

DÉVELOPPEMENT .NET C#

TP NOTÉ

GESTION D'UNE BIBLIOTHÈQUE EN C# AVEC SÉRIALISATION ET CRYPTAGE

Maxence LAURENT

2025-2026



TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

Objectif :

Développer une application console (ou graphique à vous de voir) en C# permettant de gérer une bibliothèque avec des livres et des utilisateurs. Les livres seront rangés par catégories, et l'application devra supporter la persistance des données via la sérialisation (binaire et XML). Les données doivent être protégées par cryptage.

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

1. Socle applicatif

1.1 Modèle de données

Créez deux classes : Livre et Utilisateur, et une classe Categorie pour gérer les catégories de livres.

Classe Livre :

- Titre : Le titre du livre.
- Auteur : Le nom de l'auteur.
- DateDePublication : La date de publication.
- ISBN : Le numéro ISBN unique du livre.
- Categorie : La catégorie dans laquelle le livre est classé (e.g., Fiction, Science, Histoire).
- DateAjout : Date d'ajout du livre dans le système.

Classe Utilisateur :

- Nom : Le nom de l'utilisateur.
- Prenom : Le prénom de l'utilisateur.
- Email : L'email de l'utilisateur.
- DateInscription : La date d'inscription de l'utilisateur.
- LivresEmpruntés : Liste des livres empruntés par l'utilisateur.

Classe Categorie :

- Nom : Nom de la catégorie (e.g., "Science", "Histoire").
- Livres : Liste des livres associés à cette catégorie.

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

2. Sérialisation des données

2.1 Sérialisation binaire et XML

Implémentez une méthode de sérialisation pour la gestion des livres et des utilisateurs

- **Sérialisation XML** en utilisant XmlSerializer.

2.2 Gestion des fichiers

- Utilisez le design pattern **Factory** pour déterminer si les données seront sérialisées en XML.
- Sauvegarder les fichiers dans un répertoire spécifique, par exemple, dans C:\Users\<NomUtilisateur>\Documents\Bibliotheque.
- Gérez les erreurs de chargement si les fichiers n'existent pas ou sont corrompus, avec un message d'erreur informatif.

Bonus : Utilisez l'identité Windows pour déterminer dynamiquement le nom du fichier (par exemple : "Bibliotheque_<NomUtilisateur>.xml").

2.3 Gestion de mot de passe

- Lors du chargement du fichier de bibliothèque, demandez un mot de passe pour l'accès.
- Si un mot de passe erroné est entré 3 fois, supprimez le fichier pour garantir la sécurité.

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

3. Cryptage des données

3.1 Cryptage et Décryptage

- Protégez les fichiers de bibliothèque (livres, utilisateurs, catégories) en utilisant **CryptoStream** pour effectuer un cryptage réversible des fichiers.
- Demandez à l'utilisateur de fournir une clé de cryptage lors de l'enregistrement et du chargement des fichiers. Si aucune clé n'est fournie, utilisez l'identifiant SID de l'utilisateur Windows.

3.2 Gestion des erreurs de cryptage

- Implémentez un système de gestion des erreurs pour éviter que l'application ne plante en cas de problème de cryptage/décryptage.
- Si une erreur de cryptage survient, affichez un message d'erreur et proposez à l'utilisateur de tenter de nouveau l'opération.

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

4. Mise en œuvre

1. Création des classes de données

- Livre : Classe représentant un livre dans la bibliothèque.
- Utilisateur : Classe représentant un utilisateur de la bibliothèque.
- Catégorie : Classe représentant une catégorie de livres.

Méthodes : Créez des méthodes pour ajouter des livres, des utilisateurs, des catégories, emprunter des livres, et pour sérialiser/désérialiser les données.

2. Application Console (ou graphique si vous le souhaitez)

- Créez un menu interactif permettant :
 - D'ajouter et lister des livres, des utilisateurs, des catégories.
 - D'emprunter et rendre des livres.
 - De sérialiser et désérialiser les données de la bibliothèque.
 - D'appliquer le cryptage pour protéger les fichiers.

3. Implémentation de la sérialisation et du cryptage

- Implémentez des méthodes pour la sérialisation des objets et pour la gestion du cryptage des fichiers.

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

5. Extension du TP (Facultatif)

Ajouter des fonctionnalités comme :

- Importer/exporter des livres depuis/vers un fichier texte ou CSV.
- Implémenter une interface graphique avec Windows Forms ou WPF pour rendre l'application plus interactive.
- Ajouter un mécanisme pour générer des rapports sur les livres empruntés et les utilisateurs actifs.
- Implémenter des tests unitaires pour valider le bon fonctionnement des différentes méthodes. (avec la librairie de votre choix)

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

6. Remarques importantes :

- **Test de sérialisation** : Testez la sérialisation sur des objets Livre, Utilisateur et Catégorie pour vérifier que les fichiers sont correctement sauvegardés et chargés.
- **Sécurité** : Assurez-vous que la gestion des mots de passe et des clés de cryptage est sécurisée et ne permet pas des accès non autorisés.

TP : Gestion d'une bibliothèque en C# avec sérialisation et cryptage

- Le rendu du projet sera uniquement une solution visual studio à envoyer à l'adresse suivante : Maxence.lau@gmail.com
- Si la taille du projet compressé dépasse 7Mo, optez pour un stockage en ligne et envoyez le lien.
- L'archive compressée devra être au format PRENOM_NOM
- Présentez brièvement dans le mail votre solution, choix techniques, problèmes rencontrés et résolution des problèmes.
- La solution contiendra au minimum 3 projets :
 - Une application console(ou WPF ou Winform)
 - Le projet de data
 - Le projet de serialization
- Bien que construite dans un premier temps autour d'une interface de type console texte, votre application doit être compatible avec une interface de type application graphique.
- **Le tout est à rendre pour le 15 Février !!**
- Pensez à commenter votre code et à séparer les classes correctement.
- Si vous avez des questions n'hésitez pas à prendre contact par mail.