Tests unitaires

Classe FrmMediatek

Classe AbonnementTest



Contrôler la qualité du code avec SonarLint et SonarQube

SonarLint

SonarLint étant un plugin pouvant être intégré sur Visual Studio, la grande majorité des morceaux de codes mal écrits ou d'erreurs de sécurité peuvent être rapportés directement dans un console dédiée.

Ainsi, plusieurs alertes comme celle illustrée en *figure 67* ont été directement gérées. Sur cette figure, l'alerte signale que la méthode d'écoute sur un bouton avec `setOnClickListener()` doit être mise sous forme de fonction lambda.

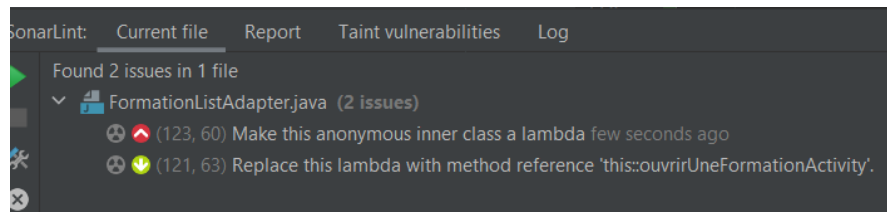


Figure 30: Exemples d'alertes traitées avec le plugin SonarLint

Aussi, pour certaines de ces fonctions lambda, la seconde alerte demande de modifier certaines fonctions lambda par un appel de méthode référencé. C'est pourquoi l'ensemble des méthodes appelant **setOnClickListener()** ont été mise soit sous cette dernière forme, soit sous forme de fonction lambda. Ceci est principalement visible dans la vue **FormationListAdapter**.

Néanmoins SonarQube permet d'apporter une plus grande précision et éventuellement de relever davantage d'erreurs de code, notamment certaines relevant de la sécurité.

Corrections de SonarQube

Il est alors possible de constater de nouvelles alertes dont 8 notées comme alertes de sécurité et 10 de mauvaises utilisation du code. Néanmoins, la majorité de ces erreurs peuvent être considérées comme des faux positifs puisqu'elles relèvent de obsolescence de la classe **AsyncTask** et de ses méthodes.

Figure 31: Exemples d'alertes relevés par SonarQube.

La résolution de ces alertes se fait individuellement, en cliquant sur le message de l'erreur. Aussi, il est possible d'avoir une explication de l'alerte avant d'estimer s'il faut la réparer ou non. Ce choix se fait par l'intermédiaire du bouton Open visible sur chacun des messages de la figure 69. Il est possible de constater qu'une erreur relevant de l'utilisation de l'extension Comparable est détectée. Celle-ci demande de réécrire aussi la méthode **equals()** pour respecter la mise en place avec **CompareTo()**. N'ayant ici besoin que de **compareTo()** cette alerte est notée comme « à ne pas réparer » puisque seule la comparaison des valeurs nous intéresse dans l'usage de cette méthode.

Concernant les alertes de sécurité, la plus importante relevée concerne l'activation du langage **Javascript** dans la vue **VideoActivity** permettant ainsi d'avoir accès directement aux vidéos présentes sur la plateforme Youtube. Cette alerte informe que cela peut présenter une faille de sécurité de type **XXS** permettant d'injecter du contenu dans une page web, cf [figure 71](#).

Cette ligne étant exécutée lors de la méthode `init()` de la classe `VideoActivity` appelé par `onCreate()`, la correction de cette alerte a demandé l'écriture d'une nouvelle méthode `onDestroy()` qui va alors s'exécuter à la fermeture de l'activity. Cette dernière demande alors la désactivation de l'utilisation de Javascript en mettant la valeur `setJavaScriptEnabled` sur `false`, cf [figure 72](#).

Finalement, il a été possible d'obtenir les notes suivantes :

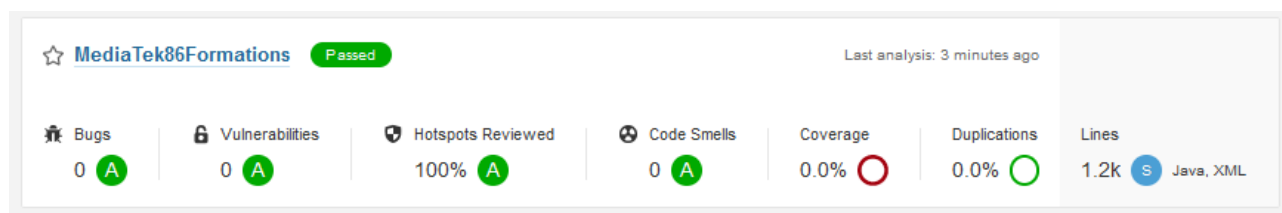


Figure 32: Mesures des différents points d'alerte de SonarQube

Doctechnique

La documentation technique générée avec Doxyfile est visible à l'adresse [suivante](#).

Déploiement

Création de l'installateur

Setup Wizard

La création de l'installateur de l'application a été effectué avec l'extension « Microsoft Visual Studio Installer Project » récupérable depuis le gestionnaire d'extension de l'IDE. Sa mise en place demande d'ajouter un nouveau projet type « Setup Wizard » dans la solution ici sous le nom de « Setup ».

Une fois le projet créé, une fenêtre de configuration « Choose a project type » s'ouvre à chaque étape, les options suivantes ont été sélectionnées :

- Create a setup for a Windows application
- Sortie principale from Mediatek86



Figure 33: Etapes de configuration de setup wizard

Une fois le projet créé, il faut configurer le projet Setup_MediatekGest_Documentaire.

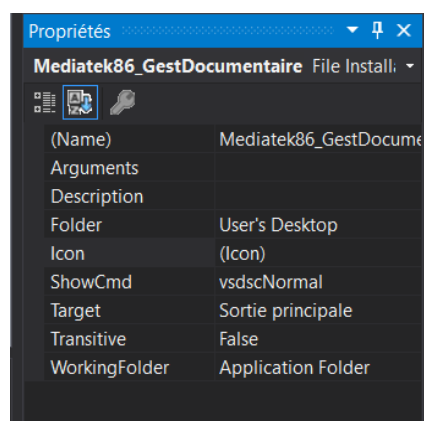
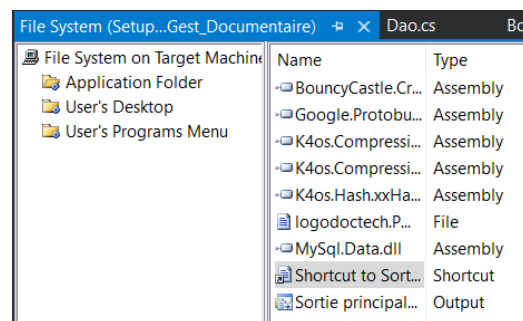
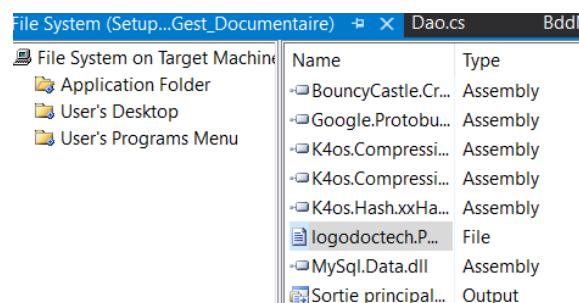
Configurer Setup_MediatekGest_Documentaire

Dans le dossier Application Folder, un fichier logodoctech.ico a été ajouté, ce dernier permet de créer une icône personnalisée.

Enfin deux raccourcis sont créés depuis l'output « **Sortie principale from Mediatek** » qui ont été glissés respectivement dans les dossiers « **User's Desktop** » et « **User's Programs Menu** ».

Dans les propriétés de ces raccourcis, il est possible de les renommer afin de leur donner un nom correspondant au nom de l'application, **MediatekGest_Documentaire**.

Enfin dans la propriété **Icon**, il est possible de sélectionner le fichier ico préalablement déposé dans le dossier **Application Folder**.



ico

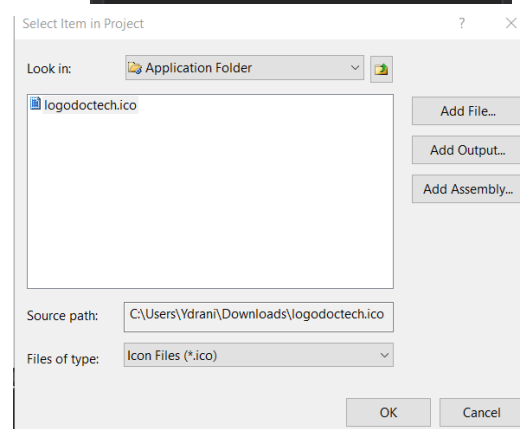


Figure 35: Mise en place des propriétés des raccourcis

Ainsi, nous avons dans les dossiers User's Desktop et User's Programs Menu le contenu suivant :

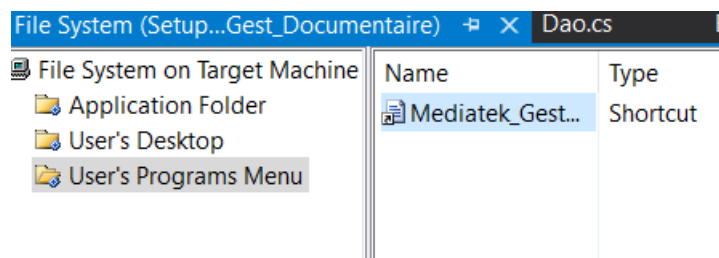


Figure 36: Exemple de contenu de setup

Enfin, il est possible de modifier les propriétés **Author** et **Manufacturer** du projet Setup. Cette dernière propriété permet de choisir le nom du dossier d'installation.

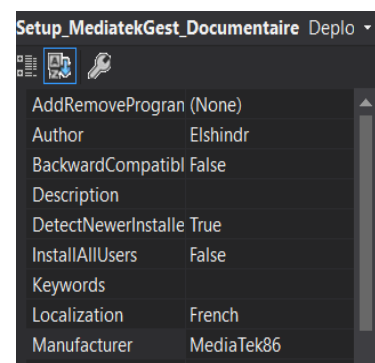


Figure 37: Propriétés du projet Setup_MediatekGest_Documentaire

Enfin dans l'onglet de l'IDE « générer », certaines modifications ont été effectués dans le gestionnaire de configurations. L'ensemble des checkbox de la colonne Générer ont été cochés pour les configurations de solutions actives Debug et Release.

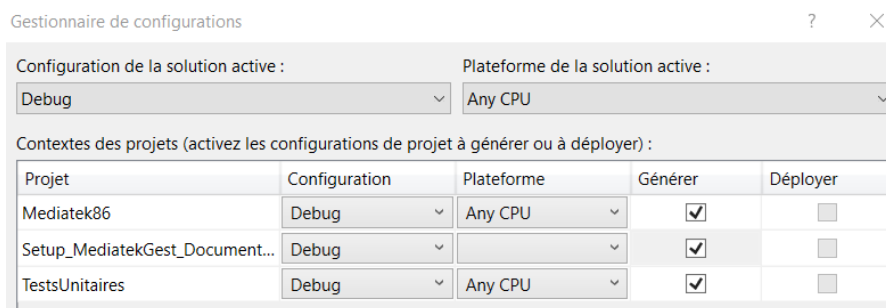


Figure 38: Gestionnaire de configuration de génération de projet

Enfin, il suffit de cliquer sur la solution du projet, puis de sélectionner « générer la solution ». Ainsi, l'installateur est créé dans le dossier du projet de la solution. Il convient de cliquer sur le fichier en **.msi** pour lancer l'installation.

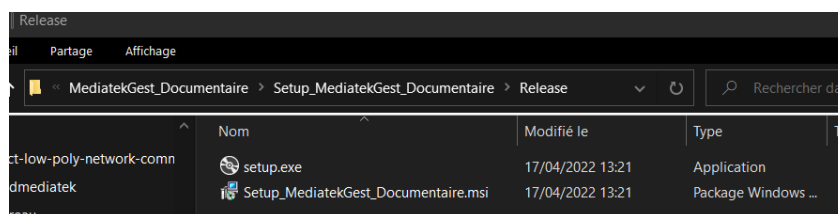


Figure 39: Chemin du dossier vers l'installateur de l'application

Mise en ligne de la base de données

Heroku

La plateforme de déploiement choisie est [Heroku](#) qui permet la mise en ligne de plateforme web. Pour mettre la base de donnée locale en ligne, il suffit de créer gratuitement un compte, de créer une nouvelle application et de le lier au compte du dépôt GitHub.



Figure 40: Fenêtre de liaison au dépôt GitHub pour le déploiement sur Heroku

JawsDB MySQL

La dernière étape consiste à mettre en place la base de donnée en ligne, pour cela la plateforme **Heroku** dispose de plusieurs add-ons dont **JawsDB MySQL** qui sera utilisé à cette intention. Pour cela, il suffit d'aller dans la liste des add-ons afin de trouver l'add-on, puis de choisir la formule concordant aux besoins de l'application ici « **Kitefin Shared** » qui est gratuite et suffisante pour les besoins actuels de l'application.

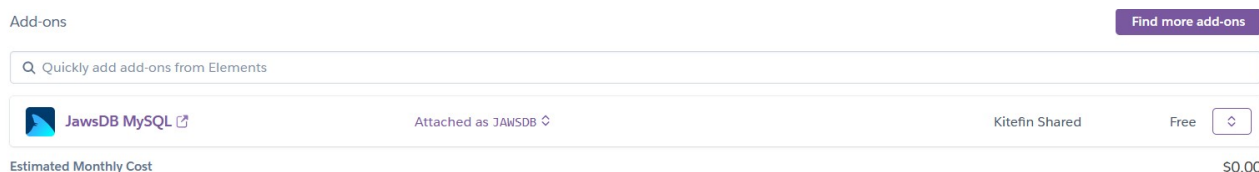


Figure 41: Fenêtre Heroku des addons

En cliquant sur «**JawsDB MySQL**», une nouvelle s'ouvre dans laquelle il est possible de trouver les différentes informations de connexion à la base de donnée en cliquant sur le nom de la base mise à disposition, cf *figure*.

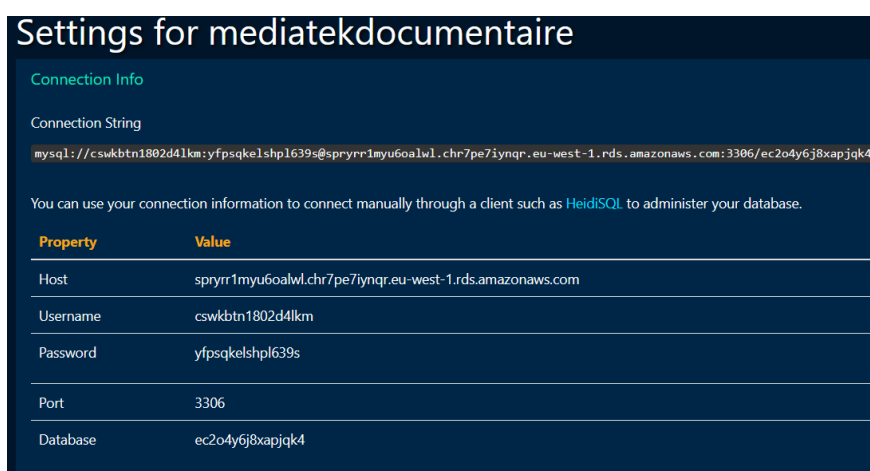


Figure 42: Fenêtre de ClearDB contenant les informations de connexion

Ces informations sont à remplacer avec celles qui définissent l'accès local dans **Doa.cs**, cf *figure*

```
private static readonly string server = "sprrrr1myu6oalwl.chr7pe7iynqr.eu-west-1.rds.amazonaws.com";
private static readonly string userid = "cswkbtn1802d4lkm";
private static readonly string password = "yfpsqkelshpl639s";
private static readonly string database = "ec2o4y6j8xapjqk4";
private static readonly string connectionString = "server=" + server + ";user id=" + userid + ";password=" + password + ";database=" + database + ";SslMode=no
```

Figure 43: Classe Doa.cs informations de connexion à la base de donnée distante

Enfin, il faut remplir la nouvelle base de donnée. Afin d'y accéder avec phpMyAdmin il est nécessaire de modifier son fichier config.inc.php, puis d'aller en bas du fichier afin de rajouter les données de connexion afin que le nouveau serveur soit disponible lors de la connexion à phpMyAdmin.

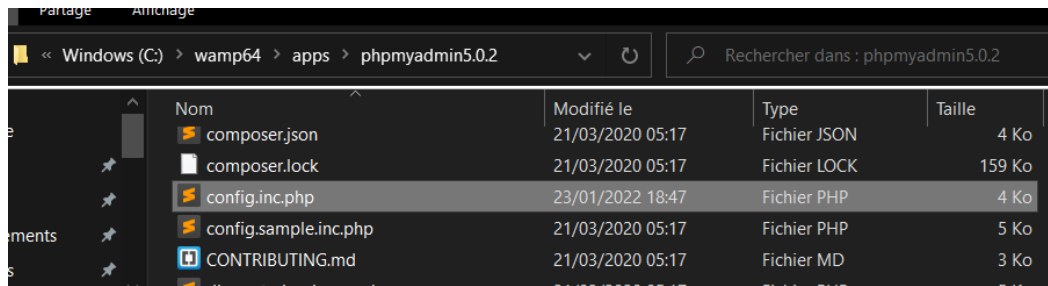


Figure 44: Localisation du fichier config.inc.php

Ainsi, il faut rajouter le fichier config.inc.php le contenu suivant :

```
$i++;
$config['Servers'][$i]['host'] = 'spryr1myu6oalwl.chr7pe7iynqr.eu-west-1.rds.amazonaws.com'; //hostname and port
$config['Servers'][$i]['user'] = 'cswkbtn1802d4lkm'; //user name
$config['Servers'][$i]['password'] = 'yfpsqkelshpl639s'; //password
$config['Servers'][$i]['auth_type'] = 'config'; //config
```

Après avoir sauvegardé ce fichier, il est alors possible d'accéder au serveur de la base de donnée via l'interface de connexion de phpMyAdmin. Enfin, la base de donnée peut être rempli comme précédemment via le menu d'importation de script SQL après avoir exporté la base modifiée, ici trouvable à ce lien.

Figure 45: Interface de connexion à phpMyAdmin sur le nouveau serveur

Bilan sur les objectifs atteints

Finalités

Actuellement, l'évolution de l'application permet la gestion des commandes des différents types de documents : livres, dvd et revue, ainsi que l'apparition d'une alerte énumérant les fins d'abonnements. Aussi, la mise en place d'une authentification par le biais d'une fenêtre de connexion permet aux utilisateurs de se connecter à l'application et d'avoir uniquement accès aux éléments nécessaires à leur service d'affection. Ces évolutions semblent fonctionnelles et respectueuses de l'expression des besoins.

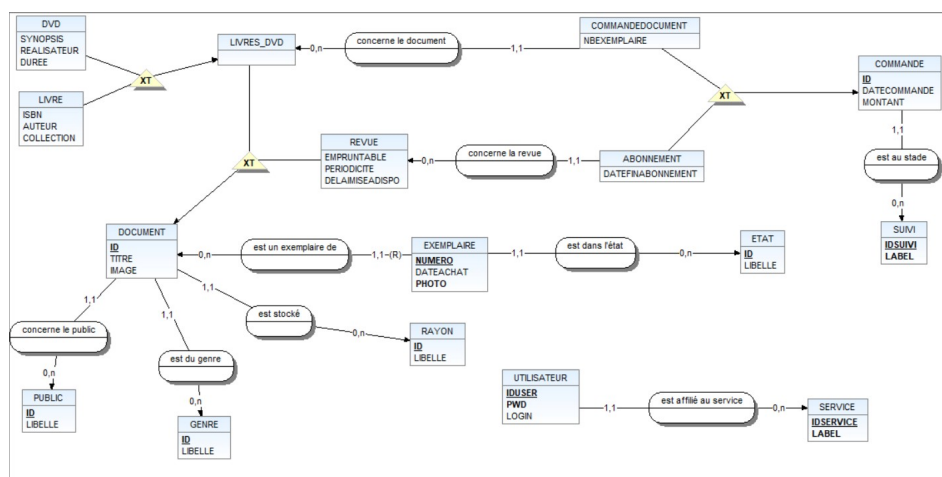


Figure 46: Schéma UML de la base de données finale

Liste des compétences couvertes (B1, B2, B3)

B1 Gérer le patrimoine informatique

- Recenser et identifier les ressources numériques
- Exploiter des référentiels, normes et standards adoptés par le prestataire informatique
- Vérifier les conditions de la continuité d'un service informatique

B1 Répondre aux incidents et aux demandes d'assistance et d'évolution

- Traiter des demandes concernant les services réseau et système, applicatifs
- Traiter des demandes concernant les applications

B1 Travailler en mode projet

- Analyser les objectifs et les modalités d'organisation d'un projet
- Planifier les activités

- Évaluer les indicateurs de suivi d'un projet et analyser les écarts

B1 Mettre à disposition des utilisateurs un service informatique

- Réaliser les tests d'intégration et d'acceptation d'un service
- Déployer un service
- Accompagner les utilisateurs dans la mise en place d'un serv

B1 Organiser son développement professionnel

- Mettre en place son environnement d'apprentissage personnel
- Mettre en œuvre des outils et stratégies de veille informationnelle
- Gérer son identité professionnelle
- Développer son projet professionnel

B2 Concevoir et développer une solution applicative

- Participer à la conception de l'architecture d'une solution applicative
- Modéliser une solution applicative
- Exploiter les ressources du cadre applicatif (framework)
- Identifier, développer, utiliser ou adapter des composants logiciels
- Exploiter les technologies Web pour mettre en œuvre les échanges entre applications, y compris de mobilité
- Utiliser des composants d'accès aux données
- Intégrer en continu les versions d'une solution applicative
- Réaliser les tests nécessaires à la validation ou à la mise en production d'éléments adaptés ou développés
- Rédiger des documentations technique et d'utilisation d'une solution applicative
- Exploiter les fonctionnalités d'un environnement de développement et de tests

B2 Assurer la maintenance corrective ou évolutive d'une solution applicative

- Recueillir, analyser et mettre à jour les informations sur une version d'une solution applicative
- Évaluer la qualité d'une solution applicative
- Analyser et corriger un dysfonctionnement
- Mettre à jour des documentations technique et d'utilisation de solution applicative

B2 Gérer les données

- Développer des fonctionnalités applicatives au sein d'un système de gestion de base de données (relationnel ou non)
- Concevoir ou adapter une base de données
- Administrer et déployer une base de données

B3 Assurer la cybersécurité d'une solution applicative et de son développement

- Participer à la vérification des éléments contribuant à la qualité d'un développement informatique

- Prendre en compte la sécurité dans un projet de développement d'une solution applicative
- Mettre en œuvre et vérifier la conformité d'une solution applicative et de son développement à un référentiel, une norme ou un standard de sécurité