Programme de création de facture sous forme SEPA

Table of Contents

Objectif du projet :

Etant en étude informatique et intéressé par le monde de la programmation, ma volonté était d'améliorer mes compétences en codage afin de prendre de l'avance et élargir mes connaissances. C'est pour cela que je me suis proposé à mon réseau pour coder d'éventuels programme pouvant me pousser à pratiquer l'auto-formation et consolider mes acquis.

Objectif du programme :

Ce projet est une demande d'une entreprise programmant une solution pour un sous-traitant. Dans le cadre du développement d'une facturation automatique, il m'a été demandé de créer un programme qui, à partir d'information sur une base de données, formule une facture sous forme SEPA.

Développement :

Qu'est-ce que le format SEPA :

Ce projet étant une option non confirmée de la part du client, le cahier des charges était très simple : créer un programme créant une facture dans un chier de type « .xml » regroupant toutes les informations de virements nécessaire.

Tout d'abord il a donc fallu que je me renseigne sur les factures de type SEPA. Ces factures sont sous formes de balises et donnent toutes les informations nécessaires à un virement bancaire : BIC, IBAN, nom, montant, etc. Voici quelques liens intéressent sur ce format et son mode de fonctionnement :

https://fr.wikipedia.org/wiki/Espace_unique_de_paiement_en_euros

https://docs.oracle.com/cd/E39124_01/doc.91/e60210/fields_sepa_pay_file_appx.htm#EOAEL00519

Récupérations données :

Ce programme a pour but, comme dit plus haut, de récupérer des informations sur la facturation de services et de créer à partir de celles-ci une facture dans un format qui permet l'initialisation automatique de virements.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[SousTraitant](
    [idold] [int] NOT NULL,
    [Client] [varchar](255) NULL,
    [CodeSousTraitant] [int] IDENTITY(1,1) NOT NULL,
    [RaisonSociale] [varchar](255) NULL,
    [Capital] [varchar](255) NULL,
    [NomRepresentant] [varchar](255) NULL,
    [PrenomRepresentant] [varchar](255) NULL,
    [Poste] [varchar](255) NULL,
    [Responsable] [varchar](255) NULL,
    [Adresse] [varchar](255) NULL,
    [ComplAdresse] [varchar](255) NULL,
    [CodePostal] [varchar](255) NULL,
    [Ville] [varchar](255) NULL,
    [VilleImmatric] [varchar](255) NULL,
```

```
// Creer une facture
Virement facture = creerFacture();
```

Les données sont donc récupérées sur une base de données, ici une monotable avec toutes les informations sur une chaque facture et compléter cela avec des informations prédéfinies et qui ne changent pas tel que le nom de l'entreprise qui effectue la facture ou encore son IBAN. Pour des raisons de confidentialités et de simplicité, j'ai décidé de faire une saisie automatique des données afin de me focaliser sur la création du fichier XML.

```
Virement facture = new Virement();
Random aleatoire = new Random();
// Information Société Créditée et facture
facture.IdIntern = "NumSoc/NumCB/74/01";
facture.Date = DateTime.Now.ToString("yyyy-MM-ddTHH:mm:ss");
```

D'autres paramètres sont remplis automatiquement comme la date puis une boucle remplit le nombre de factures demandé avec les mêmes informations mais qui seront différentes lors de la livraison.

```
for (int i = 0; i <= facture.NbVirmTotal; i = i + facture.Debitee[cpt-1].NbFac)
{
    Debiteur debitee = new Debiteur();
    //Information Société Débitée et motif
    debitee.Bic = "CRLYFRPP";
    debitee.Nom = "Nom Societe Debiteur " + Convert.ToString(cpt + 1);
```

Classe virement :

La classe virement contient tous les attributs nécessaires à la création d'une facture en format XML. Elle contient une méthode ToXML() qui créer le fichier de facture demandé en appelant d'une part les variables contenue dans ces attributs : identifiant interne du fichier de facturation, le numéro de facture de l'entreprise éditrice du fichier; et puis fait appelle aux méthode ToXML() des classes débiteur et créditeur qui sont instanciées en temps que attributs de la classe virement.

```
7 références
class Virement
{
    private string idIntern;
    private Crediteur crediteur;
    private List<Debiteur> debiteurList;
    private string date;
```

Du fait d'un nombre important de test et de lancement de ce programme, j'ai dû trouver un nom temporaire pour les fichiers créer afin de les reconnaitre et qu'ils se classent correctement dans un fichier. La solution est donc simple : nommé le fichier avec l'heure à laquelle il est créé. Cela permettait de les voir afficher dans un même dossier du plus ancien au plus récent.

```
facture.Nom = DateTime.Now.ToString("yyyy-MM-dd_HH-mm-ss");
StreamWriter sw = new StreamWriter("FichierXML/" + "Virement_" + facture.Nom + ".xml", false, Encoding.UTF8);
```

2020-02-10_14-45-38.xml	10/02/2020 14:45	Document XML	6 Ko
2020-02-10_14-47-30.xml	10/02/2020 14:47	Document XML	6 Ko
2020-02-10_14-55-17.xml	10/02/2020 16:59	Document XML	6 Ko
2020-02-10_17-03-05.xml	10/02/2020 17:03	Document XML	6 Ko
2020-02-10 17-06-20.xml	10/02/2020 17:06	Document XML	6 Ko

Classe Créditeur et Débiteur :

Comme vu dans le paragraphe précédent, les classes créditeur et débiteur contiennent les informations sur les entreprises nécessaires pour la facture. Ces classes héritent de la classe société qui définit les attributs propres aux deux sociétés et ensuite ajoutent les données nécessaires à la rédaction de la facture.

```
3 références
class Crediteur : Societe
{
    4 références
    public override string[] ToXML()
    {
        string[] textCrediteur = new string[2];
```

```
2 références
abstract class Societe
{

    private string idFisc;
    private string nom;
    private string iBAN;
```

Les informations sur les deux sociétés ne sont pas utilisées à la suite durant la création du fichier, de ce fait il a fallu séparer les informations en différents parties qui sont donc stockées dans un tableau de variables type string. C'est pour cela que les méthodes ToXML() de ces deux classes renvoient des tableaux de string qui sont utilisés dans la méthode de la classe virement.

La variable « debiteurList » est une liste de débiteurs à mettre dans la facture. Toutes « cases » de cette liste sont donc mise dans la facture tour à tour.

```
string[] textDebiteur = debiteurList[cpt].ToXML();
```

Vérification :

La facturation en format SEPA est une norme très stricte et nécessite donc d'effectuer une vérification à partir de modèles. Du fait de l'importance de cette étape, j'ai donc pris une grande partie sur des sources sur comme la vérification de fichier sur la documentation Microsoft, ou encore un site sur lequel est donné le schéma des fichiers de type xml.

```
//verification :
XmlReaderSettings booksSettings = new XmlReaderSettings();
booksSettings.Schemas.Add("urn:iso:std:iso:20022:tech:xsd:pain.001.001.03", "pain.001.001.03.xsd"); // Possible aussi 2pain.001.001.03
booksSettings.ValidationType = ValidationType.Schema;
booksSettings.ValidationEventHandler += new ValidationEventHandler(booksSettingsValidationEventHandler);

XmlReader books = XmlReader.Create("FichierXML/" + "Virement_" + facture.Nom + ".xml", booksSettings);
while (books.Read()) { }
```

https://docs.microsoft.com/fr-fr/dotnet/api/system.xml.schema.xmlschemaset.validationeventhandler?view=netframework-4.8

Exemple d'exécution :

```
  Espace de facturation


Le fichier: Virement_2020-04-15_16-36-24.xml a été créé
```

Des messages d'erreurs peuvent s'afficher si le programme ne fonctionne pas normalement, comme par exemple si le fichier créé n'est pas conforme et ne passe pas la vérification.

Conclusion :

L'initiative d'effectuer ce programme était un bon choix car il remplit tout à fait les objectifs que je me suis fixés en tant que jeune développeur afin de m'améliorer dans ce domaine. Ce projet contenait beaucoup de nouvelles notions pour moi ce qui a été bénéfiques quand bien même son but final fut changé. De nouvelles bases ont été aguerris comme par exemple la création de fichier en C# ou encore connaitre ce que sont les fichiers XML formés de balises et quels sont leurs utilités.

Contact :

Si vous souhaitez tester ou voir plus de ce logiciel vous pouvez me contacter à l'aide de mon adresse professionnel : pro@gavazziadrien.fr.