

**RAPPORT DE PROJET**

Système d’information d’une Bibliothèque

**Rédigé par**

Soldi Ahmed Amine

**Encadré par**

Prof. Nofisse Jihad

###### Année Universitaire

2021/2022

**Remerciement**

Je remercie l’équipe pédagogique de l’ESMA

et tout ce qui a contribué à la réalisation de ce projet

**Sommaire**

[Problématique 5](#_bookmark0)

[Objectifs 5](#_bookmark1)

[Spécifications 6](#_bookmark2)

[Structure et organisation 7](#_bookmark3)

[Modélisation 8](#_bookmark4)

[MCD 8](#_bookmark5)

[MLD 9](#_bookmark6)

[Outils utilisés 10](#_bookmark7)

[Langages de programmation 11](#_bookmark8)

[T-SQL 12](#_bookmark9)

[Captures d’écran 30](#_bookmark10)

**Introduction**

# Problématique

#### La bibliothèque est un établissement qui conserve les livres, les dictionnaires, et d'autres matérielles pour la lecture, l’étude, et la référence. Elle permet aussi aux adhérents d'emprunter son contenu. Une bibliothèque peut contenir des milliers d'ouvrages et garde un registre des emprunts pour faire le suivi des livres rendus et appliquer des pénalités en cas de retards. Ce grand volume d'information hétérogènes rend le système d'information un outil indispensable pour la bonne gestion de l’établissement.

# Objectifs

#### ✔ Faciliter les taches administratives.

#### ✔ Rendre l’information accessible à tout moment et à tout endroit.

#### ✔ Traitement et manipulation des données automatique et rapides.

#### ✔ Minimiser les erreurs humaines.

**Cadre Théorique**

# Spécifications

### Stockage et sauvegarde des données

### Base de données

### SGBDR

### Facile et intuitif à utiliser

### Interface graphique

### Accessibilité à l’information depuis n’importe quel machine ou poste de travail

### Centralisation des données

### Interconnexion et réseau

### Sécurité et authentification

### Agent d’authentification

# Structure et organisation



*les flèches indiquent le flux des données*

**Utilisateur :** lit et saisit les données. Un utilisateur qui est un Admin possède un identifiant qui lui donne l’accès total à l’application.

**Application :** collecte et fait le traitement des données. Elle s ‘occupe aussi de l’affichage et présentation des informations.

**SGBDR :** Système de Gestion de Base de Données Relationnelles, logiciel qui joue le rôle d’intermédiaire entre l’application et la base de données. Il stocke les procédures, fonctions et vues invoqués par l’application.

**BD :** Base de Données relationnelle qui stocke les données sous forme de tableaux a deux dimensions.

# Modélisation

## MCD

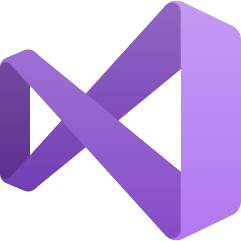
## MLD

#### Livre(ISBN, Titre, Edition, Categorie, Langue, Annee, Pages, LectureSeule, CoutReparation) Auteur(NumAuteur, NomAuteur, PrenomAuteur) Exemplaire(NumExemplaire, Etat, #ISBN) Adherent(NumAdherent, NomAdherent, PrenomAdherent, DateNaissance, Email, Adresse) Suspension(NumSuspension, DateDebut, DateFin, RaisonSuspension, #NumAdherent)

#### Penalite(NumPenalite, Montant, RaisonPenalite, #NumAdherent)

#### Ecrivain(#NumAuteur, #ISBN) Pret(#NumAdherent, #NumExemplaire, DatePret, DateConclusion, Conclusion)

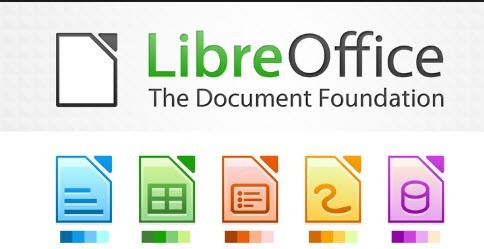
# Outils utilisés



***Visual Studio****, IDE choisi pour ce projet* ***SQL SERVER MANAGEMENT STUDIO****,*

##### outil de gestion et configuration des bases de

##### données



***LibreOffice****, pour le besoin de communication*

# Langages de programmation

**C#**

##### Langage utilisé pour l’application

**T-SQL**

##### Langage utilisé pour la création des procédures et fonctions dans le coté serveur

**Cadre Pratique**

**T-SQL**

CREATE DATABASE Bibliotheque;

USE Bibliotheque;

--Tables

CREATE TABLE Livre (

ISBN NVARCHAR(20) PRIMARY KEY CHECK (ISBN LIKE ' '), Titre NVARCHAR(100) NOT NULL,

Edition NVARCHAR(50),

Categorie NVARCHAR(20) CHECK (Categorie IN ('Roman', 'Histoire',

'Sciences &

Maths', 'Bande Dessiné', ', 'Philosophie',

& Économie', 'Langues & Cultures')) NOT NULL, Langue NVARCHAR(20) NOT NULL,

Annee SMALLINT NOT NULL, Pages INT NOT NULL,

LectureSeule BIT NOT NULL, CoutReparation MONEY NOT NULL,

'Biographie 'Politique

Rayon SMALLINT

);

CREATE TABLE Auteur (

NumAuteur INT IDENTITY(100,1) PRIMARY KEY, NomAuteur NVARCHAR(50) NOT NULL

);

CREATE TABLE Adherent (

NumAdherent INT IDENTITY(200,1) PRIMARY KEY, NomAdherent NVARCHAR(20) NOT NULL,

PrenomAdherent NVARCHAR(20) NOT NULL, DateNaissance DATE NOT NULL,

Email NVARCHAR(255) NOT NULL, Adresse NVARCHAR(255) NOT NULL

);

CREATE TABLE Exemplaire (

NumExemplaire INT IDENTITY(1000,1) PRIMARY KEY,

ISBN NVARCHAR(20) FOREIGN KEY REFERENCES Livre(ISBN) ON DELETE CASCADE ON UPDATE CASCADE,

Etat NVARCHAR(20) CHECK (Etat IN ('Disponible', 'Perdu', 'Endommagé', 'Emprunté')),

);

CREATE TABLE Ecrivain

(

NumAuteur INT FOREIGN KEY REFERENCES Auteur(NumAuteur) ON DELETE

CASCADE ON UPDATE CASCADE,

ISBN NVARCHAR(20) FOREIGN KEY REFERENCES Livre(ISBN) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT PK\_Ecrivain PRIMARY KEY(NumAuteur, ISBN)

);

CREATE TABLE Pret (

NumAdherent INT FOREIGN KEY REFERENCES Adherent(NumAdherent) ON DELETE CASCADE ON UPDATE CASCADE,

NumExemplaire INT FOREIGN KEY REFERENCES Exemplaire(NumExemplaire) ON DELETE CASCADE ON UPDATE CASCADE,

DatePret DATE DEFAULT GETDATE(),

DateConclusion DATE,

Conclusion NVARCHAR(20) CHECK (Conclusion IN ('Rendu', 'Perdu', 'Endommagé', 'Delai depassé', NULL))

CONSTRAINT PK\_Pret PRIMARY KEY(NumAdherent, NumExemplaire, DatePret)

);

CREATE TABLE Penalite (

NumPenalite INT IDENTITY(1,1) PRIMARY KEY,

NumAdherent INT FOREIGN KEY REFERENCES Adherent(NumAdherent) NOT NULL, DatePenalite DATE DEFAULT GETDATE(),

Montant MONEY NOT NULL, RaisonPenalite NVARCHAR(200)

);

CREATE TABLE Suspension (

NumSuspension INT IDENTITY(1,1) PRIMARY KEY,

NumAdherent INT FOREIGN KEY REFERENCES Adherent(NumAdherent) NOT NULL, DateDebut DATE DEFAULT GETDATE() CHECK (DateDebut >= CONVERT(DATE,

GETDATE())),

DateFin DATE CHECK (DateFin > CONVERT(DATE, GETDATE())),

RaisonSuspension NVARCHAR(200)

); GO;

DROP TABLE Livre

DROP TABLE Exemplaire DROP TABLE Ecrivain DROP TABLE Pret

CREATE VIEW [Adherents] AS

SELECT NumAdherent Numero, NomAdherent Nom, PrenomAdherent Prenom, DateNaissance [Date Naissance], Email, Adresse

FROM Adherent GO;

CREATE VIEW [Mineurs] AS

SELECT \* FROM Adherents WHERE (DATEDIFF(year, [Date Naissance], GETDATE())

< 18) GO;

CREATE VIEW [Majeurs] AS

SELECT \* FROM Adherents WHERE (DATEDIFF(year, [Date Naissance], GETDATE())

>= 18) GO;

CREATE VIEW [Livres] AS

SELECT \* FROM Livre GO;

CREATE VIEW [Ecrivains] AS

SELECT L.Titre, L.ISBN, A.NomAuteur FROM Livre L, Auteur A, Ecrivain E WHERE L.ISBN = E.ISBN AND A.NumAuteur = E.NumAuteur

GO;

CREATE VIEW [Tout] AS

SELECT ISBN, Titre, Edition, Categorie, Langue, Annee, Pages, LectureSeule, Rayon FROM Livre

GO;

CREATE VIEW [Nouveautes] AS

SELECT \* FROM [Tout] WHERE Annee = YEAR(GETDATE()) GO;

CREATE VIEW [Litterature Anglaise] AS

SELECT \* FROM [Tout] WHERE Langue = 'Anglais' GO;

CREATE VIEW [Litterature Arabe] AS

SELECT \* FROM [Tout] WHERE Langue = 'Arabe' GO;

CREATE VIEW [Roman] AS

SELECT \* FROM [Tout] WHERE Categorie = 'Roman' GO;

CREATE VIEW [Histoire] AS

SELECT \* FROM [Tout] WHERE Categorie = 'Histoire' GO;

CREATE VIEW [Bande Dessine] AS

SELECT \* FROM [Tout] WHERE Categorie = 'Bande Dessiné' GO;

CREATE VIEW [Philosophie] AS

SELECT \* FROM [Tout] WHERE Categorie = 'Philosophie' GO;

--Types

CREATE TYPE TypeTableEcrivain AS TABLE (

NumAuteur INT PRIMARY KEY, ISBN NVARCHAR(20)

);

--Procedures

CREATE PROC sp\_ajouter\_exemplaire(@ISBN NVARCHAR(20)) AS

INSERT INTO Exemplaire(ISBN, Etat) VALUES (@ISBN, 'Disponible')

GO;

CREATE PROC modifier\_exemplaire(@NumExemplaire INT, @Etat NVARCHAR(20)) AS UPDATE Exemplaire

SET Rayon = @Rayon, Etat = @Etat

WHERE (NumExemplaire = @NumExemplaire)

GO;

CREATE PROC sp\_modifier\_etat\_exemplaire(@NumExemplaire INT, @Etat NVARCHAR(20)) AS

UPDATE Exemplaire SET Etat = @Etat

WHERE (NumExemplaire = @NumExemplaire)

GO;

CREATE PROC sp\_ajouter\_livre(@ISBN NVARCHAR(20),

@Titre NVARCHAR(100),

@Edition NVARCHAR(50), @Categorie NVARCHAR(50), @Langue NVARCHAR(50), @Annee SMALLINT,

@Pages INT, @LectureSeule BIT, @NbrExemplaires INT, @Rayon INT, @CoutReparation MONEY,

@TableEcrivains TypeTableEcrivain

READONLY) AS

IF EXISTS (SELECT \* FROM Livre WHERE ISBN = @ISBN) BEGIN

EXEC sp\_modifier\_livre @ISBN,

@Titre, @Edition, @Categorie, @Langue, @Annee, @Pages, @LectureSeule,

@NbrExemplaires, @Rayon, @CoutReparation, @TableEcrivains

RETURN

END

DECLARE @err INT = 0

BEGIN TRAN NouveauLivre

INSERT INTO Livre VALUES (@ISBN, @Titre, @Edition, @Categorie, @Langue, @Annee, @Pages, @LectureSeule, @CoutReparation, @Rayon)

SET @err += @@ERROR

WHILE @NbrExemplaires > 0 BEGIN

EXEC sp\_ajouter\_exemplaire @ISBN SET @err += @@ERROR

SET @NbrExemplaires -= 1

END

EXEC ajouter\_ecrivain @TableEcrivains SET @err += @@ERROR

GO;

IF @err = 0 COMMIT TRAN NouveauLivre ELSE ROLLBACK TRAN NouveauLivre

CREATE PROC sp\_modifier\_livre(@ISBN NVARCHAR(20),

@Titre NVARCHAR(100),

@Edition NVARCHAR(50), @Categorie NVARCHAR(50), @Langue NVARCHAR(50), @Annee SMALLINT,

@Pages INT, @LectureSeule BIT, @NbrExemplaires INT, @Rayon INT, @CoutReparation MONEY,

@TableEcrivains TypeTableEcrivain

READONLY) AS

DECLARE @err INT = 0

BEGIN TRAN Modification UPDATE Livre

SET Titre = @Titre,

Edition = @Edition, Categorie = @Categorie, Langue = @Langue,

Annee = @Annee, Pages = @Pages,

LectureSeule = @LectureSeule, Rayon = @Rayon,

CoutReparation = @CoutReparation

WHERE (ISBN = @ISBN)

SET @err += @@ERROR

WHILE @NbrExemplaires > 0 BEGIN

END

EXEC sp\_ajouter\_exemplaire @ISBN SET @err += @@ERROR

SET @NbrExemplaires -= 1

IF (SELECT COUNT(\*) FROM @TableEcrivains) > 0 BEGIN

DELETE FROM Ecrivain WHERE (ISBN = @ISBN)

SET @err += @@ERROR

END

EXEC ajouter\_ecrivain @TableEcrivains SET @err += @@ERROR

GO;

IF @err = 0 COMMIT TRAN Modification ELSE ROLLBACK TRAN Modification

CREATE PROC sp\_supprimer\_livre(@ISBN NVARCHAR(20)) AS DELETE FROM Livre WHERE (ISBN = @ISBN)

GO;

CREATE PROC sp\_ajouter\_auteur(@NomAuteur NVARCHAR(20)) AS INSERT INTO Auteur(NomAuteur) VALUES (@NomAuteur)

GO;

CREATE PROC sp\_supprimer\_auteur(@NumAuteur INT) AS DELETE FROM Auteur WHERE (NumAuteur = @NumAuteur)

GO;

CREATE PROC sp\_modifier\_auteur(@NumAuteur INT, @NomAuteur NVARCHAR(20)) AS UPDATE Auteur

SET NomAuteur = @NomAuteur WHERE (NumAuteur = @NumAuteur)

GO;

CREATE PROC sp\_ajouter\_adherent(@NomAdherent NVARCHAR(20),

@PrenomAdherent NVARCHAR(20), @DateNaissance DATE,

@Email NVARCHAR(255), @Adresse NVARCHAR(255)) AS

INSERT INTO Adherent(NomAdherent, PrenomAdherent, DateNaissance, Email, Adresse) VALUES (@NomAdherent, @PrenomAdherent, @DateNaissance, @Email, @Adresse)

GO;

CREATE PROC sp\_modifier\_adherent(@NumAdherent INT,

@NomAdherent NVARCHAR(20), @PrenomAdherent NVARCHAR(20), @DateNaissance DATE,

@Email NVARCHAR(255), @Adresse NVARCHAR(255)) AS

GO;

UPDATE Adherent

SET NomAdherent = @NomAdherent, PrenomAdherent = @PrenomAdherent, DateNaissance = @DateNaissance, Email = @Email,

Adresse = @Adresse

WHERE (NumAdherent = @NumAdherent)

DROP proc modifier\_adherent

CREATE PROC sp\_supprimer\_adherent(@NumAdherent INT) AS DELETE FROM Adherent WHERE (NumAdherent = @NumAdherent)

GO;

CREATE FUNCTION udf\_auth(@User NVARCHAR(50), @Pwd NVARCHAR(50)) RETURNS BIT AS

BEGIN

IF @User = 'BiblioAdmin' AND @Pwd = 'admin123'

END GO;

RETURN (1)

RETURN (0)

CREATE PROC ajouter\_ecrivain(@TableEcrivains TypeTableEcrivain READONLY) AS INSERT INTO Ecrivain SELECT \* FROM @TableEcrivains

GO;

CREATE PROC sp\_ajouter\_pret(@NumAdherent INT, @ISBN NVARCHAR(20)) AS DECLARE @LectureSeule BIT = (SELECT LectureSeule FROM Livre WHERE

(ISBN = @ISBN))

DECLARE @DateFinSuspension DATE = (SELECT MAX(DateFin) FROM Suspension WHERE (NumAdherent = @NumAdherent))

DECLARE @NumExemplaire INT = (SELECT MAX(NumExemplaire) FROM Exemplaire WHERE (Etat = 'disponible' AND ISBN = @ISBN))

IF NOT @DateFinSuspension IS NULL AND @DateFinSuspension > GETDATE() THROW 50001, 'Cet adherent est suspendu', 1

ELSE IF @LectureSeule = 1

THROW 50002, 'Cet article est pour la lecture ou la reference seulement', 1

ELSE IF @NumExemplaire IS NULL

THROW 50003, 'Il n''y a pas d''exemplaires disponibles', 1

ELSE

BEGIN

INSERT INTO Pret(NumAdherent, NumExemplaire) VALUES

(@NumAdherent, @NumExemplaire)

EXEC sp\_modifier\_etat\_exemplaire @NumExemplaire, 'Emprunté'

END

GO;

CREATE PROC ajouter\_penalite(@NumAdherent INT, @DatePanlite DATE, @Montant MONEY, @RaisonPenalite NVARCHAR(200)) AS

INSERT INTO Penalite(NumAdherent, DatePenalite, Montant, RaisonPenalite) VALUES (@NumAdherent, @DatePanlite, @Montant, @RaisonPenalite)

GO;

CREATE PROC sp\_conclure\_pret(@NumAdherent INT, @ISBN NVARCHAR(20), @Conclusion NVARCHAR(20)) AS

IF @NumAdherent NOT IN (SELECT NumAdherent FROM Adherent) THROW 50004, 'Numero d''adherent n''existe pas', 1

ELSE IF @ISBN NOT IN (SELECT ISBN FROM Livre)

THROW 50005, 'Ce livre n''existe pas', 1

DECLARE @NumExemplaire INT = (SELECT TOP 1 E.NumExemplaire FROM Exemplaire E INNER JOIN Pret P ON E.NumExemplaire = P.NumExemplaire WHERE (ISBN = @ISBN AND Etat = 'Emprunté' AND P.NumAdherent = @NumAdherent))

DECLARE @DatePret DATE = (SELECT TOP 1 DatePret FROM Pret WHERE (NumExemplaire = @NumExemplaire AND NumAdherent = @NumAdherent AND Conclusion IS NULL))

PRINT CAST(@NumExemplaire AS CHAR(10))

UPDATE Pret

SET DateConclusion = GETDATE(), Conclusion = @Conclusion

WHERE (NumAdherent = @NumAdherent AND NumExemplaire = @NumExemplaire AND DatePret = @DatePret)

GO;

CREATE TRIGGER tr\_appliquer\_penalite ON Pret AFTER UPDATE AS IF UPDATE(dateConclusion) AND UPDATE(Conclusion)

BEGIN

INSERTED) INSERTED)

DECLARE @err INT = 0

DECLARE @DatePret DATE = (SELECT DatePret FROM INSERTED)

DECLARE @DatePenalite DATE = GETDATE()

DECLARE @NumAdherent INT = (SELECT NumAdherent FROM INSERTED) DECLARE @Conclusion NVARCHAR(20) = (SELECT Conclusion FROM

DECLARE @NumExemplaire INT = (SELECT NumExemplaire FROM

DECLARE @retard INT = DATEDIFF(day, @DatePret, GETDATE()) - 30 DECLARE @penalite MONEY, @montant MONEY = 0, @raisonPenalite

NVARCHAR(20)

DECLARE @dateNaiss DATE = (SELECT DateNaissance FROM Adherent WHERE (NumAdherent = @NumAdherent))

DECLARE @coutReparation MONEY = (SELECT CoutReparation FROM Livre AS L INNER JOIN Exemplaire AS E ON L.ISBN = E.ISBN AND

E.NumExemplaire = @NumExemplaire)

IF DATEDIFF(year, @dateNaiss, GETDATE()) >= 18 SET @penalite = 1.5

ELSE

SET @penalite = 0.5

BEGIN TRAN ConclusionPret UPDATE Pret

SET DateConclusion = GETDATE(), Conclusion = @Conclusion

WHERE (NumAdherent = @NumAdherent AND NumExemplaire = @NumExemplaire AND DatePret = @DatePret)

SET @err += @@ERROR

IF @Conclusion IN ('Perdu', 'Delai depassé') BEGIN

'Perdu'

EXEC sp\_modifier\_etat\_exemplaire @NumExemplaire, SET @err += @@ERROR

EXEC ajouter\_penalite @NumAdherent, @DatePenalite, @coutReparation, 'Perte de l''article emprunté'

SET @err += @@ERROR

END

ELSE IF @Conclusion = 'Endommagé' BEGIN

'Endommagé'

EXEC sp\_modifier\_etat\_exemplaire @NumExemplaire, SET @err += @@ERROR

EXEC ajouter\_penalite @NumAdherent, @DatePenalite, @coutReparation, 'Article emprunté endommagé'

SET @err += @@ERROR

END

'Disponible'

ELSE BEGIN

EXEC sp\_modifier\_etat\_exemplaire @NumExemplaire,

SET @err += @@ERROR

END

IF @retard > 0 BEGIN

jour(s) de retard'

SET @montant = @retard \* @penalite

SET @raisonPenalite = CAST(@retard AS NVARCHAR(3)) + '

EXEC ajouter\_penalite @NumAdherent, @DatePenalite,

@montant, @raisonPenalite

SET @err += @@ERROR

END

GO;

END

IF @err = 0 COMMIT TRAN ConclusionPret ELSE ROLLBACK TRAN ConclusionPret

CREATE PROC sp\_ajouter\_suspension(@NumAdherent INT, @DateDebut DATE, @DateFin DATE, @RaisonSuspension NVARCHAR(200)) AS

INSERT INTO Suspension(NumAdherent, DateDebut, DateFin,

RaisonSuspension) VALUES

(@NumAdherent, @DateDebut, @DateFin, @RaisonSuspension)

GO;

CREATE TRIGGER tr\_appliquer\_suspension ON Pret AFTER UPDATE AS IF UPDATE(dateConclusion) AND UPDATE(Conclusion)

BEGIN

DECLARE @NumAdherent INT = (SELECT NumAdherent FROM INSERTED)

DECLARE @dateAvant3mois DATE = DATEADD(MONTH, -3, GETDATE())

DECLARE @totalPenalites MONEY = (SELECT SUM(Montant) FROM Penalite WHERE (NumAdherent = @NumAdherent AND DatePenalite BETWEEN @dateAvant3mois AND GETDATE()))

DECLARE @DateDebut DATE = GETDATE()

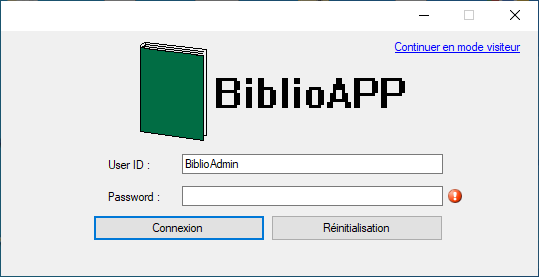
DECLARE @DateFin DATE = DATEADD(DAY, 14, @DateDebut)

IF @totalPenalites > 200

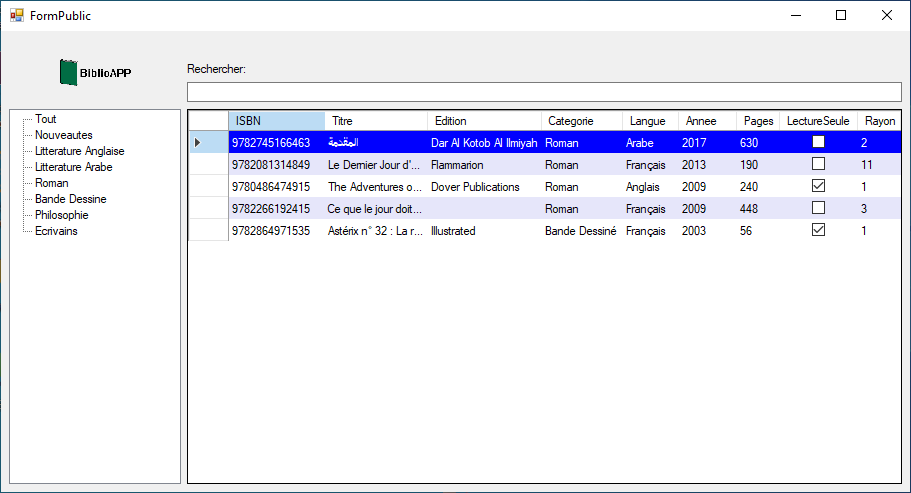
EXEC sp\_ajouter\_suspension @NumAdherent, @DateDebut, @DateFin, 'Accumulation de penalités'

END

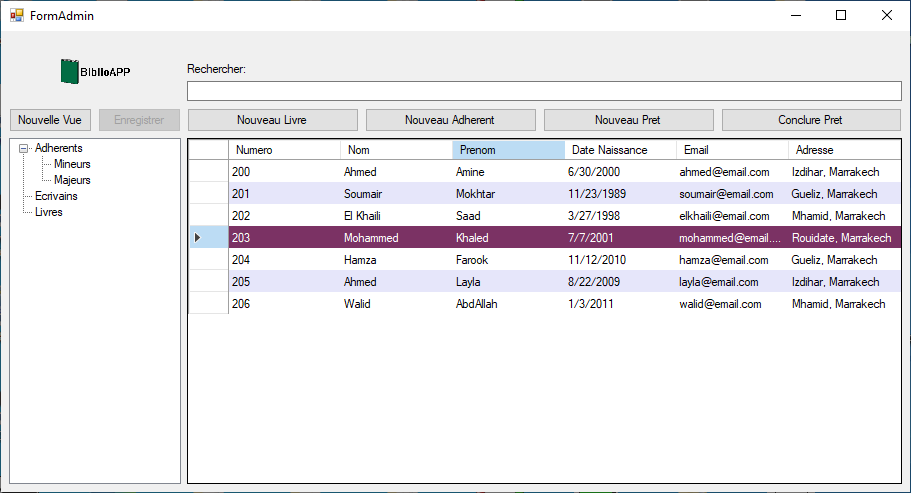
GO;

Captures d’écran

Page d’authentification



Page publique



Page Admin