1) Générer un fichier XML

Pour qu'un document soit conforme au format factur-x (conforme aussi à la norme EN16931), il faut que le fichier XML en pièce jointe soit conforme au CII (Cross Industry Invoice)

Le CII est une norme internationale créer par l'UNECE (Commission économique pour l'Europe des Nations unies) pour les factures, rendant la facturation plus facile entre différentes entreprises. Il simplifie les processus, réduit les erreurs et accélère les paiements. En normalisant les données, il améliore la visibilité financière et aide les entreprises à être plus efficaces, économiser des ressources et respecter les règlements.

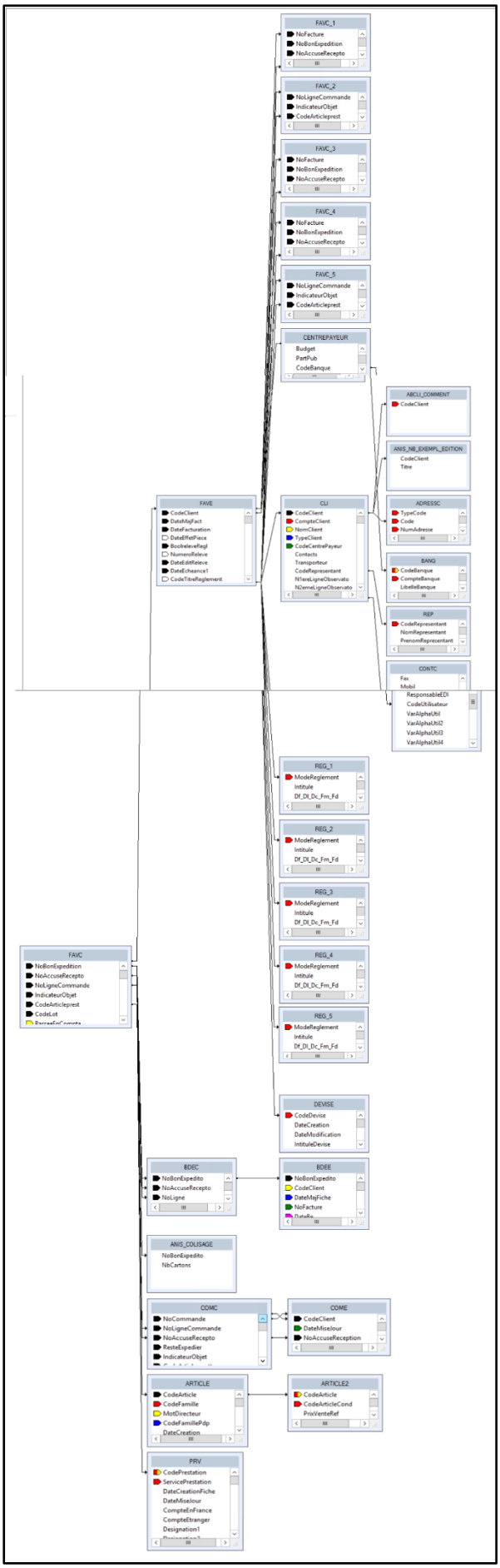
https://unece.org/trade/uncefact/e-invoice

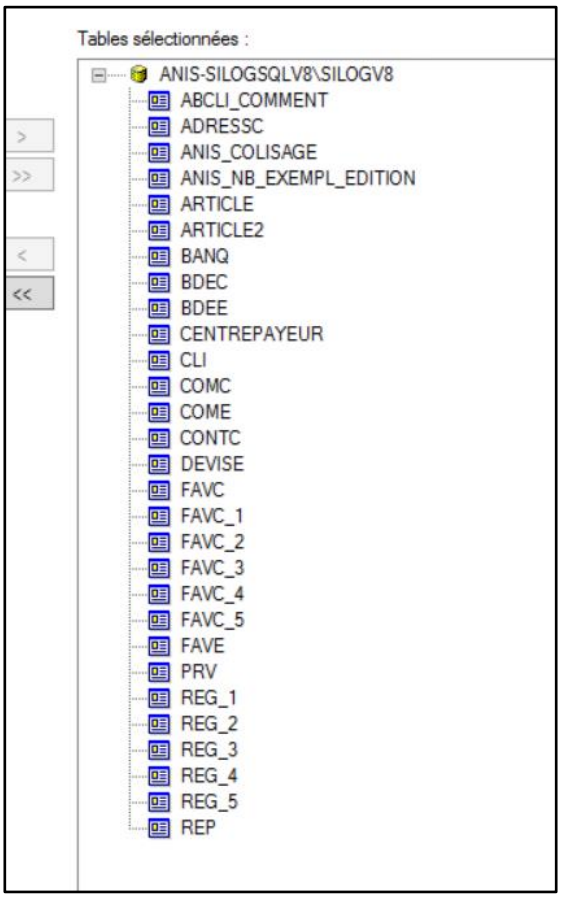
Prérequis :

- Microsoft SQL server management studio
- Installé les Library python suivant :

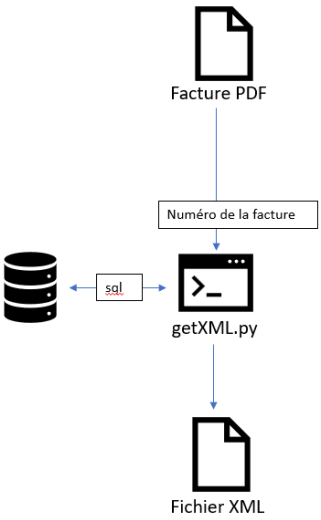
```
pip install pyodbc
pip install PyPDF2
pip install logging
pip install dateTime
```

On a une base de données contenant les données des factures :


FAVC_1
NoFacture
NoBonExpedition
NoAccuseRecepto
FAVC_2
NoLigneCommande
IndicateurObjet
CodeArticleprest
FAVC_3
NoFacture
NoBonExpedition
NoAccuseRecepto
FAVC_4
NoFacture
NoBonExpedition
NoAccuseRecepto
FAVC_5
NoLigneCommande
IndicateurObjet
CodeArticleprest
CENTREPAYEUR
Budget
PartPub
CodeBanque
ABCLI_COMMENT
CodeClient
ANIS_NB_EXEMPL_EDITION
CodeClient
Titre
FAVE
CodeClient
DateMajFact
DateFacturation
DateEffetPiece
BoolreleveRegl
NumeroReleve
DateEditReleve
DateEcheance1
CodeTitreReglement
CLI
CodeClient
CompteClient
NomClient
TypeClient
CodeCentrePayeur
Contacts
Transporteur
CodeRepresentant
NTiereLigneObservato
NDerneLigneObservato
ADRESSC
TypeCode
Code
NumAdresse
BANQ
CodeBanque
CompteBanque
LibelleBanque
REP
CodeRepresentant
NomRepresentant
PrenomRepresentant
CONTC
Fax
Mobil
ResponsableEDI
CodeUtilisateur
VarAlphaUtil
VarAlphaUtil2
VarAlphaUtil3
VarAlphaUtil4
REG_1
ModeReglement
Intitule
Df_Dl_Dc_Fm_Fd
REG_2
ModeReglement
Intitule
Df_Dl_Dc_Fm_Fd
REG_3
ModeReglement
Intitule
Df_Dl_Dc_Fm_Fd
REG_4
ModeReglement
Intitule
Df_Dl_Dc_Fm_Fd
REG_5
ModeReglement
Intitule
Df_Dl_Dc_Fm_Fd
FAVC
NoBonExpedition
NoAccuseRecepto
NoLigneCommande
IndicateurObjet
CodeArticleprest
CodeLot
DEVISE
CodeDevise
DateCreation
DateModification
IntituleDevise
BDEC
NoBonExpedito
NoAccuseRecepto
NoLigne
BDEE
NoBonExpedito
CodeClient
DateMajFiche
NoFacture
ANIS_COLISAGE
NoBonExpedito
NbCartons
COMC
NoCommande
NoLigneCommande
NoAccuseRecepto
ResteExpedier
IndicateurObjet
COME
CodeClient
DateMiseJour
NoAccuseReception
ARTICLE
CodeArticle
CodeFamille
MotDirecteur
CodeFamillePdp
DateCreation
ARTICLE2
CodeArticle
CodeArticleCond
PrixVenteRef
PRV
CodePrestation
ServicePrestation
DateCreationFiche
DateMiseJour
CompteEnFrance
CompteEtranger
Designation1


Tables sélectionnées :
ANIS-SILOGSQLV8\SILOGV8
ABCLI_COMMENT
ADRESSC
ANIS_COLISAGE
ANIS_NB_EXEMPL_EDITION
ARTICLE
ARTICLE2
BANQ
BDEC
BDEE
CENTREPAYEUR
CLI
COMC
COME
CONTC
DEVISE
FAVC
FAVC_1
FAVC_2
FAVC_3
FAVC_4
FAVC_5
FAVE
PRV
REG_1
REG_2
REG_3
REG_4
REG_5
REP

Tout d'abord, j'ai commencé par examiner le fichier XML pour répertorier les données importantes, ensuite je cherche ces données dans la base de données, puis je crée un programme qui place les données dans un modèle de fichier XML conforme au format Factur-X, et enfin le programme génère le fichier XML.



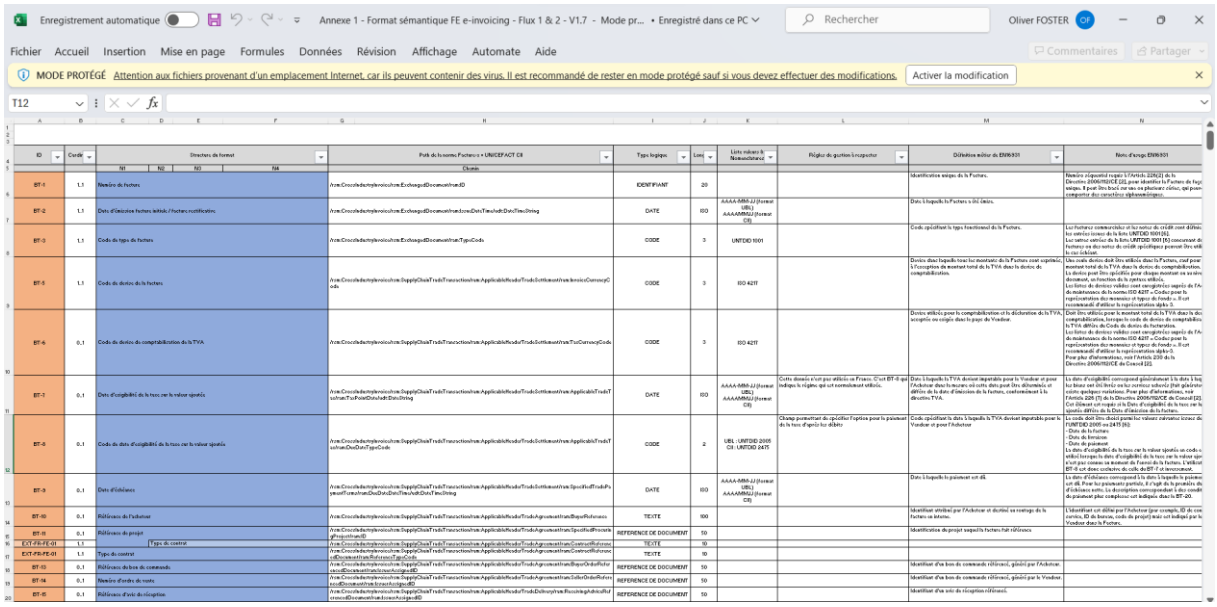
On peut trouver des exemples de fichiers sous format factur-x sur : https://fnfe-mpe.org/factur-x/factur-x-et-zugferd-2-2/

Documents indiquant la signification des balises :

https://www.impots.gouv.fr/specifications-externes-b2b

Version 2.3 du 31/07/2023 :

Télécharger les documents : Dossier de spécifications externes de la facturation électronique, annexes et swaggers (.zip)

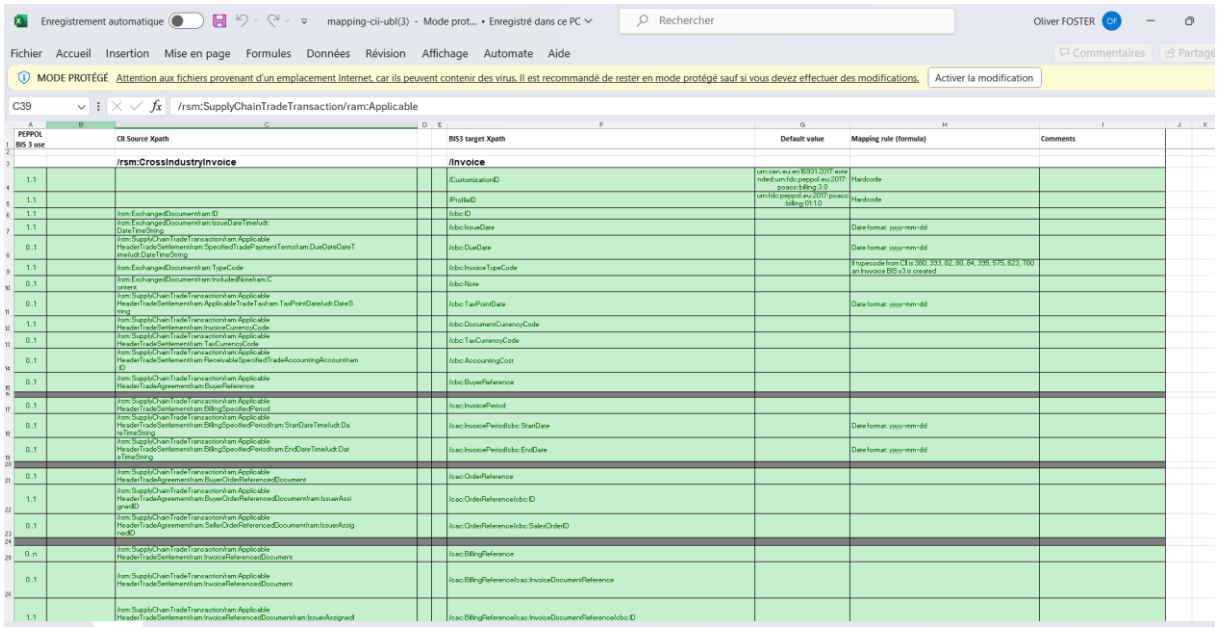


PEPPOL.eu
https://docs.peppol.eu › files › mapping-cii-ubl XLS

rsm:CrossIndustryInvoice /Invoice

1 janv. 1997 — **Mapping** rule (formula), Comments. 2. 3, /rsm:CrossIndustryInvoice, /Invoice ...
If not found in **CII** this is mapped from the line tax. 284, 0..1 ...

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwigsrXMwqqEAxXsRaQEHd-_DO8QFnoECB0QAQ&url=https%3A%2F%2Fdocs.peppol.eu%2Fpoacc%2Fbilling%2F3.0%2Ffiles%2Fmapping-cii-ubl.xls&usg=AOvVaw1goOHboba12_6avsSXJg4w&opi=89978449



Recherches des valeurs dans les tables :

	Nom balise XML	Signification	Table.ligne/Valeur	
	ID	numéro facture	FAVE.NoFacture	
	TypeCode	type de doc commercial (380)		380
	DateTimeString	date de facture	FACE.DateFacturation	
	Content	?	" "	
Chaque objet dans la fa	GlobalID	référence du produit	ARTICLE.CodeEAN	
	SellerAssignedID	référence du produit du vendeur	pas obligatoire	
	Name	nom du produit	ARTICLE.MotDirecteur	
	ChargeAmount	prix unitaire	FAVC.PrixUnitDevises	
si remise	ChargeIndicator	indicateur de charge (false)?	false	
	ActualAmount	remise	FAVC.PrixUnitDevises - FAVC.PrixUnitNet	
	ChargeAmount	prix unitaire - remise	FAVC.PrixUnitNet	
	BilledQuantity	quantité produit	FAVC.QuantiteFacturee	
	TypeCode	type de taxe (tva)	VAT	
	Category	catégorie de taxe (S)	S	
	RateApplicablePercent	% taxé	FAVC.TauxTVA	
	LineTotalAmount	prix HT	FAVC.MontHtRemisescomp	
Info Vendeur	Name	nom vendeur	Anic De Flavigny	
	ID	siret vendeur	392 006 516 00018	
	PersonName	prenom + nom contact commercial vendeur		
	CompleteNumber	tel contact commercial vendeur		
	URIID	email contact commercial vendeur		
	PostcodeCode	code postal vendeur		21150211150
	LineOne	adresse vendeur	4 Rue de l'Abbaye	
	CityName	ville vendeur	Flavigny-sur-Ozerain	
	CountryID	pays vendeur	FR	
	ID	num TVA vendeur	FR36392006516	
Info Client	Name	nom client	FAVE.NomClient	
	ID	siret client	CLI.Siret	
	PersonName	prenom + nom contact commercial clie	CONTC.Prenom + CONTC.nom	
	CompleteNumber	tel contact commercial client	CONTC.Telephone	
	URIID	email contact commercial client	CONTC.Email	
	PostcodeCode	code postal client	FAVE.CodePostal	
	LineOne	adresse client	FAVE.AdressePartie1 + FAVE.AdressePartie2	
	CityName	ville client	FAVE.VilleFacturation	
	CountryID	pays client	COALESCE(CLI.CodePays, CLI.CodePays2)	
	ID	num TVA client	FAVE.NoIdentificatoCee	
	BuyerOrderReferenceDocume	numéro engagement?		
	ContractReferenceDocument	Référence du contrat		
	PostcodeCode	code postal livraison	BDEE.CodePostal	
	LineOne	adresse livraison	BDEE.AdressePartie1 + BDEE.AdressePartie2	
	CityName	ville livraison	BDEE.VilleLivraison	
	CountryID	pays livraison	BDEE.CodePaysLivraison	
	ApplicableHeaderTradeSettle	numéro de facture	FAVE.NoFacture	
	InvoiceCurrencyCode	devise	DEVISE.SigleDevise	
	TypeCode	type code		30
	Information	information		
	IBANID	iban	BANQ.VarAlphaUtil2	
	BICID	bic	BANQ.VarAlphaUtil	
Par % de taxe différents	CalculatedAmount	ht * tva /100	SELECT sum(FAVC.MontHtRemisescomp)*FAVC.TauxTVA/100 WHERE FAVC.CodeTva="	
	TypeCode	type de taxe (tva)	VAT	
	BasisAmount	prix ht	SELECT sum(FAVC.MontHtRemisescomp) WHERE FAVC.CodeTva="	
	CategoryCode	catégorie de taxe (S)	S	
	DueDateTypeCode	Option de paiement de TVA		5
	RateApplicablePercent	% taxé	FAVC.TauxTVA	
	DueDateTypeCode	date échéance	COALESCE(FAVE.DateEcheance1, FAVE.DateEcheance2,FAVE.DateEcheance3,FAVE.DateEcheance4,FAVE.DateEcheance5)	
	LineTotalAmount	total HT	SUM(FAVC.MontHtRemisescomp)	
	TaxBasicTotalAmount	montant HT	SUM(FAVC.MontHtRemisescomp)	
	TaxTotalAmount	devise + total TVA	DEVISE.SigleDevise, (FAVE.MontantTvaNo1+FAVE.MontantTvaNo2+ FAVE.MontantTvaNo3+ FAVE.MontantTvaNo4+FAVE.MontantTvaNo5+ FAV	
	GrandTotalAmount	total TTC	(SELECT SUM(FAVC.MontantTtc) FROM FAVC WHERE FAVC.NoFacture = FAVE.NoFacture) AS 'prixTTC'	
	TotalPrepaidAmount	montant déjà payer	SUM(FAVE.AcompteEnDevises)	
	DuePayableAmount	montant à payer	SUM(FAVC.MontantTtc) - SUM(FAVE.AcompteEnDevises)	

1

2

Les valeurs dans les cases rouges sont les valeurs qui se répètent.

1. Par produits
2. Par taux de tva (0%, 5,5%, 20%)

```
<ram:AssociatedDocumentLineDocument>
  <ram:LineID>1</ram:LineID>
</ram:AssociatedDocumentLineDocument>
<ram:SpecifiedTradeProduct>
  <ram:GlobalID schemeID="0160">3518370400049</ram:GlobalID>
  <ram:SellerAssignedID>NOUG250</ram:SellerAssignedID>
  <ram:Name>Nougat de l'Abbaye 250g</ram:Name>
</ram:SpecifiedTradeProduct>
<ram:SpecifiedLineTradeAgreement>
  <ram:GrossPriceProductTradePrice>
    <ram:ChargeAmount>4.55</ram:ChargeAmount>
    <ram:AppliedTradeAllowanceCharge>
      <ram:ChargeIndicator>
        <udt:Indicator>false</udt:Indicator>
      </ram:ChargeIndicator>
      <ram:ActualAmount>0.45</ram:ActualAmount>
    </ram:AppliedTradeAllowanceCharge>
  </ram:GrossPriceProductTradePrice>
  <ram:NetPriceProductTradePrice>
    <ram:ChargeAmount>4.10</ram:ChargeAmount>
  </ram:NetPriceProductTradePrice>
</ram:SpecifiedLineTradeAgreement>
<ram:SpecifiedLineTradeDelivery>
  <ram:BilledQuantity unitCode="C62">20.000</ram:BilledQuantity>
</ram:SpecifiedLineTradeDelivery>
<ram:SpecifiedLineTradeSettlement>
  <ram:ApplicableTradeTax>
    <ram:TypeCode>VAT</ram:TypeCode>
    <ram:CategoryCode>S</ram:CategoryCode>
    <ram:RateApplicablePercent>20.00</ram:RateApplicablePercent>
  </ram:ApplicableTradeTax>
  <ram:SpecifiedTradeSettlementLineMonetarySummation>
    <ram:LineTotalAmount>81.90</ram:LineTotalAmount>
  </ram:SpecifiedTradeSettlementLineMonetarySummation>
</ram:SpecifiedLineTradeSettlement>
</ram:IncludedSupplyChainTradeLineItem>
<ram:IncludedSupplyChainTradeLineItem>
  <ram:AssociatedDocumentLineDocument>
    <ram:LineID>2</ram:LineID>
  </ram:AssociatedDocumentLineDocument>
  <ram:SpecifiedTradeProduct>
    <ram:GlobalID schemeID="0160">3518370200090</ram:GlobalID>
    <ram:SellerAssignedID>BRAIS300</ram:SellerAssignedID>
    <ram:Name>Biscuits aux raisins 300g</ram:Name>
  </ram:SpecifiedTradeProduct>
  <ram:SpecifiedLineTradeAgreement>
    <ram:GrossPriceProductTradePrice>
      <ram:ChargeAmount>3.20</ram:ChargeAmount>
    </ram:GrossPriceProductTradePrice>
    <ram:NetPriceProductTradePrice>
      <ram:ChargeAmount>3.20</ram:ChargeAmount>
    </ram:NetPriceProductTradePrice>
  </ram:SpecifiedLineTradeAgreement>
  <ram:SpecifiedLineTradeDelivery>
    <ram:BilledQuantity unitCode="C62">15.000</ram:BilledQuantity>
  </ram:SpecifiedLineTradeDelivery>
  <ram:SpecifiedLineTradeSettlement>
    <ram:ApplicableTradeTax>
      <ram:TypeCode>VAT</ram:TypeCode>
      <ram:CategoryCode>S</ram:CategoryCode>
      <ram:RateApplicablePercent>5.50</ram:RateApplicablePercent>
    </ram:ApplicableTradeTax>
    <ram:SpecifiedTradeSettlementLineMonetarySummation>
      <ram:LineTotalAmount>48.00</ram:LineTotalAmount>
    </ram:SpecifiedTradeSettlementLineMonetarySummation>
  </ram:SpecifiedLineTradeSettlement>
</ram:IncludedSupplyChainTradeLineItem>
<ram:IncludedSupplyChainTradeLineItem>
  <ram:AssociatedDocumentLineDocument>
    <ram:LineID>3</ram:LineID>
  </ram:AssociatedDocumentLineDocument>
```

```
<ram:ApplicableTradeTax>
  <ram:CalculatedAmount>16.38</ram:CalculatedAmount>
  <ram:TypeCode>VAT</ram:TypeCode>
  <ram:BasisAmount>81.90</ram:BasisAmount>
  <ram:CategoryCode>S</ram:CategoryCode>
  <ram:DueDateTypeCode>5</ram:DueDateTypeCode>
  <ram:RateApplicablePercent>20.00</ram:RateApplicablePercent>
</ram:ApplicableTradeTax>
<ram:ApplicableTradeTax>
  <ram:CalculatedAmount>29.87</ram:CalculatedAmount>
  <ram:TypeCode>VAT</ram:TypeCode>
  <ram:BasisAmount>543.00</ram:BasisAmount>
  <ram:CategoryCode>S</ram:CategoryCode>
  <ram:DueDateTypeCode>5</ram:DueDateTypeCode>
  <ram:RateApplicablePercent>5.50</ram:RateApplicablePercent>
</ram:ApplicableTradeTax>
```

La commande SQL :

```
SELECT
    FAVE.NoFacture AS 'ID',
    CONVERT(VARCHAR, FAVE.DateFacturation, 112) AS 'DateFact',
    ARTICLE.CodeEAN AS 'GlobalID',
    ARTICLE.MotDirecteur AS 'NameItem',
    FAVC.PrixUnitDevises AS 'ChargeAmount',
    (FAVC.PrixUnitDevises - FAVC.PrixUnitNet) AS 'ReducedValue',
    FAVC.PrixUnitNet AS 'ChargeAmount2',
    FAVC.QuantiteFacturee AS 'BilledQuantity',
    FAVC.TauxTVA AS 'RateApplicablePercent',
    FAVC.MonthtRemisescomp AS 'LineTotalAmount',
    FAVE.NomClient AS 'NameClient',
    CLI.Siret AS 'SIRET',
    (CONTC.Prenom + ' ' + CONTC.nom) AS 'PersonName',
    CONTC.Telephone AS 'CompleteNumber',
    CONTC.Email AS 'URIID',
    FAVE.CodePostal AS 'PostecodeCode',
    (FAVE.AdressePartie1 + ' ' + FAVE.AdressePartie2) AS 'LineOne',
    FAVE.VilleFacturation AS 'CityName',
    COALESCE(CLI.CodePays, CLI.CodePays2) AS 'CountryID',
    CLI.NoIdentificatoCee AS 'numTVA',
    BDEE.CodePostal AS 'PostecodeCodeLivr',
    (BDEE.AdressePartie1 + ' ' + BDEE.AdressePartie2) AS 'LineOneLivr',
    BDEE.VilleLivraison AS 'CityNameLivr',
    BDEE.CodePaysLivraison AS 'CountryIDLivr',
    DEVISE.SigleDevise AS 'InvoiceCurrencyCode',
    BANQ.VarAlphaUtil2 AS 'IBANID',
    BANQ.VarAlphaUtil AS 'BICID',
    SUM(FAVC.MonthtRemisescomp) AS 'BasisAmount',
    CONVERT(VARCHAR, COALESCE(FAVE.DateEcheance1, FAVE.DateEcheance2, FAVE.DateEcheance3,
FAVE.DateEcheance4, FAVE.DateEcheance5), 112) AS 'DueDateTypeCode',
    (SELECT SUM(FAVC.PrixUnitDevises * FAVC.QuantiteFacturee) FROM FAVC WHERE
FAVC.NoFacture='FA24/00206' GROUP BY FAVC.NoFacture) AS 'LineTotalAmount2',
    FAVE.MontantTtc AS 'GrandTotalAmount',
    FAVE.AcompteEnDevises AS 'TotalPrepaidAmount',
    FAVE.MontantTtc-FAVE.AcompteEnDevises AS 'DuePayableAmount',
        CLI.CodeClient
FROM
    FAVE
JOIN
    FAVC ON FAVE.NoFacture = FAVC.NoFacture
JOIN
    ARTICLE ON ARTICLE.CodeArticle = FAVC.CodeArticleprest
JOIN
    CLI ON FAVE.CodeClient = CLI.CodeClient
JOIN
    CONTC ON CLI.VarAlphaUtil = CONTC.CodeUtilisateur
JOIN
    BDEE ON BDEE.NoBonExpedito = FAVC.NoBonExpedition
JOIN
    DEVISE ON FAVE.CodeDevise = DEVISE.CodeDevise
JOIN
    BANQ ON BANQ.CodeBanque = FAVE.CodeBanque1
WHERE
    FAVE.NoFacture = ?
    AND FAVC.PrestTalon = 'N'
GROUP BY
    FAVE.NoFacture, FAVE.DateFacturation, ARTICLE.CodeEAN, ARTICLE.MotDirecteur, FAVC.PrixUnitDevises,
FAVC.PrixUnitNet, FAVC.QuantiteFacturee, FAVC.TauxTVA, FAVE.DateEcheance1, FAVE.DateEcheance2,
FAVE.DateEcheance3, FAVE.DateEcheance4, FAVE.DateEcheance5, FAVC.MonthtRemisescomp, FAVE.NomClient,
CLI.Siret, CONTC.Prenom, CONTC.nom, CONTC.Telephone, CONTC.Email, FAVE.CodePostal,
FAVE.AdressePartie1, FAVE.AdressePartie2, FAVE.VilleFacturation, CLI.CodePays, CLI.CodePays2,
CLI.NoIdentificatoCee, BDEE.CodePostal, BDEE.AdressePartie1, BDEE.AdressePartie2, BDEE.VilleLivraison,
BDEE.CodePaysLivraison, DEVISE.SigleDevise, BANQ.VarAlphaUtil2, BANQ.VarAlphaUtil, FAVE.MontantTtc,
FAVE.AcompteEnDevises, CLI.CodeClient
ORDER BY
    FAVE.NoFacture DESC
```

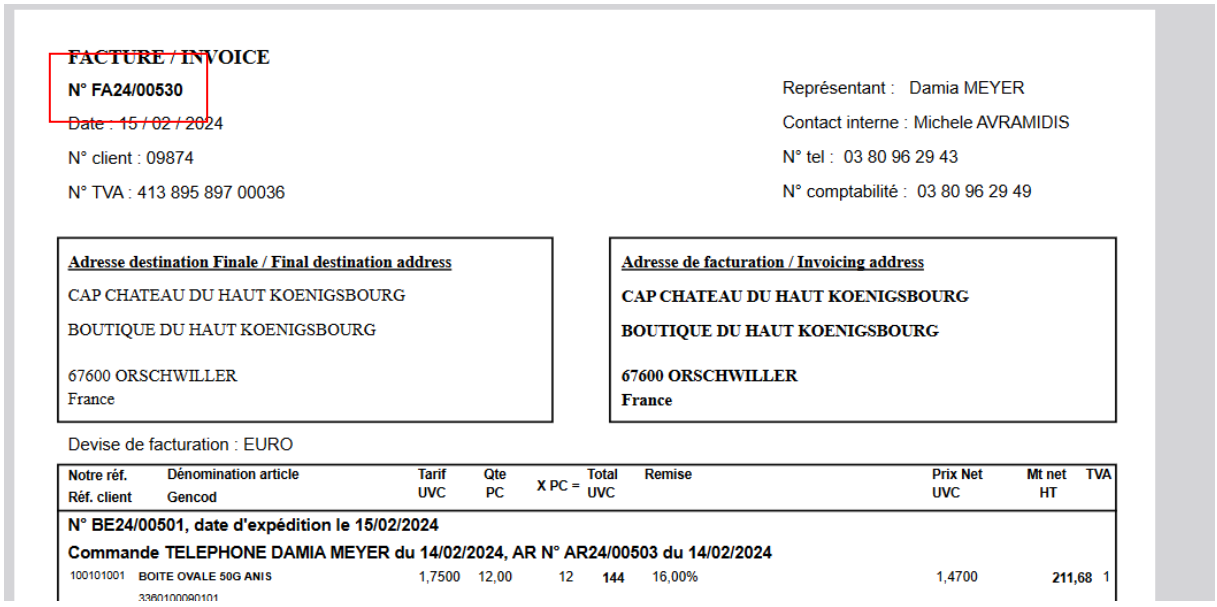
Voici un exemple :

```
SELECT
    FAVE.NoFacture AS 'ID',
    CONVERT(VARCHAR, FAVE.DateFacturation, 112) AS 'DateFact',
    ARTICLE.CodeEAN AS 'GlobalID',
    ARTICLE.MotDirecteur AS 'NameItem',
    FAVC.PrixUnitDevises AS 'ChargeAmount',
    (FAVC.PrixUnitDevises - FAVC.PrixUnitNet) AS 'ReducedValue',
    FAVC.PrixUnitNet AS 'ChargeAmount2',
    FAVC.QuantiteFacturee AS 'BilledQuantity',
    FAVC.TauxTVA AS 'RateApplicablePercent',
    FAVC.MonthtRemisescomp AS 'LineTotalAmount',
    FAVE.NomClient AS 'NameClient',
    CLI.Siret AS 'SIRET',
    (CONTC.Prenom + ' ' + CONTC.nom) AS 'PersonName',
    CONTC.Telephone AS 'CompleteNumber',
    CONTC.Email AS 'URIID',
    FAVE.CodePostal AS 'PostecodeCode',
    (FAVE.AdressePartie1 + ' ' + FAVE.AdressePartie2) AS 'LineOne',
    FAVE.VilleFacturation AS 'CityName',
    COALESCE(CLI.CodePays, CLI.CodePays2) AS 'CountryID',
    CLI.NoIdentificatoCee AS 'numTVA',
    BDEE.CodePostal AS 'PostecodeCodeLivr',
    (BDEE.AdressePartie1 + ' ' + BDEE.AdressePartie2) AS 'LineOneLivr',
    BDEE.VilleLivraison AS 'CityNameLivr',
    BDEE.CodePaysLivraison AS 'CountryIDLivr',
    DEVISE.SigleDevise AS 'InvoiceCurrencyCode',
    BANQ.VarAlphaUtil2 AS 'IBANID',
    BANQ.VarAlphaUtil AS 'BICID',
    SUM(FAVC.MonthtRemisescomp) AS 'BasisAmount',
    CONVERT(VARCHAR, COALESCE(FAVE.DateEcheance1, FAVE.DateEcheance2, FAVE.DateEcheance3, FAVE.DateEcheance4, FAVE.DateEcheance5), 112) AS 'DueDateTypeCode',
    (SELECT SUM(FAVC.PrixUnitDevises * FAVC.QuantiteFacturee) FROM FAVC WHERE FAVC.NoFacture='FA24/00206' GROUP BY FAVC.NoFacture) AS 'LineTotalAmount',
    FAVE.MontantTtc AS 'GrandTotalAmount',
    FAVE.AcompteEnDevises AS 'TotalPrepaidAmount',
    FAVE.MontantTtc-FAVE.AcompteEnDevises AS 'DuePayableAmount'
```

Results | Messages

ID	DateFact	GlobalID	NameItem	ChargeAmount	ReducedValue	ChargeAmount2	BilledQuantity	RateApplicablePercent	LineTotalAmount	NameClient	SIRET	PersonName	CompleteNumber
FA24/00205	20240118	3360100090101	BOITE OVALE 50G ANIS	1.75000000	0.00000000	1.75000000	36.00000000	20.00000000	63.00000000	GAMM VERT SA - PARIS	33789128700019	Michele AVRAMIDIS	03 80 96 29 43
FA24/00205	20240118	3360100150102	BOITE OVALE 50G FLEUR ORANGER	1.75000000	0.00000000	1.75000000	24.00000000	20.00000000	42.00000000	GAMM VERT SA - PARIS	33789128700019	Michele AVRAMIDIS	03 80 96 29 43
FA24/00205	20240118	3360100180109	SAC 500G ANIS	9.15000000	0.00000000	9.15000000	4.00000000	20.00000000	36.60000000	GAMM VERT SA - PARIS	33789128700019	Michele AVRAMIDIS	03 80 96 29 43
FA24/00205	20240118	3360100520103	BOITE OVALE 50G CITRON	1.75000000	0.00000000	1.75000000	24.00000000	20.00000000	42.00000000	GAMM VERT SA - PARIS	33789128700019	Michele AVRAMIDIS	03 80 96 29 43
FA24/00205	20240118	3360102150100	BOITE OVALE 50G CASSIS	1.75000000	0.00000000	1.75000000	24.00000000	20.00000000	42.00000000	GAMM VERT SA - PARIS	33789128700019	Michele AVRAMIDIS	03 80 96 29 43
FA24/00205	20240118	3360102280012	BOITE OVALE 50G EUCALYPTUS BIOLOGIQUE	2.17000000	0.00000000	2.17000000	12.00000000	20.00000000	26.04000000	GAMM VERT SA - PARIS	33789128700019	Michele AVRAMIDIS	03 80 96 29 43

Comme on a maintenant les données, il faut récupérer le numéro de la facture. On parcourt le fichier pour retrouver du texte que ressemble à ce format : "FAXX/XXXXX"



FACTURE / INVOICE

N° FA24/00530

Date : 15 / 02 / 2024

N° client : 09874

N° TVA : 413 895 897 00036

Représentant : Damia MEYER

Contact interne : Michele AVRAMIDIS

N° tel : 03 80 96 29 43

N° comptabilité : 03 80 96 29 49

Adresse destination Finale / Final destination address

CAP CHATEAU DU HAUT KOENIGSBOURG

BOUTIQUE DU HAUT KOENIGSBOURG

67600 ORSCHWILLER
France

Adresse de facturation / Invoicing address

CAP CHATEAU DU HAUT KOENIGSBOURG

BOUTIQUE DU HAUT KOENIGSBOURG

67600 ORSCHWILLER
France

Devise de facturation : EURO

Notre réf. / Réf. client	Dénomination article / Gencod	Tarif UVC	Qte PC	X PC =	Total UVC	Remise	Prix Net UVC	Mt net HT	TVA
N° BE24/00501, date d'expédition le 15/02/2024									
Commande TELEPHONE DAMIA MEYER du 14/02/2024, AR N° AR24/00503 du 14/02/2024									
100101001	BOITE OVALE 50G ANIS 3360100090101	1,7500	12,00	12	144	16,00%	1,4700	211,68	1

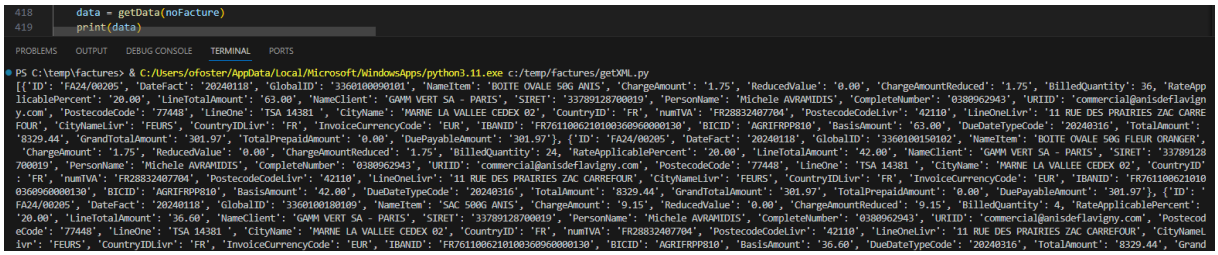
```python
def getNoFacture(pdf_path,file_path,errorPath):
    text = ""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for i in range(len(reader.pages)):
            page = reader.pages[i]
            text += page.extract_text()
    if re.findall(r"FA \d{2}/\d{5}",text):
        return re.findall(r"FA \d{2}/\d{5}",text)[0].replace(' ','')
    else :
        destination_path = os.path.join(errorPath, os.path.basename(file_path))
        shutil.move(file_path, destination_path)
        print("err " + pdf_path + f"\n numéro de facture pas trouvé")
        logErr.error(pdf_path + f"\n numéro de facture pas trouvé")
        log.error(pdf_path + f"\n numéro de facture pas trouvé")
        return False
```

Puis on récupère les données sur python :

```
# Récupération des données à partir de la base de données
def getData(noFacture):
    # Commande SQL pour récupérer les données
    sqlCommand=''' commande sql
    '''

    server = 'PC-02095' #à modifier
    database = 'SILOG' #à modifier

    try:
        # Connexion à la base de données et exécution de la commande SQL
        conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
        with pyodbc.connect(conn_str) as conn:
            cursor = conn.cursor()
            cursor.execute(sqlCommand,noFacture)
            rows = cursor.fetchall()
            # Formatage des données en une liste de dictionnaires
            result = []
            for row in rows:
                # Traitement de chaque ligne et ajout au résultat
                result.append({
                    # Mappage des colonnes de la base de données aux clés des dictionnaires
                    # et conversion des types si nécessaire
                    'ID': row[0], #numéro de la facture
                    'DateFact': row[1], #date de la facture
                    'GlobalID': row[2].replace(' ',''), #code ean du produit
                    'NameItem': checkString(row[3]), #nom du produit
                    'ChargeAmount': '%.2f' % float(row[4]), #prix produit
                    'ReducedValue': '%.2f' % float(row[5]), #remise
                    'ChargeAmountReduced': '%.2f' % float(row[6]), #prix produit - remise
                    'BilledQuantity': int(row[7]), #quantité de produit
                    'RateApplicablePercent': '%.2f' % float(row[8]), #% tva
                    'LineTotalAmount': '%.2f' % float(row[9]), # prix HT
                    'NameClient': checkString(row[10]), #nom du client
                    'SIRET': row[11].replace(' ',''), #siret du client
                    'PersonName': checkString(row[12]), #nom de la personne représentant le client
                    'CompleteNumber': row[13].replace(' ',''), # téléphone de la personne représentant
le client
                    'URIID': row[14], #email de la personne représentant le client
                    'PostecodeCode': row[15], #code postale du client
                    'LineOne': checkString(row[16]), #num porte + rue du client
                    'CityName': checkString(row[17]), #ville client
                    'CountryID': row[18], #num pays client
                    'numTVA': row[19].replace(' ',''), #numéro TVA
                    'PostecodeCodeLivr': checkString(row[20]), #code postale livraison
                    'LineOneLivr': checkString(row[21]), #num porte + rue de livraison
                    'CityNameLivr': checkString(row[22]), #nom ville de livraison
                    'CountryIDLivr': row[23], #num pays de livraison
                    'InvoiceCurrencyCode': row[24], #code devise
                    'IBANID': row[25].replace(' ',''), #iban
                    'BICID': row[26].replace(' ',''), #bic
                    'BasisAmount': '%.2f' % float(row[27]),# prix ht
                    'DueDateTypeCode': row[28], #data d'échéance
                    'TotalAmount': '%.2f' % float(row[29]), #prix tcc
                    'GrandTotalAmount': '%.2f' % float(row[30]), #prix ttc
                    'TotalPrepaidAmount': '%.2f' % float(row[31]), #quantité déjà payé
                    'DuePayableAmount': '%.2f' % float(row[32]), #quantité à payé
                    'CodeClient' : str(row[33])
                })
            return result
    except pyodbc.Error as ex:
        error_message = f"Erreur de connection à la base de données avec {noFacture}: {ex}"
        logErr.error(error_message)
        log.error(error_message)
        raise RuntimeError(error_message) from ex
    except Exception as ex:
        error_message = f"Une erreur est survenue avec {noFacture}: {ex}"
        logErr.error(error_message)
        log.error(error_message)
        raise RuntimeError(error_message) from ex
```



La variable `data` est un tableau de dictionnaires.

Chaque valeur dans la table représente un produit dans la facture.

Produit
ID
DateFact
GlobalID
NameItem
ChargeAmount
ReducedValue
ChargeAmountReduced
BilledQuantity
RateApplicablePercent
LineTotalAmount
NameClient
SIRET
PersonName
CompleteNumber
PostecodeCode
LineOne
Cityname
CountryID
numTVA
PostecodeCodeLivr
LineOneLivr
CityNameLivr
CountryIDLivr
InvoiceCurrencyCode
IBANID
BICID
BasisAmount
DueDateTypeCode
TotalAmount
GrandTotalAmount
TotalPrepaidAmount
DuePayableAmount
CodeClient

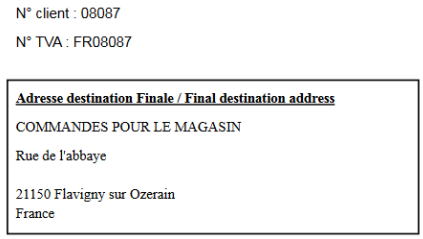
On fait passer les données par des fonctions pour vérifier la validité les données et de signaler l'utilisateur aux erreurs des valeurs dans la base.

```
# Vérification des données pour s'assurer qu'il n'y a pas de valeurs manquantes
def verifData(data, noFacture,file_path,errorPath):
    vide = []
    for i in range(len(data)):
        for key, val in data[i].items():
            if val == '':
                vide.append((i+1,key))
    if vide:
        error_message = f"{noFacture}: Il y a des données manquantes dans les lignes et colonnes:\n " +
'\n'.join([f"({idx}, '{key}')" for idx, key in vide])
        destination_path = os.path.join(errorPath, os.path.basename(file_path))
        shutil.move(file_path, destination_path)
        print("err " + error_message)
        logErr.error(error_message)
        log.error(error_message)
        return False
    return True

def verifFormat(data, noFacture, file_path, errorPath):
    emailFormat = re.compile(r'^[\w\.-]+@[\w\.-]+\.\w+$')
    destination_path = os.path.join(errorPath, os.path.basename(file_path))

    for entry in data:
        if (len(entry['GlobalID']) != 13) or not(entry['GlobalID'].isnumeric()):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['GlobalID']}, le format de l'EAN est invalide, il doit être composé
de 13 chiffres"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if (len(entry['SIRET']) != 14) or not(entry['SIRET'].isnumeric()):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['SIRET']}, le format du SIRET est invalide, il doit être composé de
14 chiffres"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if (len(entry['CompleteNumber']) != 10) or not(entry['CompleteNumber'].isnumeric()):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['CompleteNumber']}, le format du numéro de téléphone du client est
invalide, il doit être composé de 10 chiffres"
            print("err " + error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if not(emailFormat.match(entry['URIID'])):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['URIID']}, le format de l'adresse e-mail du client est invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if not(entry['numTVA'][0:2].isalpha()) or len(entry['numTVA']) < 8:
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['numTVA']} le format du numéro de TVA est invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if not(entry['IBANID'][0:2].isalpha()) or len(entry['IBANID']) < 14 or len(entry['IBANID']) > 34:
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['IBANID']} le format de l'IBAN est invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if len(entry['BICID']) < 8 or len(entry['BICID']) > 11:
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['BICID']} le format du numéro BIC est invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
    return True
```

L’entreprise envoie certaines factures à lui-même, donc on ne va pas générer le fichier xml, on va le déplacer le fichier PDF dans un dossier


N° client : 08087

N° TVA : FR08087

Adresse destination Finale / Final destination address

COMMANDES POUR LE MAGASIN

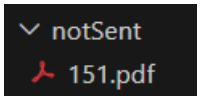
Rue de l'abbaye

21150 Flavigny sur Ozerain
France

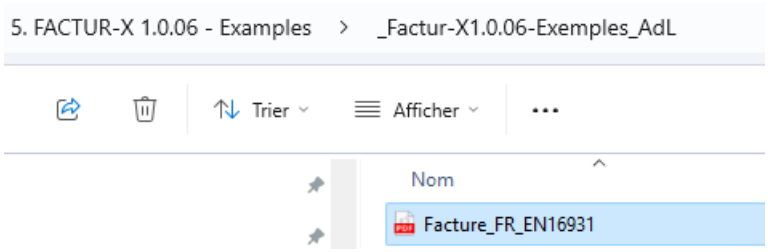
Donc on crée aussi cette fonction suivante :

```python
def verifClient(data,file_path):
    notSentPath = r"C:\temp\v4\factures\notSent"
    data = data[0]
    if data['CodeClient'] == '08087':
        shutil.move(file_path, notSentPath)
        return False
    else:
        return True
```

Exemple :



Ensuite, nous allons insérer les données dans des balises XML en utilisant ce fichier comme format :



On va tout d’abord gérer les balises qui se répète.

Pour les produits, on parcourt la liste `data` et on place les données dans les balises xml:

```
def setProduit(data):
    xmlProd = ''
    for i, row in enumerate(data):
        xmlProd += f'''
            <ram:IncludedSupplyChainTradeLineItem>
                <ram:AssociatedDocumentLineDocument>
                    <ram:LineID>{i+1}</ram:LineID>
                </ram:AssociatedDocumentLineDocument>
                <ram:SpecifiedTradeProduct>
                    <ram:GlobalID schemeID="0160">{row['GlobalID']}</ram:GlobalID>
                    <ram:Name>{row['NameItem']}</ram:Name>
                </ram:SpecifiedTradeProduct>
                <ram:SpecifiedLineTradeAgreement>
                    <ram:GrossPriceProductTradePrice>
                        <ram:ChargeAmount>{row['ChargeAmount']}</ram:ChargeAmount>
            '''
        if row['ReducedValue'] != '0.00':
            xmlProd += f'''
                <ram:AppliedTradeAllowanceCharge>
                    <ram:ChargeIndicator>
                        <udt:Indicator>false</udt:Indicator>
                    </ram:ChargeIndicator>
                    <ram:ActualAmount>{row['ReducedValue']}</ram:ActualAmount>
                </ram:AppliedTradeAllowanceCharge>
            '''
        xmlProd +=f'''
                    </ram:GrossPriceProductTradePrice>
                    <ram:NetPriceProductTradePrice>
                        <ram:ChargeAmount>{row['ChargeAmountReduced']}</ram:ChargeAmount>
                    </ram:NetPriceProductTradePrice>
                </ram:SpecifiedLineTradeAgreement>
                <ram:SpecifiedLineTradeDelivery>
                    <ram:BilledQuantity
unitCode="C62">{int(row['BilledQuantity'])}</ram:BilledQuantity>
                </ram:SpecifiedLineTradeDelivery>
                <ram:SpecifiedLineTradeSettlement>
                    <ram:ApplicableTradeTax>
                    <ram:TypeCode>VAT</ram:TypeCode>
                    <ram:CategoryCode>S</ram:CategoryCode>
                    <ram:RateApplicablePercent>{'%.2f' %
float(row['RateApplicablePercent'])}</ram:RateApplicablePercent>
                    </ram:ApplicableTradeTax>
                    <ram:SpecifiedTradeSettlementLineMonetarySummation>
                    <ram:LineTotalAmount>{row['LineTotalAmount']}</ram:LineTotalAmount>
                    </ram:SpecifiedTradeSettlementLineMonetarySummation>
                </ram:SpecifiedLineTradeSettlement>
            </ram:IncludedSupplyChainTradeLineItem>
        '''
    return xmlProd
```

Pour les taux de tva, c'est plus complexe, on répète chaque balise par le taux de tva différent.

Ex :

```
<ram:ApplicableTradeTax>
  <ram:CalculatedAmount>16.38</ram:CalculatedAmount>
  <ram:TypeCode>VAT</ram:TypeCode>
  <ram:BasisAmount>81.90</ram:BasisAmount>
  <ram:CategoryCode>S</ram:CategoryCode>
  <ram:DueDateTypeCode>5</ram:DueDateTypeCode>
  <ram:RateApplicablePercent>20.00</ram:RateApplicablePercent>
</ram:ApplicableTradeTax>
<ram:ApplicableTradeTax>
  <ram:CalculatedAmount>29.87</ram:CalculatedAmount>
  <ram:TypeCode>VAT</ram:TypeCode>
  <ram:BasisAmount>543.00</ram:BasisAmount>
  <ram:CategoryCode>S</ram:CategoryCode>
  <ram:DueDateTypeCode>5</ram:DueDateTypeCode>
  <ram:RateApplicablePercent>5.50</ram:RateApplicablePercent>
</ram:ApplicableTradeTax>
```

BasisAmount : somme du prix HT par taux de tva

CalculatedAmount : (BasisAmount*taux_tva)/100, montant TVA

```python
def getTVA(data):
    vat_dict = {}  # Dictionnaire pour stocker les totaux TVA par type
    total_taxed_amount = 0  # Initialisation du montant total taxé à zéro
    # Parcours de chaque ligne de données
    for row in data:
        # Obtention du type de TVA en pourcentage avec 4 décimales
        vat_type = '%.4f' % float(row['RateApplicablePercent'])
        # Calcul du montant de la taxe pour le produit actuel
        prixHT = float(row['LineTotalAmount'])
        tax_amount = prixHT * float(vat_type) / 100
        # Ajout du montant de la taxe au montant total taxé
        total_taxed_amount += tax_amount
        # Si le type de TVA existe déjà dans le dictionnaire, mettez à jour ses valeurs
        if vat_type in vat_dict:
            vat_dict[vat_type]['SumProduits'] += prixHT
            vat_dict[vat_type]['TotalTVA'] += tax_amount
        else:
            # Si le type de TVA n'existe pas dans le dictionnaire, ajoutez-le avec des
valeurs initiales
            vat_dict[vat_type] = {
                'SumProduits': prixHT,
                'TotalTVA': tax_amount
            }
    # Retourne le dictionnaire de TVA et le montant total taxé
    return vat_dict, total_taxed_amount
```

```
PS C:\temp\factures> & C:/Users/ofoster/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/temp/factures/getXML.py
({'20.0000': {'SumProduits': 251.64, 'TotalTVA': 50.327999999999996}}, 50.327999999999996)
PS C:\temp\factures>
```

Type TVA :
SumProduits
TotalTVA
Total TVA

Fonction pour créer les balises :

```
def setTva(data):
    xmlTVA = ''
    # Obtenir les types de TVA et les valeurs associées pour les données fournies
    typeTVA = getTVA(data)[0]
    # Parcours de chaque type de TVA et de ses valeurs
    for vat_type, values in typeTVA.items():
        # Construction de la section XML pour chaque type de TVA
        xmlTVA += f'''
            <ram:ApplicableTradeTax>
                <ram:CalculatedAmount>{'%.2f' %
float(values['TotalTVA'])}</ram:CalculatedAmount>
                <ram:TypeCode>VAT</ram:TypeCode>
                <ram:BasisAmount>{values['SumProduits']}</ram:BasisAmount>
                <ram:CategoryCode>S</ram:CategoryCode>
                <ram:DueDateTypeCode>5</ram:DueDateTypeCode>
                <ram:RateApplicablePercent>{vat_type}</ram:RateApplicablePercent>
            </ram:ApplicableTradeTax>
        '''
    return xmlTVA
```

On a donc tous ce dont on a besoin pour générer le fichier xml.

```
def setXML(data):
    #récupérer la valeur de TVA
    valTVA = getTVA(data)[1]
    row = data[0]
    xmlCode=f'''<?xml version='1.0' encoding='UTF-8'?>
    <rsm:CrossIndustryInvoice
xmlns:qdt="urn:un:unece:uncefact:data:standard:QualifiedDataType:100"
xmlns:ram="urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEntity:1
00" xmlns:rsm="urn:un:unece:uncefact:data:standard:CrossIndustryInvoice:100"
xmlns:udt="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:100"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <rsm:ExchangedDocumentContext>
            <ram:GuidelineSpecifiedDocumentContextParameter>
            <ram:ID>urn:cen.eu:en16931:2017</ram:ID>
            </ram:GuidelineSpecifiedDocumentContextParameter>
        </rsm:ExchangedDocumentContext>
        <rsm:ExchangedDocument>
            <ram:ID>{row['ID']}</ram:ID>
            <ram:TypeCode>380</ram:TypeCode>
            <ram:IssueDateTime>
            <udt:DateTimeString format="102">{row['DateFact']}</udt:DateTimeString>
            </ram:IssueDateTime>
            <ram:IncludedNote>
            <ram:Content></ram:Content>
            </ram:IncludedNote>
        </rsm:ExchangedDocument>
        <rsm:SupplyChainTradeTransaction>
'''
    xmlCode += setProduit(data)

    xmlCode += f'''
<ram:ApplicableHeaderTradeAgreement>
    <ram:SellerTradeParty>
        <ram:Name>Anis De Flavigny</ram:Name>
        <ram:SpecifiedLegalOrganization>
        <ram:ID schemeID="0002">45739641818284</ram:ID>
    '''
    #remplacer siret par celui du vendeur
```

```
    xmlCode += f'''
        </ram:SpecifiedLegalOrganization>
        <ram:DefinedTradeContact>
        <ram:PersonName></ram:PersonName>
        <ram:TelephoneUniversalCommunication>
            <ram:CompleteNumber></ram:CompleteNumber>
        </ram:TelephoneUniversalCommunication>
        <ram:EmailURIUniversalCommunication>
            <ram:URIID schemeID="SMTP"></ram:URIID>
        </ram:EmailURIUniversalCommunication>
        </ram:DefinedTradeContact>
        <ram:PostalTradeAddress>
        <ram:PostcodeCode>21150</ram:PostcodeCode>
        <ram:LineOne>4 Rue de l'Abbaye</ram:LineOne>
        <ram:CityName>Flavigny-sur-Ozerain</ram:CityName>
        <ram:CountryID>FR</ram:CountryID>
        </ram:PostalTradeAddress>
        <ram:SpecifiedTaxRegistration>
        <ram:ID schemeID="VA">FR96392006516</ram:ID>
        </ram:SpecifiedTaxRegistration>
    </ram:SellerTradeParty>
    <ram:BuyerTradeParty>
        <ram:Name>{row['NameClient']}</ram:Name>
        <ram:SpecifiedLegalOrganization>
        <ram:ID schemeID="0002">12790703541348</ram:ID>
        '''
    #remplacer siret par celui du client
    xmlCode += f'''
        </ram:SpecifiedLegalOrganization>
        <ram:DefinedTradeContact>
        <ram:PersonName>{row['PersonName']}</ram:PersonName>
        <ram:TelephoneUniversalCommunication>
            <ram:CompleteNumber>{row['CompleteNumber']}</ram:CompleteNumber>
        </ram:TelephoneUniversalCommunication>
        <ram:EmailURIUniversalCommunication>
            <ram:URIID schemeID="SMTP">{row['URIID']}</ram:URIID>
        </ram:EmailURIUniversalCommunication>
        </ram:DefinedTradeContact>
        <ram:PostalTradeAddress>
            <ram:PostcodeCode>{row['PostecodeCode']}</ram:PostcodeCode>
            <ram:LineOne>{row['LineOne']}</ram:LineOne>
            <ram:CityName>{row['CityName']}</ram:CityName>
            <ram:CountryID>{row['CountryID']}</ram:CountryID>
        </ram:PostalTradeAddress>
        <ram:SpecifiedTaxRegistration>
        <ram:ID schemeID="VA">{row['numTVA']}</ram:ID>
        </ram:SpecifiedTaxRegistration>
    </ram:BuyerTradeParty>
    <ram:BuyerOrderReferencedDocument>
        <ram:IssuerAssignedID></ram:IssuerAssignedID>
    </ram:BuyerOrderReferencedDocument>
    <ram:ContractReferencedDocument>
        <ram:IssuerAssignedID></ram:IssuerAssignedID>
    </ram:ContractReferencedDocument>
</ram:ApplicableHeaderTradeAgreement>
<ram:ApplicableHeaderTradeDelivery>
    <ram:ShipToTradeParty>
        <ram:PostalTradeAddress>
        <ram:PostcodeCode>{row['PostecodeCodeLivr']}</ram:PostcodeCode>
        <ram:LineOne>{row['LineOneLivr']}</ram:LineOne>
        <ram:CityName>{row['CityNameLivr']}</ram:CityName>
        <ram:CountryID>{row['CountryIDLivr']}</ram:CountryID>
        </ram:PostalTradeAddress>
    </ram:ShipToTradeParty>
</ram:ApplicableHeaderTradeDelivery>
<ram:ApplicableHeaderTradeSettlement>
```

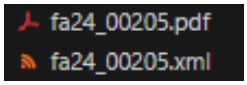
```
    <ram:PaymentReference>{row['ID']}</ram:PaymentReference>
    <ram:InvoiceCurrencyCode>{row['InvoiceCurrencyCode']}</ram:InvoiceCurrencyCode>
    <ram:SpecifiedTradeSettlementPaymentMeans>
        <ram:TypeCode>30</ram:TypeCode>
        <ram:Information></ram:Information>
        <ram:PayeePartyCreditorFinancialAccount>
            <ram:IBANID>{row['IBANID']}</ram:IBANID>
        </ram:PayeePartyCreditorFinancialAccount>
        <ram:PayeeSpecifiedCreditorFinancialInstitution>
        <ram:BICID>{row['BICID']}</ram:BICID>
        </ram:PayeeSpecifiedCreditorFinancialInstitution>
    </ram:SpecifiedTradeSettlementPaymentMeans>
    '''
    xmlCode += setTva(data)

    xmlCode +=f'''
            <ram:SpecifiedTradePaymentTerms>
                <ram:Description></ram:Description>
                <ram:DueDateDateTime>
                    <udt:DateTimeString
format="102">{row['DueDateTypeCode']}</udt:DateTimeString>
                </ram:DueDateDateTime>
            </ram:SpecifiedTradePaymentTerms>
            <ram:SpecifiedTradeSettlementHeaderMonetarySummation>
                <ram:LineTotalAmount>{'%.2f' % (float(row['GrandTotalAmount']) -
valTVA)}</ram:LineTotalAmount>
                <ram:TaxBasisTotalAmount>{'%.2f' % (float(row['GrandTotalAmount']) -
valTVA)}</ram:TaxBasisTotalAmount>
                <ram:TaxTotalAmount currencyID="{row['InvoiceCurrencyCode']}">{'%.2f' %
valTVA}</ram:TaxTotalAmount>
                <ram:GrandTotalAmount>{'%.2f' %
float(row['GrandTotalAmount'])}</ram:GrandTotalAmount>
                <ram:TotalPrepaidAmount>{'%.2f' %
float(row['TotalPrepaidAmount'])}</ram:TotalPrepaidAmount>
                <ram:DuePayableAmount>{'%.2f' %
float(row['DuePayableAmount'])}</ram:DuePayableAmount>
            </ram:SpecifiedTradeSettlementHeaderMonetarySummation>
        </ram:ApplicableHeaderTradeSettlement>
    </rsm:SupplyChainTradeTransaction>
</rsm:CrossIndustryInvoice>
    '''
    return xmlCode
```

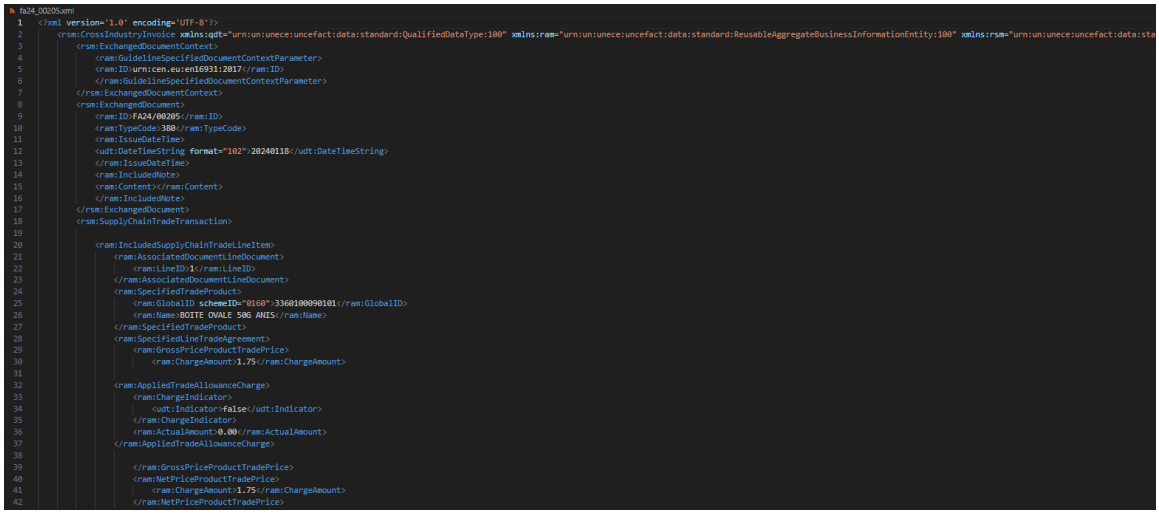
Finalement, Il faut générer le fichier xml

```
xmlCode = setXML(data)
with open(xmlPath, 'w', encoding='utf-8') as xmlFile:
      xmlFile.write(xmlCode)
```

Résultat :



fa24_00205.xml



```xml
<?xml version='1.0' encoding='UTF-8'?>
<rsm:CrossIndustryInvoice xmlns:qdt="urn:un:unece:uncefact:data:standard:QualifiedDataType:100" xmlns:ram="urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEntity:100" xmlns:rsm="urn:un:unece:uncefact:data:sta
    <rsm:ExchangedDocumentContext>
        <ram:GuidelineSpecifiedDocumentContextParameter>
        <ram:ID>urn:cen.eu:en16931:2017</ram:ID>
        </ram:GuidelineSpecifiedDocumentContextParameter>
    </rsm:ExchangedDocumentContext>
    <rsm:ExchangedDocument>
        <ram:ID>FA24/00205</ram:ID>
        <ram:TypeCode>380</ram:TypeCode>
        <ram:IssueDateTime>
        <udt:DateTimeString format="102">20240118</udt:DateTimeString>
        </ram:IssueDateTime>
        <ram:IncludedNote>
        <ram:Content></ram:Content>
        </ram:IncludedNote>
    </rsm:ExchangedDocument>
    <rsm:SupplyChainTradeTransaction>

        <ram:IncludedSupplyChainTradeLineItem>
            <ram:AssociatedDocumentLineDocument>
                <ram:LineID>1</ram:LineID>
            </ram:AssociatedDocumentLineDocument>
            <ram:SpecifiedTradeProduct>
                <ram:GlobalID schemeID="0160">3360100090101</ram:GlobalID>
                <ram:Name>BOITE OVALE 500G ANIS</ram:Name>
            </ram:SpecifiedTradeProduct>
            <ram:SpecifiedLineTradeAgreement>
                <ram:GrossPriceProductTradePrice>
                    <ram:ChargeAmount>1.75</ram:ChargeAmount>

            <ram:AppliedTradeAllowanceCharge>
                <ram:ChargeIndicator>
                    <udt:Indicator>false</udt:Indicator>
                </ram:ChargeIndicator>
                <ram:ActualAmount>0.00</ram:ActualAmount>
            </ram:AppliedTradeAllowanceCharge>

                </ram:GrossPriceProductTradePrice>
                <ram:NetPriceProductTradePrice>
                    <ram:ChargeAmount>1.75</ram:ChargeAmount>
                </ram:NetPriceProductTradePrice>
```

Code en entier :

```
import glob
import os
import pyodbc
import re
import PyPDF2
import shutil
'''
explication du programme:
On récupère les données essentiels dans la base de données
On place les données dans du code xml pour le générer ensuite
Dans le code xml, il y a des balises qui se répète:
- par produit
- par type de taxe tva (0, 5.5, 20, etc...)

'''
import logging
from datetime import datetime

current_datetime = datetime.now()
# Formater la date et l'heure
formatted_datetime = current_datetime.strftime("%Y-%m-%d")

# Configure logging for the first log file
log = logging.getLogger('log')
log.setLevel(logging.INFO)
log_formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
file_handler = logging.FileHandler(fr'C:\temp\v4\factures\logs\{formatted_datetime}-
log.log')
file_handler.setFormatter(log_formatter)
log.addHandler(file_handler)

logErr = logging.getLogger('logErr')
logErr.setLevel(logging.ERROR)
logErr_formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
fileErr_handler = logging.FileHandler(fr'C:\temp\v4\factures\logs\{formatted_datetime}-
logErreur.log')
fileErr_handler.setFormatter(logErr_formatter)
logErr.addHandler(fileErr_handler)

def getNoFacture(pdf_path,file_path,errorPath):
    text = ""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for i in range(len(reader.pages)):
            page = reader.pages[i]
            text += page.extract_text()
    if re.findall(r"FA \d{2}/\d{5}",text):
        return re.findall(r"FA \d{2}/\d{5}",text)[0].replace(' ','')
    else :
        destination_path = os.path.join(errorPath, os.path.basename(file_path))
        shutil.move(file_path, destination_path)
        print("err " + pdf_path + f"\n numéro de facture pas trouvé")
        logErr.error(pdf_path + f"\n numéro de facture pas trouvé")
        log.error(pdf_path + f"\n numéro de facture pas trouvé")
        return False

def checkString(text):
    text = text.replace('<','&lt;')
    text = text.replace('>','&gt;')
    text = text.replace('"','&quot;')
    text = text.replace('\'','&#39;')
    text = text.replace('&','&amp;')
    return text
```

```python
# Vérification des données pour s'assurer qu'il n'y a pas de valeurs manquantes
def verifData(data, noFacture,file_path,errorPath):
    vide = []
    for i in range(len(data)):
        for key, val in data[i].items():
            if val == '':
                vide.append((i+1,key))
    if vide:
        error_message = f"{noFacture}: Il y a des données manquantes dans les lignes et
colonnes:\n " + '\n'.join([f"({idx}, '{key}')" for idx, key in vide])
        destination_path = os.path.join(errorPath, os.path.basename(file_path))
        shutil.move(file_path, destination_path)
        print("err " + error_message)
        logErr.error(error_message)
        log.error(error_message)
        return False
    return True

def verifFormat(data, noFacture, file_path, errorPath):
    emailFormat = re.compile(r'^[\w\.-]+@[\w\.-]+\.\w+$')
    destination_path = os.path.join(errorPath, os.path.basename(file_path))

    for entry in data:
        if (len(entry['GlobalID']) != 13) or not(entry['GlobalID'].isnumeric()):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['GlobalID']}, le format de l'EAN est
invalide, il doit être composé de 13 chiffres"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if (len(entry['SIRET']) != 14) or not(entry['SIRET'].isnumeric()):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['SIRET']}, le format du SIRET est
invalide, il doit être composé de 14 chiffres"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if (len(entry['CompleteNumber']) != 10) or
not(entry['CompleteNumber'].isnumeric()):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['CompleteNumber']}, le format du numéro de
téléphone du client est invalide, il doit être composé de 10 chiffres"
            print("err " + error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if not(emailFormat.match(entry['URIID'])):
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['URIID']}, le format de l'adresse e-mail
du client est invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if not(entry['numTVA'][0:2].isalpha()) or len(entry['numTVA']) < 8:
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['numTVA']} le format du numéro de TVA est
invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if not(entry['IBANID'][0:2].isalpha()) or len(entry['IBANID']) < 14 or
len(entry['IBANID']) > 34:
```

```
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['IBANID']} le format de l'IBAN est
invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
        if len(entry['BICID']) < 8 or len(entry['BICID']) > 11:
            shutil.move(file_path, destination_path)
            error_message = f"{noFacture},{entry['BICID']} le format du numéro BIC est
invalide"
            print("err " +error_message)
            logErr.error(error_message)
            log.error(error_message)
            return False
    return True

def verifClient(data,file_path):
    notSentPath = r"C:\temp\v4\factures\notSent"
    data = data[0]
    if data['CodeClient'] == '08087':
        shutil.move(file_path, notSentPath)
        return False
    else:
        return True


# Récupération des données à partir de la base de données
def getData(noFacture):
    # Commande SQL pour récupérer les données
    sqlCommand='''
SELECT
    FAVE.NoFacture AS 'ID',
    CONVERT(VARCHAR, FAVE.DateFacturation, 112) AS 'DateFact',
    ARTICLE.CodeEAN AS 'GlobalID',
    ARTICLE.MotDirecteur AS 'NameItem',
    FAVC.PrixUnitDevises AS 'ChargeAmount',
    (FAVC.PrixUnitDevises - FAVC.PrixUnitNet) AS 'ReducedValue',
    FAVC.PrixUnitNet AS 'ChargeAmount2',
    FAVC.QuantiteFacturee AS 'BilledQuantity',
    FAVC.TauxTVA AS 'RateApplicablePercent',
    FAVC.MonthtRemisescomp AS 'LineTotalAmount',
    FAVE.NomClient AS 'NameClient',
    CLI.Siret AS 'SIRET',
    (CONTC.Prenom + ' ' + CONTC.nom) AS 'PersonName',
    CONTC.Telephone AS 'CompleteNumber',
    CONTC.Email AS 'URIID',
    FAVE.CodePostal AS 'PostecodeCode',
    (FAVE.AdressePartie1 + ' ' + FAVE.AdressePartie2) AS 'LineOne',
    FAVE.VilleFacturation AS 'CityName',
    COALESCE(CLI.CodePays, CLI.CodePays2) AS 'CountryID',
    CLI.NoIdentificatoCee AS 'numTVA',
    BDEE.CodePostal AS 'PostecodeCodeLivr',
    (BDEE.AdressePartie1 + ' ' + BDEE.AdressePartie2) AS 'LineOneLivr',
    BDEE.VilleLivraison AS 'CityNameLivr',
    BDEE.CodePaysLivraison AS 'CountryIDLivr',
    DEVISE.SigleDevise AS 'InvoiceCurrencyCode',
    BANQ.VarAlphaUtil2 AS 'IBANID',
    BANQ.VarAlphaUtil AS 'BICID',
    SUM(FAVC.MonthtRemisescomp) AS 'BasisAmount',
    CONVERT(VARCHAR, COALESCE(FAVE.DateEcheance1, FAVE.DateEcheance2, FAVE.DateEcheance3,
FAVE.DateEcheance4, FAVE.DateEcheance5), 112) AS 'DueDateTypeCode',
    (SELECT SUM(FAVC.PrixUnitDevises * FAVC.QuantiteFacturee) FROM FAVC WHERE
FAVC.NoFacture='FA24/00206' GROUP BY FAVC.NoFacture) AS 'LineTotalAmount2',
    FAVE.MontantTtc AS 'GrandTotalAmount',
    FAVE.AcompteEnDevises AS 'TotalPrepaidAmount',
```

```
    FAVE.MontantTtc-FAVE.AcompteEnDevises AS 'DuePayableAmount',
    CLI.CodeClient
FROM
    FAVE
JOIN
    FAVC ON FAVE.NoFacture = FAVC.NoFacture
JOIN
    ARTICLE ON ARTICLE.CodeArticle = FAVC.CodeArticleprest
JOIN
    CLI ON FAVE.CodeClient = CLI.CodeClient
JOIN
    CONTC ON CLI.VarAlphaUtil = CONTC.CodeUtilisateur
JOIN
    BDEE ON BDEE.NoBonExpedito = FAVC.NoBonExpedition
JOIN
    DEVISE ON FAVE.CodeDevise = DEVISE.CodeDevise
JOIN
    BANQ ON BANQ.CodeBanque = FAVE.CodeBanque1
WHERE
    FAVE.NoFacture = ?
    AND FAVC.PrestTalon = 'N'
GROUP BY
    FAVE.NoFacture, FAVE.DateFacturation, ARTICLE.CodeEAN, ARTICLE.MotDirecteur,
FAVC.PrixUnitDevises, FAVC.PrixUnitNet, FAVC.QuantiteFacturee, FAVC.TauxTVA,
FAVE.DateEcheance1, FAVE.DateEcheance2, FAVE.DateEcheance3, FAVE.DateEcheance4,
FAVE.DateEcheance5, FAVC.MonthtRemisescomp, FAVE.NomClient, CLI.Siret, CONTC.Prenom,
CONTC.nom, CONTC.Telephone, CONTC.Email, FAVE.CodePostal, FAVE.AdressePartie1,
FAVE.AdressePartie2, FAVE.VilleFacturation, CLI.CodePays, CLI.CodePays2,
CLI.NoIdentificatoCee, BDEE.CodePostal, BDEE.AdressePartie1, BDEE.AdressePartie2,
BDEE.VilleLivraison, BDEE.CodePaysLivraison, DEVISE.SigleDevise, BANQ.VarAlphaUtil2,
BANQ.VarAlphaUtil, FAVE.MontantTtc, FAVE.AcompteEnDevises, CLI.CodeClient
ORDER BY
    FAVE.NoFacture DESC
    '''

    server = 'PC-02095' #à modifier
    database = 'SILOG' #à modifier

    try:
        # Connexion à la base de données et exécution de la commande SQL
        conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
        with pyodbc.connect(conn_str) as conn:
            cursor = conn.cursor()
            cursor.execute(sqlCommand,noFacture)
            rows = cursor.fetchall()
            # Formatage des données en une liste de dictionnaires
            result = []
            for row in rows:
                # Traitement de chaque ligne et ajout au résultat
                result.append({
                    # Mappage des colonnes de la base de données aux clés des dictionnaires
                    # et conversion des types si nécessaire
                    'ID': row[0], #numéro de la facture
                    'DateFact': row[1], #date de la facture
                    'GlobalID': row[2].replace(' ',''), #code ean du produit
                    'NameItem': checkString(row[3]), #nom du produit
                    'ChargeAmount': '%.2f' % float(row[4]), #prix produit
                    'ReducedValue': '%.2f' % float(row[5]), #remise
                    'ChargeAmountReduced': '%.2f' % float(row[6]), #prix produit - remise
                    'BilledQuantity': int(row[7]), #quantité de produit
                    'RateApplicablePercent': '%.2f' % float(row[8]), #% tva
                    'LineTotalAmount': '%.2f' % float(row[9]), # prix HT
                    'NameClient': checkString(row[10]), #nom du client
                    'SIRET': row[11].replace(' ',''), #siret du client
```

```
                    'PersonName': checkString(row[12]), #nom de la personne représentant le
client
                    'CompleteNumber': row[13].replace(' ',''), # téléphone de la personne
représentant le client
                    'URIID': row[14], #email de la personne représentant le client
                    'PostecodeCode': row[15], #code postale du client
                    'LineOne': checkString(row[16]), #num porte + rue du client
                    'CityName': checkString(row[17]), #ville client
                    'CountryID': row[18], #num pays client
                    'numTVA': row[19].replace(' ',''), #numéro TVA
                    'PostecodeCodeLivr': checkString(row[20]), #code postale livraison
                    'LineOneLivr': checkString(row[21]), #num porte + rue de livraison
                    'CityNameLivr': checkString(row[22]), #nom ville de livraison
                    'CountryIDLivr': row[23], #num pays de livraison
                    'InvoiceCurrencyCode': row[24], #code devise
                    'IBANID': row[25].replace(' ',''), #iban
                    'BICID': row[26].replace(' ',''), #bic
                    'BasisAmount': '%.2f' % float(row[27]),# prix ht
                    'DueDateTypeCode': row[28], #data d'échéance
                    'TotalAmount': '%.2f' % float(row[29]), #prix tcc
                    'GrandTotalAmount': '%.2f' % float(row[30]), #prix ttc
                    'TotalPrepaidAmount': '%.2f' % float(row[31]), #quantité déjà payé
                    'DuePayableAmount': '%.2f' % float(row[32]), #quantité à payé
                    'CodeClient' : str(row[33])
                })
            return result
    except pyodbc.Error as ex:
        error_message = f"Erreur de connection à la base de données avec {noFacture}: {ex}"
        logErr.error(error_message)
        log.error(error_message)
        raise RuntimeError(error_message) from ex
    except Exception as ex:
        error_message = f"Une erreur est survenue avec {noFacture}: {ex}"
        logErr.error(error_message)
        log.error(error_message)
        raise RuntimeError(error_message) from ex

def setProduit(data):
    xmlProd = ''
    for i, row in enumerate(data):
        xmlProd += f'''
            <ram:IncludedSupplyChainTradeLineItem>
                <ram:AssociatedDocumentLineDocument>
                    <ram:LineID>{i+1}</ram:LineID>
                </ram:AssociatedDocumentLineDocument>
                <ram:SpecifiedTradeProduct>
                    <ram:GlobalID schemeID="0160">{row['GlobalID']}</ram:GlobalID>
                    <ram:Name>{row['NameItem']}</ram:Name>
                </ram:SpecifiedTradeProduct>
                <ram:SpecifiedLineTradeAgreement>
                    <ram:GrossPriceProductTradePrice>
                        <ram:ChargeAmount>{row['ChargeAmount']}</ram:ChargeAmount>
            '''
        if row['ReducedValue'] != '0.00':
            xmlProd += f'''
                <ram:AppliedTradeAllowanceCharge>
                    <ram:ChargeIndicator>
                        <udt:Indicator>false</udt:Indicator>
                    </ram:ChargeIndicator>
                    <ram:ActualAmount>{row['ReducedValue']}</ram:ActualAmount>
                </ram:AppliedTradeAllowanceCharge>
            '''
        xmlProd +=f'''
                    </ram:GrossPriceProductTradePrice>
                    <ram:NetPriceProductTradePrice>
                        <ram:ChargeAmount>{row['ChargeAmountReduced']}</ram:ChargeAmount>
```

```
                    </ram:NetPriceProductTradePrice>
                </ram:SpecifiedLineTradeAgreement>
                <ram:SpecifiedLineTradeDelivery>
                    <ram:BilledQuantity
unitCode="C62">{int(row['BilledQuantity'])}</ram:BilledQuantity>
                </ram:SpecifiedLineTradeDelivery>
                <ram:SpecifiedLineTradeSettlement>
                    <ram:ApplicableTradeTax>
                    <ram:TypeCode>VAT</ram:TypeCode>
                    <ram:CategoryCode>S</ram:CategoryCode>
                    <ram:RateApplicablePercent>{'%.2f' %
float(row['RateApplicablePercent'])}</ram:RateApplicablePercent>
                    </ram:ApplicableTradeTax>
                    <ram:SpecifiedTradeSettlementLineMonetarySummation>
                    <ram:LineTotalAmount>{row['LineTotalAmount']}</ram:LineTotalAmount>
                    </ram:SpecifiedTradeSettlementLineMonetarySummation>
                </ram:SpecifiedLineTradeSettlement>
            </ram:IncludedSupplyChainTradeLineItem>
        '''
    return xmlProd


def getTVA(data):
    vat_dict = {}  # Dictionnaire pour stocker les totaux TVA par type
    total_taxed_amount = 0  # Initialisation du montant total taxé à zéro
    # Parcours de chaque ligne de données
    for row in data:
        # Obtention du type de TVA en pourcentage avec 4 décimales
        vat_type = '%.4f' % float(row['RateApplicablePercent'])
        # Calcul du montant de la taxe pour le produit actuel
        prixHT = float(row['LineTotalAmount'])
        tax_amount = prixHT * float(vat_type) / 100
        # Ajout du montant de la taxe au montant total taxé
        total_taxed_amount += tax_amount
        # Si le type de TVA existe déjà dans le dictionnaire, mettez à jour ses valeurs
        if vat_type in vat_dict:
            vat_dict[vat_type]['SumProduits'] += prixHT
            vat_dict[vat_type]['TotalTVA'] += tax_amount
        else:
            # Si le type de TVA n'existe pas dans le dictionnaire, ajoutez-le avec des
valeurs initiales
            vat_dict[vat_type] = {
                'SumProduits': prixHT,
                'TotalTVA': tax_amount
            }
    # Retourne le dictionnaire de TVA et le montant total taxé
    return vat_dict, total_taxed_amount


def setTva(data):
    xmlTVA = ''
    # Obtenir les types de TVA et les valeurs associées pour les données fournies
    typeTVA = getTVA(data)[0]
    # Parcours de chaque type de TVA et de ses valeurs
    for vat_type, values in typeTVA.items():
        # Construction de la section XML pour chaque type de TVA
        xmlTVA += f'''
            <ram:ApplicableTradeTax>
                <ram:CalculatedAmount>{'%.2f' %
float(values['TotalTVA'])}</ram:CalculatedAmount>
                <ram:TypeCode>VAT</ram:TypeCode>
                <ram:BasisAmount>{values['SumProduits']}</ram:BasisAmount>
                <ram:CategoryCode>S</ram:CategoryCode>
                <ram:DueDateTypeCode>5</ram:DueDateTypeCode>
                <ram:RateApplicablePercent>{vat_type}</ram:RateApplicablePercent>
```

```
            </ram:ApplicableTradeTax>
        '''
    return xmlTVA


def setXML(data):
    #récupérer la valeur de TVA
    valTVA = getTVA(data)[1]
    row = data[0]
    xmlCode=f'''<?xml version='1.0' encoding='UTF-8'?>
    <rsm:CrossIndustryInvoice
xmlns:qdt="urn:un:unece:uncefact:data:standard:QualifiedDataType:100"
xmlns:ram="urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEntity:1
00" xmlns:rsm="urn:un:unece:uncefact:data:standard:CrossIndustryInvoice:100"
xmlns:udt="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:100"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <rsm:ExchangedDocumentContext>
            <ram:GuidelineSpecifiedDocumentContextParameter>
            <ram:ID>urn:cen.eu:en16931:2017</ram:ID>
            </ram:GuidelineSpecifiedDocumentContextParameter>
        </rsm:ExchangedDocumentContext>
        <rsm:ExchangedDocument>
            <ram:ID>{row['ID']}</ram:ID>
            <ram:TypeCode>380</ram:TypeCode>
            <ram:IssueDateTime>
            <udt:DateTimeString format="102">{row['DateFact']}</udt:DateTimeString>
            </ram:IssueDateTime>
            <ram:IncludedNote>
            <ram:Content></ram:Content>
            </ram:IncludedNote>
        </rsm:ExchangedDocument>
        <rsm:SupplyChainTradeTransaction>
'''
    xmlCode += setProduit(data)

    xmlCode += f'''
<ram:ApplicableHeaderTradeAgreement>
    <ram:SellerTradeParty>
        <ram:Name>Anis De Flavigny</ram:Name>
        <ram:SpecifiedLegalOrganization>
        <ram:ID schemeID="0002">45739641818284</ram:ID>
    '''
    #remplacer siret par celui du vendeur
    xmlCode += f'''
        </ram:SpecifiedLegalOrganization>
        <ram:DefinedTradeContact>
        <ram:PersonName></ram:PersonName>
        <ram:TelephoneUniversalCommunication>
            <ram:CompleteNumber></ram:CompleteNumber>
        </ram:TelephoneUniversalCommunication>
        <ram:EmailURIUniversalCommunication>
            <ram:URIID schemeID="SMTP"></ram:URIID>
        </ram:EmailURIUniversalCommunication>
        </ram:DefinedTradeContact>
        <ram:PostalTradeAddress>
        <ram:PostcodeCode>21150</ram:PostcodeCode>
        <ram:LineOne>4 Rue de l'Abbaye</ram:LineOne>
        <ram:CityName>Flavigny-sur-Ozerain</ram:CityName>
        <ram:CountryID>FR</ram:CountryID>
        </ram:PostalTradeAddress>
        <ram:SpecifiedTaxRegistration>
        <ram:ID schemeID="VA">FR96392006516</ram:ID>
        </ram:SpecifiedTaxRegistration>
    </ram:SellerTradeParty>
    <ram:BuyerTradeParty>
        <ram:Name>{row['NameClient']}</ram:Name>
```

```
        <ram:SpecifiedLegalOrganization>
        <ram:ID schemeID="0002">12790703541348</ram:ID>
        '''
    #remplacer siret par celui du client
    xmlCode += f'''
        </ram:SpecifiedLegalOrganization>
        <ram:DefinedTradeContact>
        <ram:PersonName>{row['PersonName']}</ram:PersonName>
        <ram:TelephoneUniversalCommunication>
            <ram:CompleteNumber>{row['CompleteNumber']}</ram:CompleteNumber>
        </ram:TelephoneUniversalCommunication>
        <ram:EmailURIUniversalCommunication>
            <ram:URIID schemeID="SMTP">{row['URIID']}</ram:URIID>
        </ram:EmailURIUniversalCommunication>
        </ram:DefinedTradeContact>
        <ram:PostalTradeAddress>
            <ram:PostcodeCode>{row['PostecodeCode']}</ram:PostcodeCode>
            <ram:LineOne>{row['LineOne']}</ram:LineOne>
            <ram:CityName>{row['CityName']}</ram:CityName>
            <ram:CountryID>{row['CountryID']}</ram:CountryID>
        </ram:PostalTradeAddress>
        <ram:SpecifiedTaxRegistration>
        <ram:ID schemeID="VA">{row['numTVA']}</ram:ID>
        </ram:SpecifiedTaxRegistration>
    </ram:BuyerTradeParty>
    <ram:BuyerOrderReferencedDocument>
        <ram:IssuerAssignedID></ram:IssuerAssignedID>
    </ram:BuyerOrderReferencedDocument>
    <ram:ContractReferencedDocument>
        <ram:IssuerAssignedID></ram:IssuerAssignedID>
    </ram:ContractReferencedDocument>
</ram:ApplicableHeaderTradeAgreement>
<ram:ApplicableHeaderTradeDelivery>
    <ram:ShipToTradeParty>
        <ram:PostalTradeAddress>
        <ram:PostcodeCode>{row['PostecodeCodeLivr']}</ram:PostcodeCode>
        <ram:LineOne>{row['LineOneLivr']}</ram:LineOne>
        <ram:CityName>{row['CityNameLivr']}</ram:CityName>
        <ram:CountryID>{row['CountryIDLivr']}</ram:CountryID>
        </ram:PostalTradeAddress>
    </ram:ShipToTradeParty>
</ram:ApplicableHeaderTradeDelivery>
<ram:ApplicableHeaderTradeSettlement>
    <ram:PaymentReference>{row['ID']}</ram:PaymentReference>
    <ram:InvoiceCurrencyCode>{row['InvoiceCurrencyCode']}</ram:InvoiceCurrencyCode>
    <ram:SpecifiedTradeSettlementPaymentMeans>
        <ram:TypeCode>30</ram:TypeCode>
        <ram:Information></ram:Information>
        <ram:PayeePartyCreditorFinancialAccount>
            <ram:IBANID>{row['IBANID']}</ram:IBANID>
        </ram:PayeePartyCreditorFinancialAccount>
        <ram:PayeeSpecifiedCreditorFinancialInstitution>
        <ram:BICID>{row['BICID']}</ram:BICID>
        </ram:PayeeSpecifiedCreditorFinancialInstitution>
    </ram:SpecifiedTradeSettlementPaymentMeans>
    '''
    xmlCode += setTva(data)

    xmlCode +=f'''
            <ram:SpecifiedTradePaymentTerms>
                <ram:Description></ram:Description>
                <ram:DueDateDateTime>
                    <udt:DateTimeString
format="102">{row['DueDateTypeCode']}</udt:DateTimeString>
                </ram:DueDateDateTime>
            </ram:SpecifiedTradePaymentTerms>
```

```
            <ram:SpecifiedTradeSettlementHeaderMonetarySummation>
                <ram:LineTotalAmount>{'%.2f' % (float(row['GrandTotalAmount']) -
valTVA)}</ram:LineTotalAmount>
                <ram:TaxBasisTotalAmount>{'%.2f' % (float(row['GrandTotalAmount']) -
valTVA)}</ram:TaxBasisTotalAmount>
                <ram:TaxTotalAmount currencyID="{row['InvoiceCurrencyCode']}">{'%.2f' %
valTVA}</ram:TaxTotalAmount>
                <ram:GrandTotalAmount>{'%.2f' %
float(row['GrandTotalAmount'])}</ram:GrandTotalAmount>
                <ram:TotalPrepaidAmount>{'%.2f' %
float(row['TotalPrepaidAmount'])}</ram:TotalPrepaidAmount>
                <ram:DuePayableAmount>{'%.2f' %
float(row['DuePayableAmount'])}</ram:DuePayableAmount>
            </ram:SpecifiedTradeSettlementHeaderMonetarySummation>
        </ram:ApplicableHeaderTradeSettlement>
    </rsm:SupplyChainTradeTransaction>
</rsm:CrossIndustryInvoice>
    '''
    return xmlCode

folderPath = r'C:\temp\v4\factures'
errorPath = r'C:\temp\v4\factures\errorPdf'

def main():
    for file_path in glob.glob(os.path.join(folderPath, "*.pdf")):
        if not os.path.exists(file_path):
            continue  # Passer au fichier suivant s'il n'existe pas

        noFacture = getNoFacture(file_path, file_path, errorPath)
        if not noFacture:
            continue  # Passer au fichier suivant si aucun numéro de facture n'a été trouvé

        data = getData(noFacture)
        if not data:
            error_message = "le numéro de facture contient aucune donnée"
            destination_path = os.path.join(errorPath, os.path.basename(file_path))
            shutil.move(file_path, destination_path)
            print("err " + noFacture + ': ' + error_message)
            continue  # Passer au fichier suivant si aucune donnée n'a été récupérée
        print(data)
        if not verifClient(data,file_path):
            continue

        if not verifData(data, noFacture, file_path, errorPath):
            continue  # Passer au fichier suivant si les données ne passent pas la
vérification


        if not verifFormat(data, noFacture, file_path, errorPath):
            continue  # Passer au fichier suivant si le format des données est incorrect

        filename = os.path.basename(file_path)
        xmlPath = os.path.join(folderPath, filename[:-4] + ".xml")
        xmlCode = setXML(data)
        with open(xmlPath, 'w', encoding='utf-8') as xmlFile:
            xmlFile.write(xmlCode)


#pour lancer le programme
#main()
```