	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>1/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	----------------------

## CHIMIE 1 & 2 2300

### ANALYSE FONCTIONNELLE De l'interface de gestion des modes opératoires chimie

#### Rédacteur

Nom /prénom : Rivière Julien

Fonction : Développeur Logiciels

Service : R&D

Signature : \_\_\_\_\_ Date : \_\_\_\_\_

#### Vérificateur

Nom /prénom : Bonami Laurent

Fonction : Technicien AQ

Service : Qualification

Signature : \_\_\_\_\_ Date : \_\_\_\_\_


#### Approbateur

Nom /prénom : Manette Netty


Fonction : Responsable AQ

Service : AQ Chimie


Signature : \_\_\_\_\_ Date : \_\_\_\_\_

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>2/156</b>

<i><b>Révision</b></i>	<i><b>Date</b></i>	<i><b>Objet de la révision</b></i>	<i><b>Version Interface</b></i>
A	21 Mars 2019	Création du document	2.0.1.3
B	12 Avril 2019	<p>-Ajout numéro de version complet dans info</p> <p>-La source n'est plus supprimée lors d'une exportation. Revue des diagrammes d'activités Exportation et Archivage.</p> <p>Iss001 :</p> <p>-Résolution de l'exception non géré lors d'une tentative d'écriture dans la base de données [en lecture seule]</p> <p>Iss002 :</p> <p>-Révision du système d'exception de la classe WReaderException et correction du retour d'erreur de WReader suite à un mot de passe inconnu du document Word</p> <p>VBTools :</p> <p>-Correction du bouton ok disponible sans sélection de fichier ou dossier</p> <p>-Tri des fichiers à choisir en fonction de l'extension de fichier souhaité</p>	2.0.1.4
C	23 Avril 2019	<p>Iss003 :</p> <p>-Révision du module de fermeture / libération des ressources Word [WReader] ainsi que des méthodes appelantes [WAction]</p>	2.0.1.5
D	30 Avril 2019	<p>Iss004 :</p> <p>-Correction de signets incontrôlables (ouvertures intempestives) dû à une méthode de gestion de signets ne les prenant pas en charge [WReader]</p>	2.0.1.6
E	06 Mai 2019	<p>-Réduction du nombre de caractères maximum dans la case audit trails de la vue d'impression. [VueImpression]</p> <p>-Ajout d'une phrase demandant de remplir les signets lors d'une consultation / impression [WAction]</p> <p>Iss005 :</p> <p>-Correction du nombre de page à imprimer après le changement de sélection de personnalisée à interval [VueImpression]</p>	2.0.1.7
F	10 Mai 2019	-Réduction du nombre de caractère pour la définition d'un log Fareva de 6 à 4 suite à un	2.0.1.8


	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>3/156</b>

		problème avec le log d'un chef d'équipe n'ayant que 5 caractères.	
<b>Révision</b>	<b>Date</b>	<b>Objet de la révision</b>	<b>Version Interface</b>

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>4/156</b>

## Table des matières


<b>1</b>	<b>OBJECT DU DOCUMENT</b>	<b>6</b>
<b>2</b>	<b>CONFIGURATION</b>	<b>6</b>
2.1	CONFIGURATION RECOMMANDEE	6
2.2	CONFIGURATION DE DEVELOPPEMENT	6
<b>3</b>	<b>DESCRIPTION GENERALE</b>	<b>7</b>
3.1	ENSEMBLE DE DOSSIER SPECIFIQUE	8
3.2	DESCRIPTION DE LA FONCTION D'IMPRESSION	8
3.3	RECAPITULATIF DE LA RELATION FONCTION / DOSSIER	9
<b>4</b>	<b>DESCRIPTION DE LA BASE DE DONNEES</b>	<b>10</b>
4.1	MODELE CONCEPTUEL DE DONNEES (MCD)	10
4.2	MODELE LOGIQUE DE DONNEES (MLD)	11
<b>5</b>	<b>DECLENCHEURS</b>	<b>12</b>
<b>6</b>	<b>MODE DE MARCHE</b>	<b>13</b>
6.1	CONSULTATION	13
6.2	IMPRESSION	14
6.3	IMPORTATION	15
6.4	EXPORTATION	16
6.5	ARCHIVAGE	17
<b>7</b>	<b>GESTION DES DROITS UTILISATEURS</b>	<b>18</b>
<b>8</b>	<b>CODE SOURCE</b>	<b>19</b>
8.1	LA PARTIE « CONTROLLER »	19
8.2	LA PARTIE « MODELS »	20
8.3	LA PARTIE « VIEWS »	21
<b>9</b>	<b>ANNEXE</b>	<b>22</b>
9.1	SCRIPT DE CREATION DE LA BASE DE DONNEES <i>SQLITE</i>	22
9.2	WACTION	32
9.3	WACTIONEXCEPTION	43
9.4	WREADER	44
9.5	WREADEREXCEPTION	58
9.6	COLOR	59
9.7	CONFIGURATION	60
9.8	OUTILS	63
9.9	ARCHDOSSPROD	65
9.10	USERCONFIG	68
9.11	AUDITRAILS	70
9.12	ENUMERATION	72
9.13	HISTODROITS	73
9.14	HISTOVERIFICATION	75
9.15	IMPRESSION	77
9.16	SIGNETS	79
9.17	UTILISATEUR	81
9.18	ABSTRACTDAO	82
9.19	DAOEXCEPTION	86
9.20	DAOFACTORY	87
9.21	IDAO	88

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>5/156</b>

9.22	IDAO_HISTO.....	89
9.23	SINGLETON.....	90
9.24	DAOAUDITRAILS.....	93
9.25	DAOIMPRESSION.....	96
9.26	DAOSIGNET.....	98
9.27	DAOUTILISATEUR.....	101
9.28	DAO_HDROIT.....	104
9.29	DAO_HVERIF.....	107
9.30	DAOSPECIFICTRANSACTION.....	110
9.31	DAOVIEWS.....	112
9.32	DAOREQUETES.....	114
9.33	DAOSERVICE.....	115
9.34	INITIALISATION.....	116
9.35	VUEAUDITRAILS.....	121
9.36	VUEIMPRESSION.....	128
9.37	VUEPRINCIPALE.....	133
9.38	VBTOOLS : DIALOGBOX.....	138
9.39	VBTOOLS : GESTIONDATAGRIDVIEW.....	149
9.40	VBTOOLS : VBTOOLSException.....	156

## Table des illustrations

Figure 1	Architecture de l'interface.....	7
Figure 2	La fonction d'impression.....	8
Figure 3	Relation fonctions / dossiers.....	9
Figure 4	Modèle Conceptuel de Données.....	10
Figure 5	Modèle Logique de Données.....	11
Figure 6	Diagramme d'activité : Consultation.....	13
Figure 7	Diagramme d'activité : Impression.....	14
Figure 8	Diagramme d'activité : Importation.....	15
Figure 9	Diagramme d'activité : Exportation.....	16
Figure 10	Diagramme d'activité : Archivage.....	17
Figure 11	Organisation générale du code source.....	19
Figure 12	La partie "Controller".....	19
Figure 13	La partie « Models ».....	20
Figure 14	La partie « Views ».....	21

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>6/156</b>

## 1 Object du document

L'objectif du document est de décrire le fonctionnement de l'interface de gestion des modes opératoires. Il a aussi pour but de décrire le fonctionnement de l'interface Mode Opérateur entre l'utilisateur et sa base de données.

## 2 Configuration

### 2.1 Configuration Recommandée

PC avec connexion intranet Valdepharm  
Windows 7 Service Pack 1 (x64/x86)  
Framework 4.0 et suite office 2010  
RAM 2Go  
CPU 1,6GHz

### 2.2 Configuration de développement

Le développement du logiciel a été réalisé en interne avec la configuration matériel et logiciels ci-dessous

Matériel :

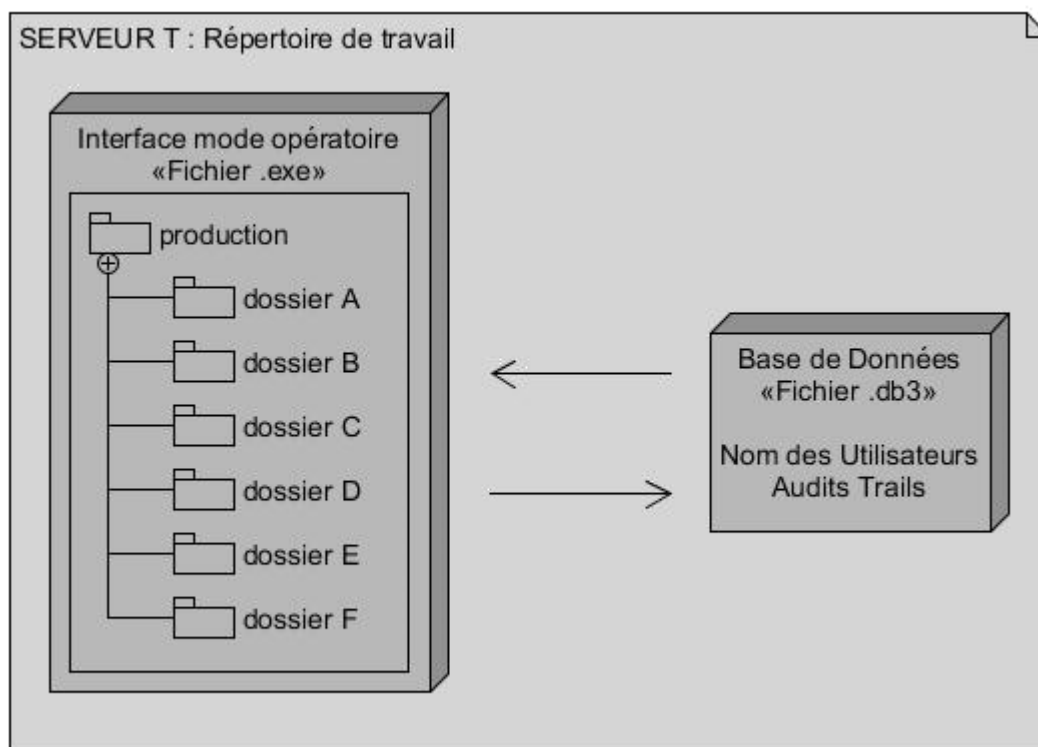
PC Portable Dell avec Intel Core I5-4310U et 4Go RAM  
Windows 7 Entreprise Service Pack 1 ( x86)  
Connexion intranet Valdepharm et accès au serveur T

Logiciels :

Office 2010  
Visual Studio Code  
Visual Basic Express 2010  
DB Browser for SQLite (v3.11.1)  
Umlet (standalone-14.3.0)  
JMerise

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>7/156</b>

### 3 Description générale



**Figure 1 Architecture de l'interface**

L'interface a pour objectif de gérer et de tracer les activités des documents Word de la production chimie via quatre fonctions principales :

- L'impression
- La consultation
- L'importation
- L'exportation
- L'archivage

Ces fonctions sont tracées et enregistrées dans une base de données. La fonction principale, celle pour laquelle ce logiciel a été conçu est la fonction d'impression. En effet, le but de ce logiciel lors de sa création était de gérer les impressions des modes opératoires afin de répondre au souci de la traçabilité documentaire, répondant ainsi aux exigences du Data Integrity.

Les modes opératoires servant aux opérations de fabrication sont crypté par l'interface dans le but d'éviter d'ouvrir le document (via Word) en dehors de l'interface. Ce système de cryptage oblige donc les utilisateurs à passer par l'interface pour la consultation et l'impression des documents de productions chimie.

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>8/156</b>

Ce système nécessite d'avoir dans le répertoire de l'application des dossiers connus ayant des rôles bien définis. Ces dossiers sont représentés par des lettres allant de A à F.

### 3.1 Ensemble de dossier spécifique

Les dossiers A et D ne sont pas utilisés, ils ne sont donc pas représentés par un dossier particulier.

Dossier Système	Fonction
A	N/D
B	Dossier de sauvegarde des documents Word après une opération d'importation.
C	Dossier de modification des documents Word (point d'entrée pour les nouveaux modes opératoires)
D	N/D
E	Dossier d'utilisation par la production, les documents présent dans ce répertoire sont des duplicata cryptés
F	Dossier d'archive des documents cryptés

### 3.2 Description de la fonction d'impression

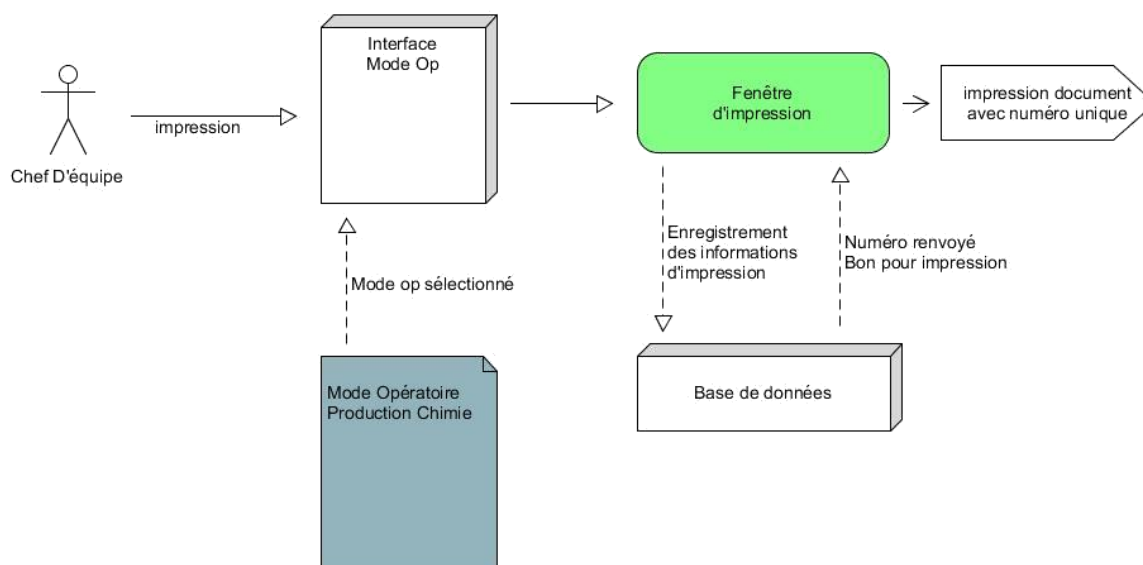


Figure 2 La fonction d'impression



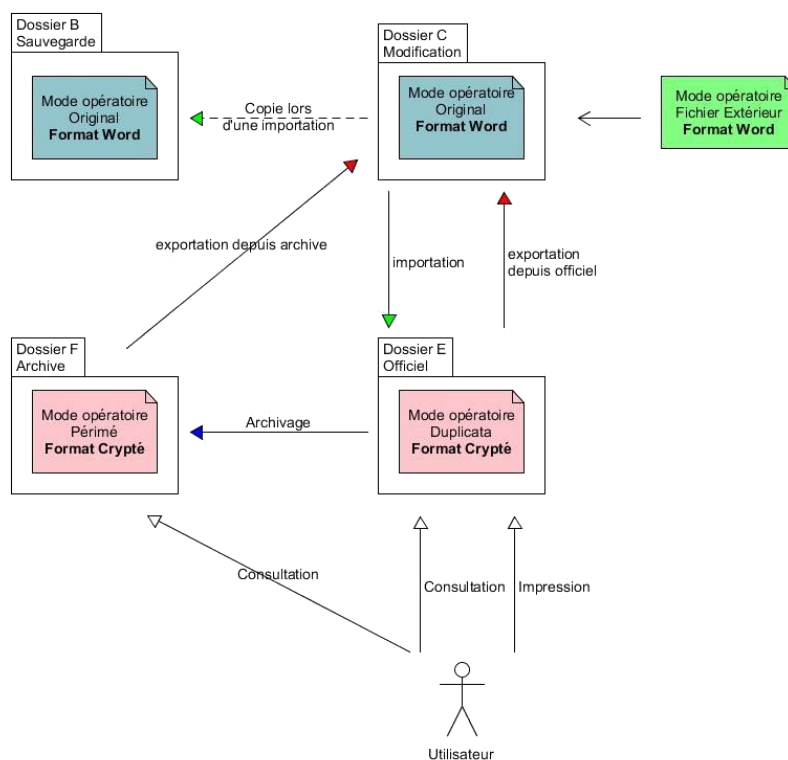
VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>9/156</b>

Ce système permet donc de retrouver toutes les informations d'impressions d'un mode opératoire (document Word) imprimé par un chef d'équipe grâce à un numéro d'impression unique renvoyé par la base de données.

Basé sur le même principe, nous pouvons retrouver les informations d'importation, d'exportation et d'archivage des documents Word.

Un tel système demande une architecture de dossier connue. Il y a donc des dossiers spécifiques dans le répertoire source de l'interface.

### 3.3 Récapitulatif de la relation fonction / dossier



**Figure 3 Relation fonctions / dossiers**

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>10/156</b>

## 4 Description de la base de données

L'objectif de la base de données est d'enregistrer l'ensemble des informations d'activité sur les documents Word à savoir l'impression, l'importation, l'exportation et l'archivage, ainsi que l'ensemble du personnel, via leurs logins Windows, ayant un droit sur ces actions.

La BDD est une base SQLite (\*.db3) présente dans le répertoire racine de l'application. Elle contient notamment des vues « views » permettant de rassembler les informations des tables de façon plus exploitable par un utilisateur ainsi que des déclencheurs « triggers » qui permettent de gérer les enregistrements utilisateurs à travers des règles de gestion spécifiques.

### 4.1 Modèle Conceptuel de Données (MCD)

Modèle conceptuel selon la méthode Merise réalisé grâce au logiciel JMerise

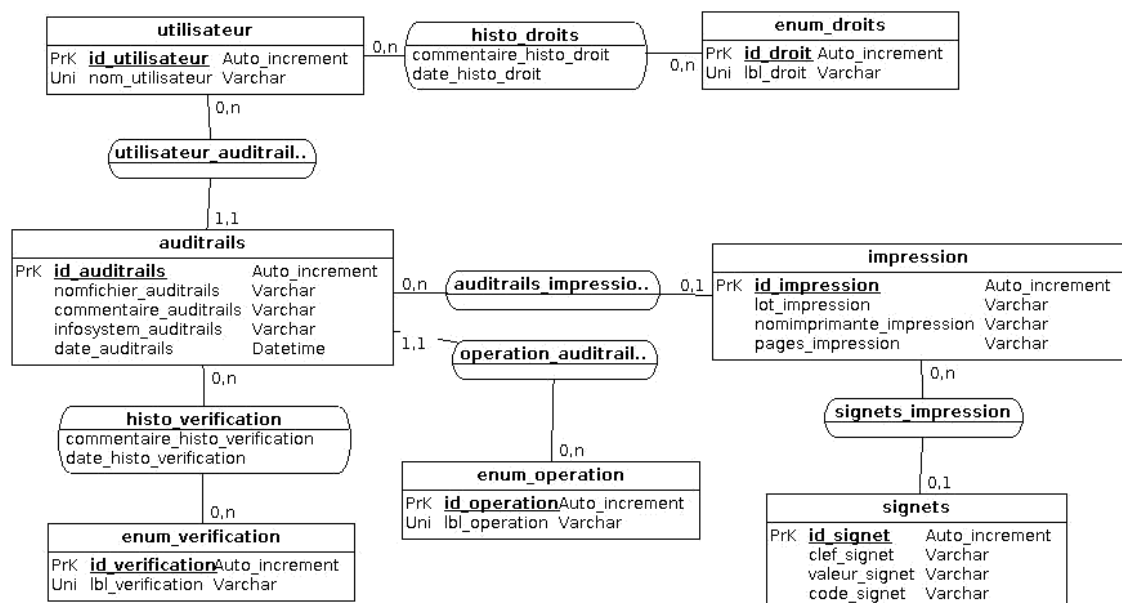


Figure 4 Modèle Conceptuel de Données

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	Interface de gestion des modes opératoires de la chimie	2300	FS	01	F	11/156

## 4.2 Modèle Logique de Données (MLD)

MLD généré automatiquement depuis le MCD précédent.

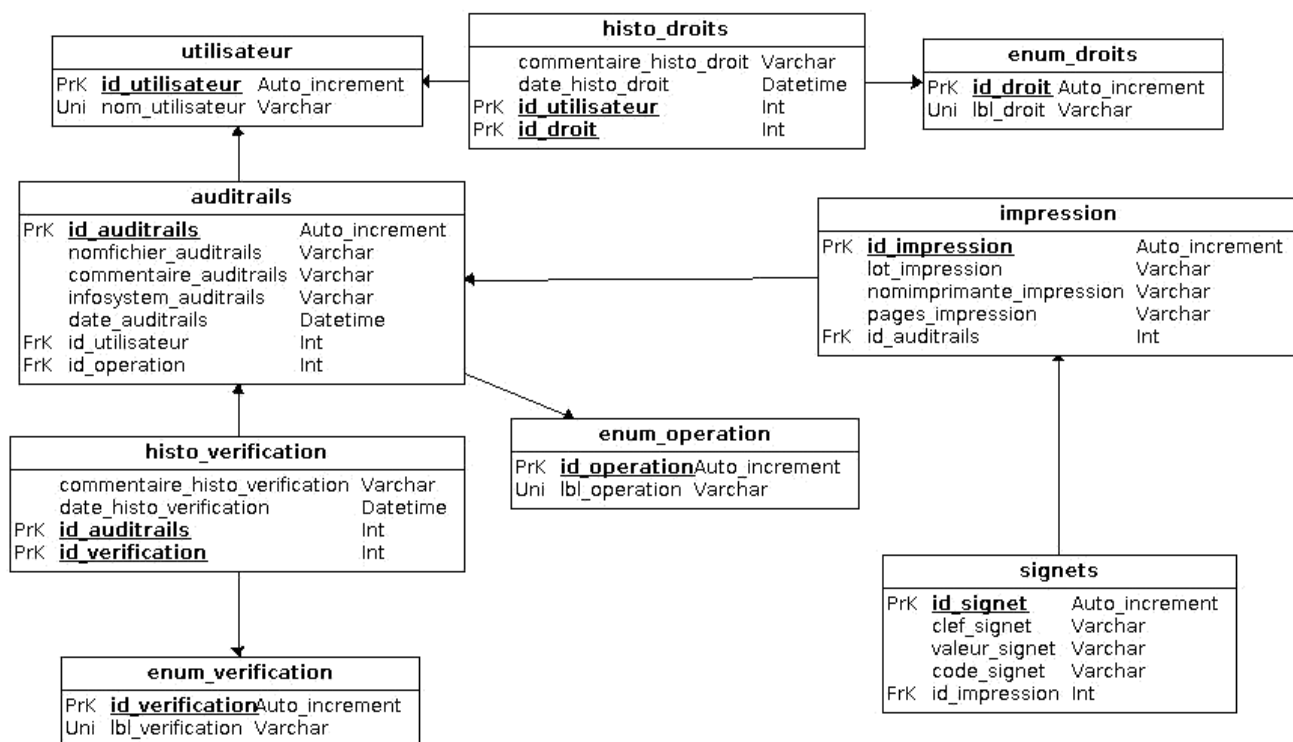



Figure 5 Modèle Logique de Données

Le schéma de la base de données généré contient donc neuf tables.

TABLE	ROLE
<b>Enum_droits</b>	Enumération des droits de l'interface (Fixe).
<b>Enum_verification</b>	Enumération des statuts de vérification de l'audit trails (Fixe).
<b>Enum_operation</b>	Enumération des types d'opération d'activité (Fixe).
<b>Utilisateur</b>	Liste des logins Windows pouvant avoir des droits dans l'interface mode opératoire.
<b>Auditrails</b>	Liste des activités des fichiers Word.
<b>Impression</b>	Information complémentaire lors d'un audit trails d'impression, renseignement sur l'imprimante ainsi que la liste des pages imprimées.
<b>Signets</b>	Information complémentaire lors d'un audit trails d'impression. Liste des signets entrés par l'utilisateur avant l'impression du document Word.
<b>Histo_droits</b>	Historique des droits attribués aux utilisateurs.
<b>Histo_verification</b>	Historique des statuts de vérification des audits trails.

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>12/156</b>

## 5 Déclencheurs

Il existe douze déclencheurs dans la base de données, intervenant en amont ou en aval d'un enregistrement (after ou before).

Nom du déclencheur	Description
<b>Enum_droits_before_delete</b>	Refuse l'opération de suppression des données sur la table enum_droits
<b>Enum_operation_before_delete</b>	Refuse l'opération de suppression des données sur la table enum_operation
<b>Enum_verification_before_delete</b>	Refuse l'opération de suppression des données sur la table enum_verification
<b>Auditrails_after_insert</b>	Lors de l'enregistrement de données dans la table auditrails, son historique de vérification (histo_verification) est initialisé à 0 (statut non vérifié)
<b>Auditrails_before_insert</b>	Contrôle que le numéro id de la table utilisateur existe avant enregistrement, dans le cas contraire, l'enregistrement est annulé
<b>Histo_droits_before_delete</b>	Refuse l'opération de suppression des données sur la table histo_droits
<b>Histo_droits_before_insert</b>	Contrôle que les numéros id des tables utilisateur et droits existent, dans le cas contraire, l'enregistrement est annulé
<b>Histo_verification_before_delete</b>	Refuse l'opération de suppression des données sur la table histo_verification
<b>Histo_verification_before_insert</b>	Contrôle que les numéros id des tables verification et auditrails existent, dans le cas contraire, l'enregistrement est annulé
<b>Impression_before_insert</b>	Contrôle que le numéro id de la table auditrails existe avant enregistrement, dans le cas contraire, l'enregistrement est annulé
<b>Signets_before_insert</b>	Contrôle que le numéro id de la table impression existe avant enregistrement, dans le cas contraire, l'enregistrement est annulé
<b>Utilisateur_after_insert</b>	Lors de l'enregistrement de données dans la table utilisateur, son historique de droits (histo_droits) est initialisé à 0 (statut Guest)

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>13/156</b>

## 6 Mode de marche

### 6.1 Consultation

Le mode consultation permet d'ouvrir un mode opératoire en format PDF avec un filigrane en travers du document de façon à ne pas l'utiliser pour des opérations de productions.

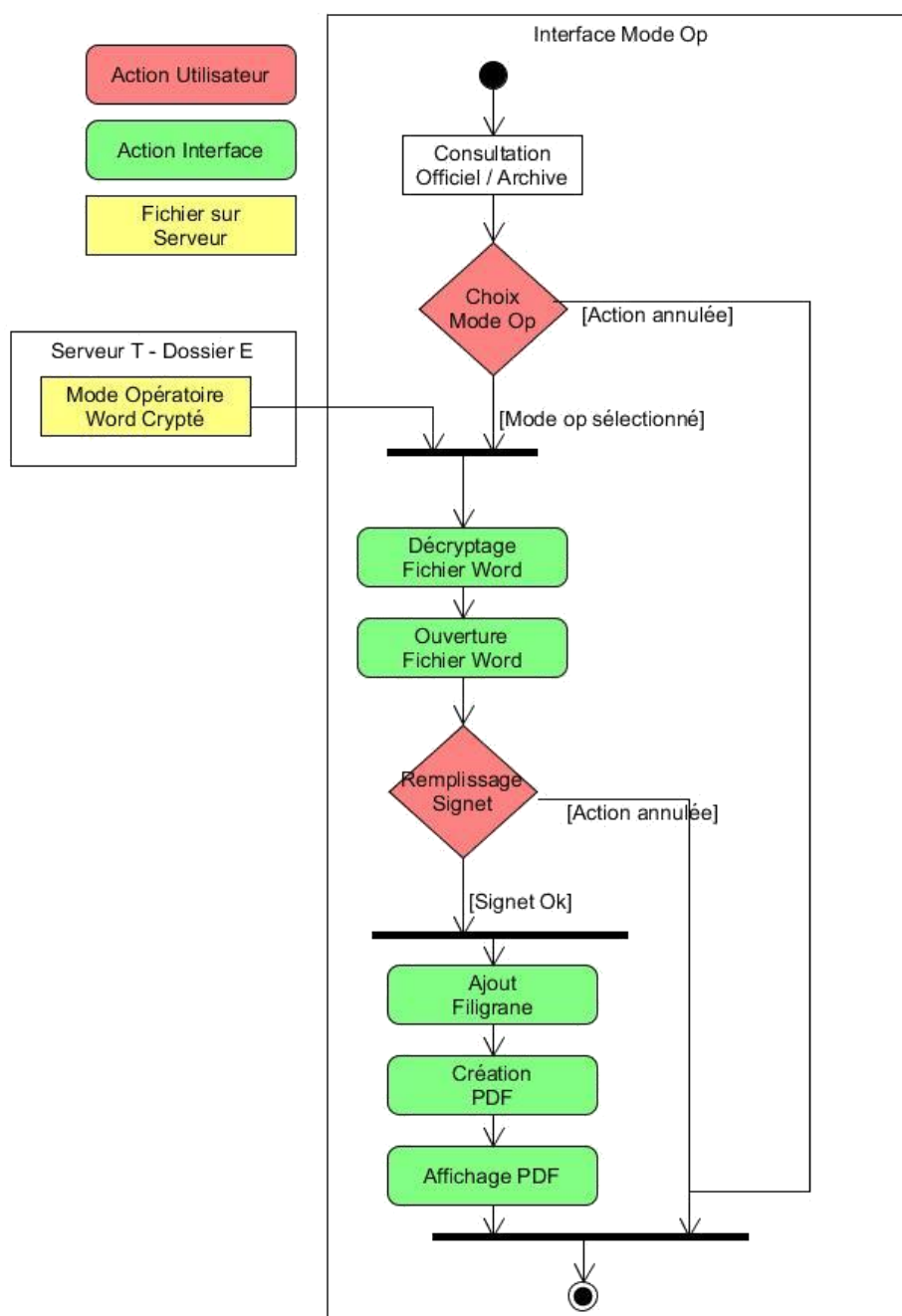


Figure 6 Diagramme d'activité : Consultation

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>14/156</b>

## 6.2 Impression

Le mode impression permet d'imprimer un document Word en enregistrant toutes les informations relatives à l'impression dans une base de données. Cette base renvoie un numéro d'enregistrement « Bon pour utilisation » qui sera placé en bas de page du document imprimé. Ce numéro permet de retrouver toutes les informations d'impression du document. Il s'agit du numéro id de la table audits.

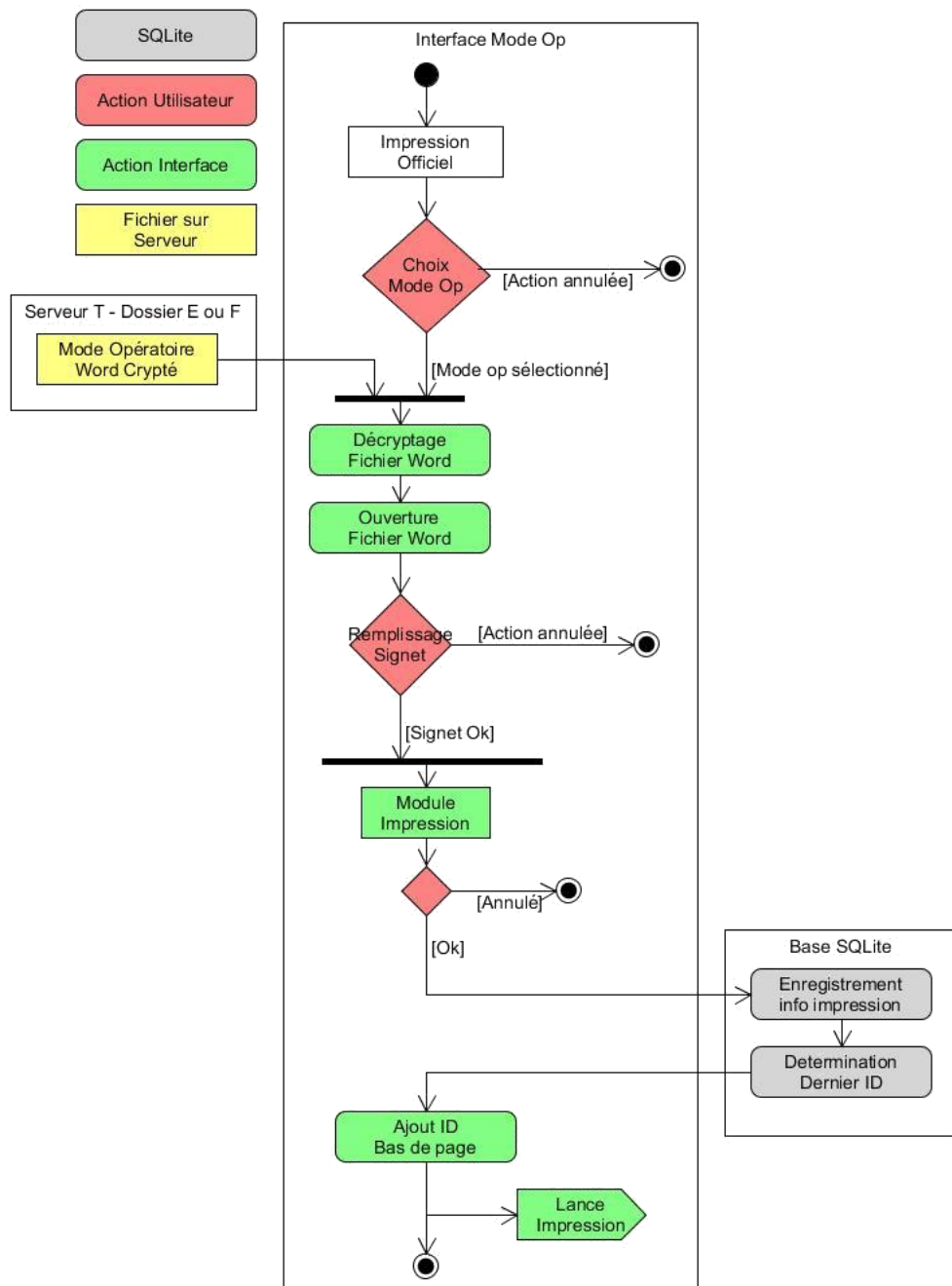



Figure 7 Diagramme d'activité : Impression

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>15/156</b>

### 6.3 Importation

Le mode importation permet d'intégrer des modes opératoires à l'architecture « ABCDEF » de l'interface. Le système se chargera de crypter le mode opératoire et d'en faire une sauvegarde.

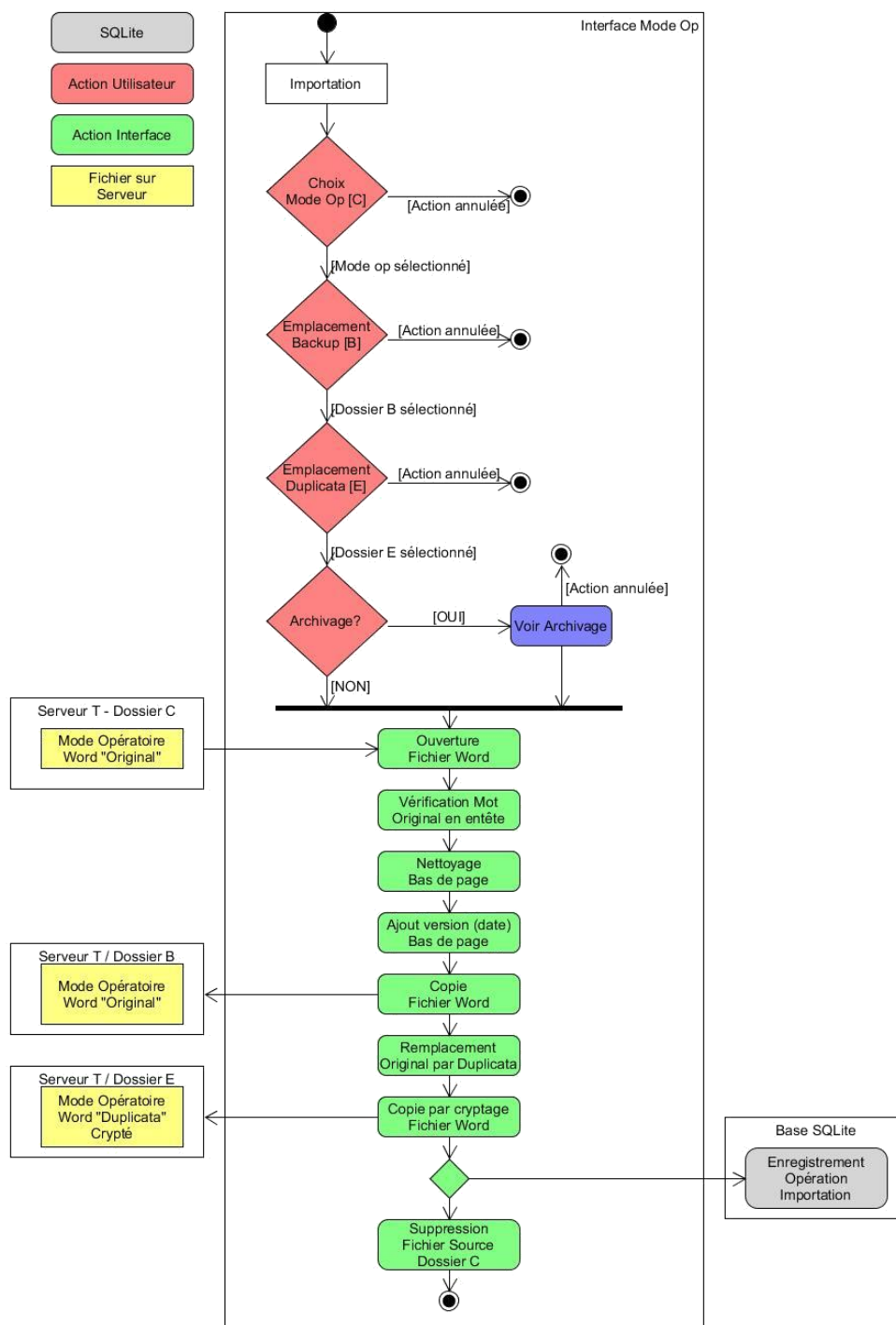


Figure 8 Diagramme d'activité : Importation

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	Interface de gestion des modes opératoires de la chimie	2300	FS	01	F	16/156

## 6.4 Exportation

Le mode exportation permet d'extraire un mode opératoire crypté de l'architecture logiciel et de pouvoir l'utiliser grâce à Word.

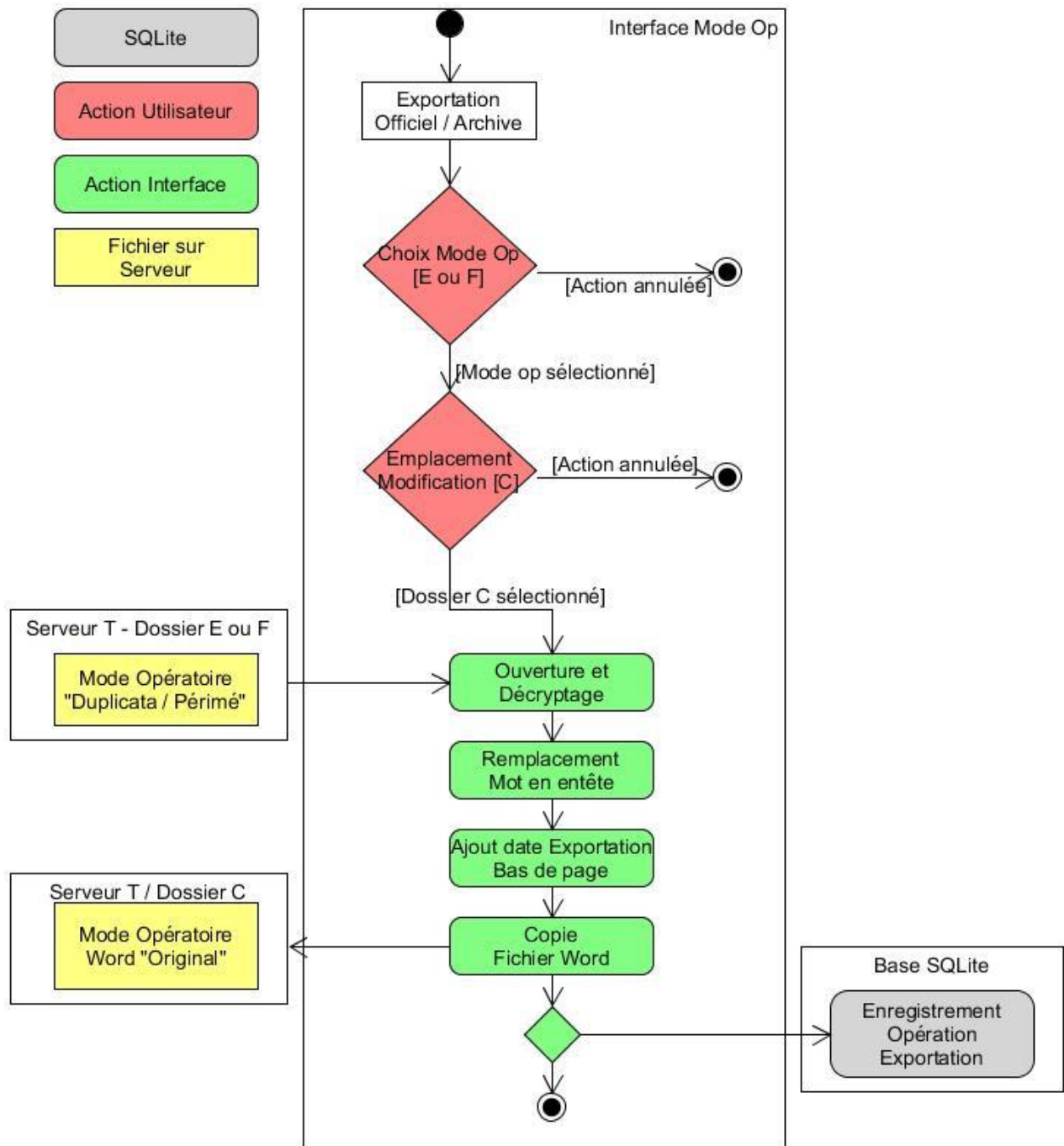


Figure 9 Diagramme d'activité : Exportation



VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>17/156</b>

## 6.5 Archivage

Le mode archivage permet d'archiver un mode opératoire crypté vers le dossier F (archive).  
Ce mode opératoire restera consultable.

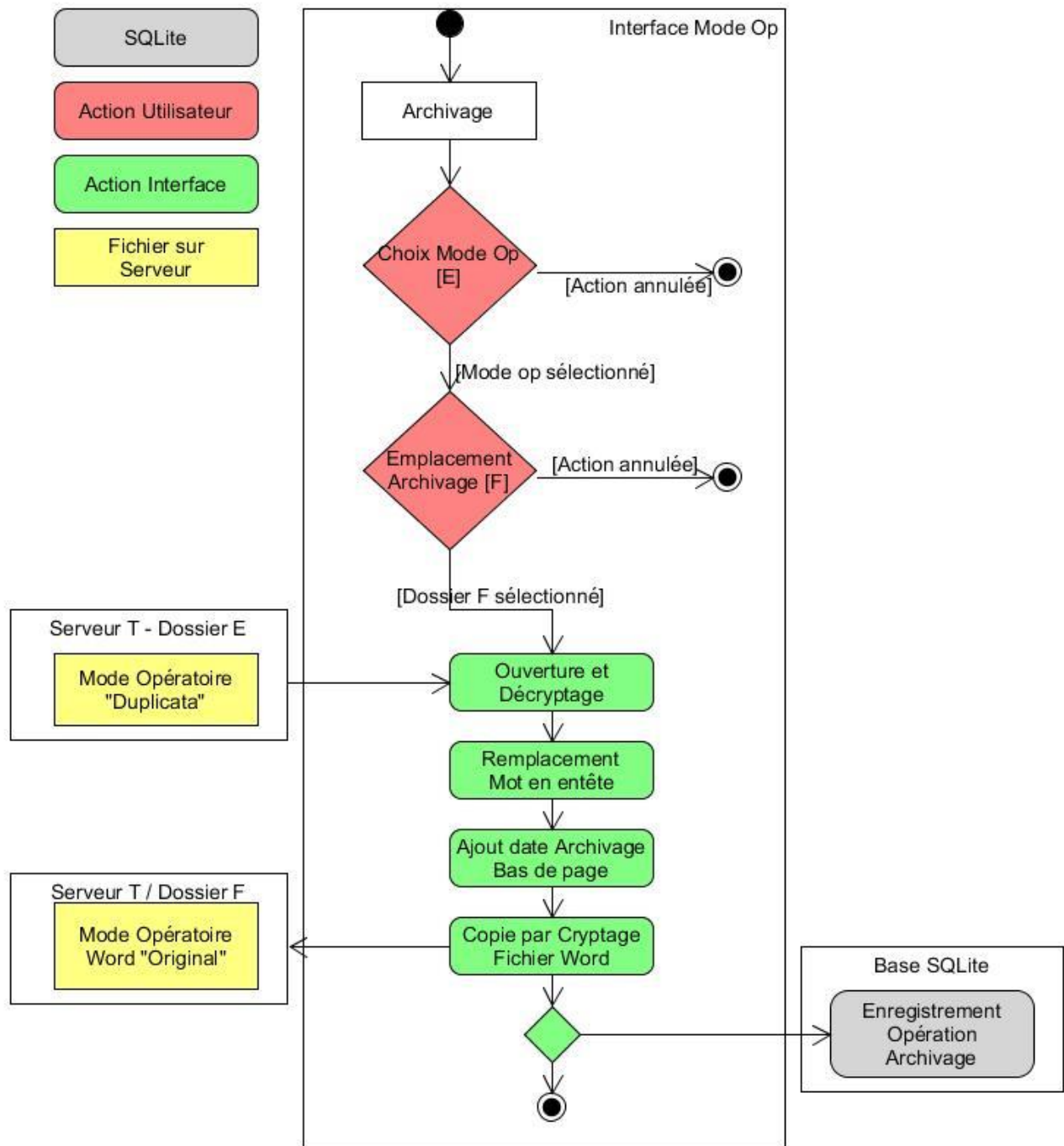



Figure 10 Diagramme d'activité : Archivage

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>18/156</b>

## 7 Gestion des droits utilisateurs

Il existe différents niveaux de droits utilisateurs pour l'interface mode opératoire.  
En fonction de ces droits, certains éléments du menu de l'interface s'afficheront ou pas.

En parallèle, les niveaux d'accès aux dossiers (Lecture / Ecriture) sont en corrélations avec les droits de l'interface. Le service gestionnaire des droits en lecture / écriture est le service informatique (IT).

Droit IT Interface	A	B [Sauvegarde]	C [Modification]	D	E [Officiel]	F [Archive]
Guest	N/D	Pas d'accès	Pas d'accès	N/D	Lecture	Lecture
User	N/D	Pas d'accès	Pas d'accès	N/D	Lecture	Lecture
UserAQ	N/D	Pas d'accès	Pas d'accès	N/D	Lecture	Lecture
KeyUser	N/D	Pas d'accès	Ecriture	N/D	Ecriture	Ecriture
AdminAQ	N/D	Ecriture	Ecriture	N/D	Ecriture	Ecriture
AdminDvlp	N/D	Ecriture	Ecriture	N/D	Ecriture	Ecriture

Par exemple, un utilisateur KeyUser devra au minimum avoir des accès Ecriture sur les dossiers C, E et F pour utiliser l'interface en tant que KeyUser. Dans le cas où ces minimaux ne sont pas respectés, l'interface réduira les droits à Guest.

Il est également possible de bloquer l'ouverture de l'interface à un utilisateur grâce à la fonction « Bloquer ».

Ci-dessous les fonctions de l'interface disponible en fonction des droits utilisateurs.

	Guest	User	User AQ	Key User	Admin AQ	Admin Dvlp
Utilisateur : Consultation	X	X	X	X	X	X
Utilisateur : Impression		X	X	X	X	X
Administrateur : Importation					X	X
Administrateur : Exportation				X	X	X
Administrateur : Archivage				X	X	X
Outils : Audit Trails			X		X	X
Développeur : Conversion ModeOp						X

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>19/156</b>

## 8 Code Source

La version du code source correspondant à cette version « 2300 FS 01 A » est 2.0.0.0.

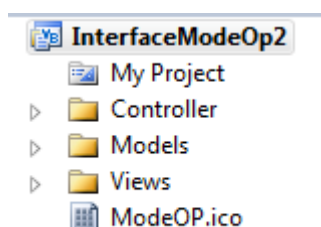
Le code source de la dll VBTools est disponible sur internet à l'adresse ci-dessous :

<https://github.com/ChatNoir76/VBTools>

Le code source de l'interface est disponible sur internet à l'adresse si dessous

<https://github.com/ChatNoir76/InterfaceModeOp>

Il est organisé, sous Visual Basic express 2010, de la façon suivante :

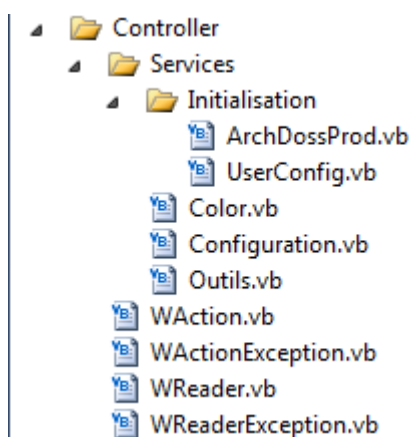


**Figure 11 Organisation générale du code source**

L'ensemble du code source est présent en annexe de ce document.

### 8.1 La partie « controller »

La partie « Controller » contient le code de type algorithmique qui détermine les actions à réaliser en fonction des choix utilisateurs.



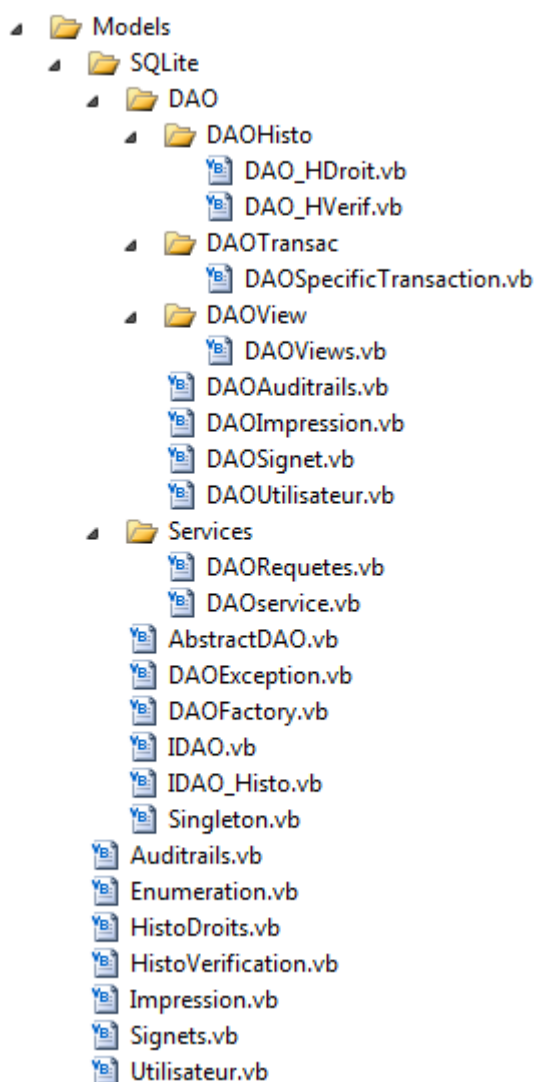
**Figure 12 La partie "Controller"**

Le module WAction joue le rôle de chef d'orchestre en fonction des actions d'impression, d'importation, d'exportation, de consultation et d'archivage. la classe WReader gère l'intégralité du document word. Le WAction appelle les fonctions de la classe WReader en fonction du choix de l'utilisateur.

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>20/156</b>

## 8.2 La partie « Models »

La partie « models » contient les classes nécessaires et suffisantes pour la communication entre l'interface et sa base de données. Les modèles sont utilisés par la couche DAO pour gérer les opérations de lecture, mise à jour, création et de suppression des données dans la base SQLite. Elle s'organise de la façon suivante :



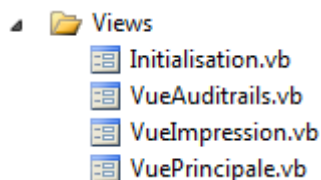
**Figure 13** La partie « Models »

La communication entre la base de données et son interface est réalisée grâce à un « singleton », ce qui permet de n'avoir qu'une seule instance et évite ainsi les appels multiples qui pourraient surcharger la base de données.

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>21/156</b>

### 8.3 La partie « Views »

La partie « views » contient toutes les vues (windows form) de l'interface.




**Figure 14 La partie « Views »**

« Initialisation » est la page d'initialisation qui détermine les droits de l'utilisateur en fonction de ses accès en lecture / écriture sur les dossiers de production.

« VuePrincipale » est la page qui permet de réaliser toutes les actions sur les documents Word et de consulter les informations de l'audit trails.

« VueAuditrails » contient toutes les informations de l'audit trails et elle permet également de changer les droits des utilisateurs.

« VueImpression » est la page affichée avant l'impression d'un mode opératoire.

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>22/156</b>

## 9 Annexe

### 9.1 Script de création de la base de données SQLite

```

-----
--      Script SQLite
-----

PRAGMA foreign_keys = FALSE; --à réaliser en manuel
drop trigger if exists enum_operation_before_delete;
drop trigger if exists enum_verification_before_delete;
drop trigger if exists enum_droits_before_delete;
drop trigger if exists histo_verification_before_delete;
drop trigger if exists histo_droits_before_delete;
-----

-- Table: utilisateur
-----

DROP TABLE IF EXISTS utilisateur;
CREATE TABLE utilisateur (
    id_utilisateur    INTEGER PRIMARY KEY AUTOINCREMENT ,
    nom_utilisateur   TEXT NOT NULL UNIQUE
);
-----


-- Table: enum_droits
-----

DROP TABLE IF EXISTS enum_droits;
CREATE TABLE enum_droits (
    id_droit          INTEGER PRIMARY KEY AUTOINCREMENT ,
    lbl_droit         TEXT NOT NULL UNIQUE
);
-----

-- Table: auditrails
-----

DROP TABLE IF EXISTS auditrails;
CREATE TABLE auditrails (
    id_auditrails      INTEGER PRIMARY KEY AUTOINCREMENT ,
    nomfichier_auditrails TEXT NOT NULL ,
    commentaire_auditrails TEXT NOT NULL ,
    infosystem_auditrails TEXT ,
    date_auditrails    NUMERIC NOT NULL ,
    id_utilisateur     INTEGER NOT NULL ,
    id_operation        INTEGER NOT NULL ,
    FOREIGN KEY (id_utilisateur) REFERENCES utilisateur (id_utilisateur),
    FOREIGN KEY (id_operation) REFERENCES operation (id_operation)
);

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>23/156</b>

-----  
-- Table: impression  
-----

```

DROP TABLE IF EXISTS impression;
CREATE TABLE impression (
    id_impression          INTEGER PRIMARY KEY AUTOINCREMENT ,
    lot_impression         TEXT ,
    nomimprimante_impression TEXT NOT NULL ,
    pages_impression       TEXT NOT NULL ,
    id_auditrails          INTEGER NOT NULL ,
    FOREIGN KEY (id_auditrails) REFERENCES auditrails (id_auditrails)
);

```

-----  
-- Table: signets  
-----

```

DROP TABLE IF EXISTS signets;
CREATE TABLE signets (
    id_signet             INTEGER PRIMARY KEY AUTOINCREMENT ,
    clef_signet           TEXT NOT NULL ,
    valeur_signet         TEXT NOT NULL ,
    code_signet           TEXT NOT NULL ,
    id_impression         INTEGER NOT NULL ,
    FOREIGN KEY (id_impression) REFERENCES impression (id_impression)
);

```

-----  
-- Table: enum\_verification  
-----

```

DROP TABLE IF EXISTS enum_verification;
CREATE TABLE enum_verification (
    id_verification       INTEGER PRIMARY KEY AUTOINCREMENT ,
    lbl_verification      TEXT NOT NULL UNIQUE
);

```

-----  
-- Table: enum\_operation  
-----

```

DROP TABLE IF EXISTS enum_operation;
CREATE TABLE enum_operation (
    id_operation          INTEGER PRIMARY KEY AUTOINCREMENT ,
    lbl_operation         TEXT NOT NULL UNIQUE
);


```

-----  
-- Table: histo\_droits  
-----

```

DROP TABLE IF EXISTS histo_droits;
CREATE TABLE histo_droits (
    commentaire_histo_droit TEXT ,

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>24/156</b>

```


date_histo_droit          TEXT ,
id_utilisateur           INTEGER NOT NULL ,
id_droit                 INTEGER NOT NULL ,
PRIMARY KEY (id_droit,id_utilisateur,date_histo_droit) ,

FOREIGN KEY (id_droit) REFERENCES enum_droits(id_droit),
FOREIGN KEY (id_utilisateur) REFERENCES utilisateur(id_utilisateur)
);
-----
-- Table: histo_verification
-----
DROP TABLE IF EXISTS histo_verification;
CREATE TABLE histo_verification (
    commentaire_histo_verification TEXT ,
    date_histo_verification        TEXT ,
    id_auditrails                  INTEGER NOT NULL ,
    id_verification                INTEGER NOT NULL ,
    PRIMARY KEY (id_auditrails,id_verification,date_histo_verification) ,

    FOREIGN KEY (id_auditrails) REFERENCES auditrails(id_auditrails),
    FOREIGN KEY (id_verification) REFERENCES enum_verification(id_verification)
);
-----
-- ForeignKeys : Activation
-----
PRAGMA foreign_keys = TRUE;
-----
-- Trigger : utilisateur
-----
drop trigger if exists utilisateur_after_insert;
create trigger utilisateur_after_insert after insert on utilisateur
begin
    UPDATE utilisateur
    SET nom_utilisateur = UPPER(nom_utilisateur)
    WHERE nom_utilisateur = NEW.nom_utilisateur;
    INSERT INTO histo_droits
    VALUES ("enregistrement auto",datetime('now','localtime'),NEW.id_utilisateur,0);
end;
-----
-- Trigger : auditrails
-----
drop trigger if exists auditrails_after_insert;
create trigger auditrails_after_insert after insert on auditrails
begin
    INSERT INTO histo_verification
    VALUES ("enregistrement auto",datetime('now','localtime'),NEW.id_auditrails,0);

```




	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>25/156</b>

```

end;
drop trigger if exists auditrails_before_insert;
create trigger auditrails_before_insert before insert on auditrails
begin
    SELECT CASE
    WHEN (select count(new.id_utilisateur) from utilisateur
        where utilisateur.id_utilisateur = new.id_utilisateur) <> 1
    THEN
        RAISE(ABORT, "L'utilisateur n'existe pas dans la liste des utilisateurs" )
    END;
end;
-----
-- Trigger : impression
-----
drop trigger if exists impression_before_insert;
create trigger impression_before_insert before insert on impression
begin
    SELECT CASE
    WHEN (select count(new.id_auditrails) from auditrails
        where auditrails.id_auditrails = new.id_auditrails) <> 1
    THEN
        RAISE(ABORT, "cet audit trails n'existe pas dans la liste d'enregistrement des
audits trails" )
    END;
end;
-----
-- Trigger : signets
-----
drop trigger if exists signets_before_insert;
create trigger signets_before_insert before insert on signets
begin
    SELECT CASE
    WHEN (select count(new.id_impression) from impression
        where impression.id_impression = new.id_impression) <> 1
    THEN
        RAISE(ABORT, "l'id d'impression sélectionné n'existe pas dans la liste
d'enregistrement des impressions" )
    END;
end;
-----
-- Trigger : histo_droits
-----
drop trigger if exists histo_droits_before_insert;
create trigger histo_droits_before_insert before insert on histo_droits
begin
    SELECT CASE


```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>26/156</b>

```

WHEN (select count(new.id_utilisateur) from utilisateur
      where utilisateur.id_utilisateur = new.id_utilisateur) <> 1
THEN
    RAISE(ABORT, "L'utilisateur n'existe pas dans la liste des utilisateurs" )
END;
SELECT CASE
WHEN (select count(new.id_droit) from enum_droits
      where enum_droits.id_droit = new.id_droit) <> 1
THEN
    RAISE(ABORT, "Ce droit n'existe pas dans la liste des droits" )
END;
end;
drop trigger if exists histo_droits_before_delete;
create trigger histo_droits_before_delete before delete on histo_droits
begin
    SELECT RAISE(ABORT, "la suppression de l'historique des droits utilisateurs n'est
pas autorisée" );
end;
-----
-- Trigger : histo_verification
-----
drop trigger if exists histo_verification_before_insert;
create trigger histo_verification_before_insert before insert on histo_verification
begin
    SELECT CASE
    WHEN (select count(new.id_auditrails) from auditrails
          where auditrails.id_auditrails = new.id_auditrails) <> 1
    THEN
        RAISE(ABORT, "cet audit trails n'existe pas dans la liste d'enregistrement des
audits trails" )
    END;
    SELECT CASE
    WHEN (select count(new.id_verification) from enum_verification
          where enum_verification.id_verification = new.id_verification) <> 1
    THEN
        RAISE(ABORT, "Ce statut de vérification n'existe pas dans la liste des
vérifications" )
    END;
end;
drop trigger if exists histo_verification_before_delete;
create trigger histo_verification_before_delete before delete on histo_verification
begin
    SELECT RAISE(ABORT, "la suppression de l'historique des vérifications n'est pas
autorisée" );
end;
-----


```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>27/156</b>

```

-- Trigger : enum_droits
-----
drop trigger if exists enum_droits_before_delete;
create trigger enum_droits_before_delete before delete on enum_droits
begin
    SELECT RAISE(ABORT, "la suppression des droits n'est pas autorisée" );
end;
-----
-- Trigger : enum_verification
-----
drop trigger if exists enum_verification_before_delete;
create trigger enum_verification_before_delete before delete on enum_verification
begin
    SELECT RAISE(ABORT, "la suppression des statuts de vérification n'est pas
autorisée" );
end;
-----
-- Trigger : enum_operation
-----
drop trigger if exists enum_operation_before_delete;
create trigger enum_operation_before_delete before delete on enum_operation
begin
    SELECT RAISE(ABORT, "la suppression des types d'opérations n'est pas autorisée" );
end;
-----
-- Views : Audit trails impression
-----
drop view if exists view_ATPrinter;
CREATE VIEW view_ATPrinter
AS
select st2.id_auditrails,
       st3.id_verification,
       st2."Bon Pour Utilisation",
       st2."Fichier ModeOp",
       st2."Commentaire AT",
       st2."Info Système",
       st2."Date d'impression",
       st2."Utilisateur",
       st2."Type Opération",
       st2."Numéro Lot",
       st2."Imprimante",
       st2."Page(s) Imprimée(s)",
       st1.commentaire_histo_verification as "Commentaire vérif",
       st3.lbl_verification as "Statut Vérification"
from histo_verification as st1,
    (select


```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>28/156</b>

```

t1.id_auditrails,
t1.id_auditrails as "Bon Pour Utilisation",
t1.nomfichier_auditrails as "Fichier ModeOp",
t1.commentaire_auditrails as "Commentaire AT",
t1.infosystem_auditrails as "Info Système",
datetime(t1.date_auditrails) as "Date d'impression",
t4.nom_utilisateur as "Utilisateur",
t3.lbl_operation as "Type Opération",
t2.lot_impression as "Numéro Lot",
t2.nomimprimante_impression as "Imprimante",
t2.pages_impression as "Page(s) Imprimée(s)"
from
    auditrails as t1,
    impression as t2,
    enum_operation as t3,
    utilisateur as t4
WHERE t1.id_auditrails = t2.id_auditrails
AND t1.id_utilisateur = t4.id_utilisateur
AND t1.id_operation = t3.id_operation) as st2,
enum_verification as st3
where st1.id_auditrails = st2.id_auditrails
AND st1.id_verification = st3.id_verification
group by st1.id_auditrails
having MAX(datetime(st1.date_histo_verification));
-----
-- Views : Audit trails Import Export Archivage
-----
drop view if exists view_AT;
create view view_AT
AS
select
    t1.id_auditrails,
    t2.id_verification,
    t1.id_auditrails as "Numéro AT",
    t1.nomfichier_auditrails as "Fichier ModeOp",
    t1.commentaire_auditrails as "Commentaire AT",
    t1.infosystem_auditrails as "Info Système",
    datetime(t1.date_auditrails) as "Date AT",
    t4.lbl_operation as "Type Opération",
    t3.lbl_verification as "Statut Vérification",
    t2.date_histo_verification as "Date Statut Vérification"
from
    auditrails as t1,
    histo_verification as t2,
    enum_verification as t3,
    enum_operation as t4

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>29/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

WHERE t1.id_auditrails = t2.id_auditrails
AND t2.id_verification = t3.id_verification
AND t1.id_operation = t4.id_operation
AND t4.id_operation <> 0
group by t1.id_auditrails
having MAX(datetime(t2.date_histo_verification));

```

-----  
-- Views : Liste Utilisateurs  
-----

```

drop view if exists view_ListeUtilisateur;
CREATE VIEW view_ListeUtilisateur
AS
select
    t1.id_utilisateur,
    t2.id_droit,
    t1.id_utilisateur as "N°",
    t1.nom_utilisateur as "Login Windows",
    t2.commentaire_histo_droit as "Commentaire",
    datetime(t2.date_histo_droit) as "Date Droits",
    t3.lbl_droit as "Privilège"
from
    utilisateur as t1,
    histo_droits as t2,
    enum_droits as t3
WHERE t1.id_utilisateur = t2.id_utilisateur
AND t2.id_droit = t3.id_droit
GROUP by t1.nom_utilisateur
having MAX(datetime(t2.date_histo_droit))
order by t1.nom_utilisateur;


```

-----  
-- Views : Liste Des Signets  
-----

```

drop view if exists view_ATPrinter_Signets;
CREATE VIEW view_ATPrinter_Signets
AS
select
    t2.id_auditrails,
    t1.id_signet,
    t1.id_signet as "Signet N°",
    t1.clef_signet as "Question posée",
    t1.valeur_signet as "Réponse Utilisateur",
    t1.code_signet as "Code source du signet"
from
    signets as t1,
    impression as t2
where

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>30/156</b>

```
t1.id_impression = t2.id_impression;
```

```
-----  
-- Views : Historique Des Droits Utilisateurs  
-----
```


```
drop view if exists view_ListeDroitUtilisateur;  
CREATE VIEW view_ListeDroitUtilisateur  
AS  
select  
    t1.id_utilisateur,  
    t2.id_droit,  
    t1.commentaire_histo_droit as "Raison du changement",  
    datetime(t1.date_histo_droit) as "Date du changement",  
    t2.lbl_droit as "Droit utilisateur"  
from  
    histo_droits as t1,  
    enum_droits as t2  
where t1.id_droit = t2.id_droit  
order by datetime(t1.date_histo_droit) DESC;
```

```
-----  
-- Views : Historique Des Droits Vérification  
-----
```


```
drop view if exists view_HistoVerif;  
CREATE VIEW view_HistoVerif  
AS  
select  
    t1.id_auditrails,  
    t2.id_verification,  
    t1.commentaire_histo_verification as "Commentaire Vérificateur",  
    datetime(t1.date_histo_verification) as "Date Statut Vérification",  
    t2.lbl_verification as "Statut Vérification"  
from  
    histo_verification as t1,  
    enum_verification as t2  
where t1.id_verification = t2.id_verification  
order by datetime(t1.date_histo_verification) DESC;
```

```
-----  
-- Jeux de données : informations de base  
-----
```

```
-- DEFINITION DES DROITS INTERFACE  
INSERT INTO enum_droits VALUES (-1, 'NoAccess');  
INSERT INTO enum_droits VALUES (0, 'Guest');  
INSERT INTO enum_droits VALUES (1, 'User');  
INSERT INTO enum_droits VALUES (2, 'KeyUser');  
INSERT INTO enum_droits VALUES (3, 'UserAQ');  
INSERT INTO enum_droits VALUES (4, 'AdminAQ');  
INSERT INTO enum_droits VALUES (5, 'AdminDvlp');
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>31/156</b>

```
-- DEFINITION DES VERIFICATIONS
INSERT INTO enum_verification VALUES (0,'non vérifié');
INSERT INTO enum_verification VALUES (1,'en cours de vérification');
INSERT INTO enum_verification VALUES (2,'vérifié');
-- DEFINITION DES TYPES D'OPERATIONS
INSERT INTO enum_operation VALUES (0,'impression');
INSERT INTO enum_operation VALUES (1,'importation');
INSERT INTO enum_operation VALUES (2,'exportation');
INSERT INTO enum_operation VALUES (3,'archivage');
-- DEFINITION DES UTILISATEURS DE BASE
INSERT INTO utilisateur VALUES (null,'VDRBASETEST1');
INSERT INTO utilisateur VALUES (null,'VDRBASETEST2');
INSERT INTO utilisateur VALUES (null,'VDRBASETEST3');
INSERT INTO utilisateur VALUES (null,'VDRBASETEST4');
INSERT INTO utilisateur VALUES (null,'VDRBASETEST5');
INSERT INTO utilisateur VALUES (null,'JRIVIERE');
-- DEFINITION DES DROITS DES UTILISATEURS TEST
INSERT INTO histo_droits VALUES ("Utilisateur Test",datetime('now','localtime','+1
seconds'),1,1);
INSERT INTO histo_droits VALUES ("Utilisateur Test",datetime('now','localtime','+1
seconds'),2,2);
INSERT INTO histo_droits VALUES ("Utilisateur Test",datetime('now','localtime','+1
seconds'),3,4);
INSERT INTO histo_droits VALUES ("Utilisateur Test",datetime('now','localtime','+1
seconds'),4,3);
INSERT INTO histo_droits VALUES ("Utilisateur Test",datetime('now','localtime','+1
seconds'),5,5);
INSERT INTO histo_droits VALUES ("Développeur du
Logiciel",datetime('now','localtime','+1 seconds'),6,5);
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>32/156</b>

## 9.2 WAction

```
Imports VBTools.DialogBox
Imports System.Text
```

```
Module WAction
```

```
#Region "CONSTANTES"
```

```
'CONSTANTE BAS DE PAGE
```

```
Private Const _FOOTER_IMPRESSION = "Bon pour utilisation n°{0}"
```

```
Private Const _FOOTER_VERSION = "Date {0} {1}"
```

```
'CONSTANTE FILIGRANE A AJOUTER AUX DOCS WORD
```

```
Private Const _FILIGRANE_NOTUSE = "PAS POUR UTILISATION"
```

```
Private Const _FILIGRANE_PERIME = "Document PÉRIMÉ"
```

```
'CONSTANTE D'ACTION DESCRIPTION BOITE DE DIALOGUE
```

```
Private Const _DESC_CONSULTATION = "Sélectionner le mode op à consulter"
```

```
Private Const _DESC_IMPRESSION = "Sélectionner le mode op à imprimer"
```

```
Private Const _DESC_IMPORTATION = "Sélectionner le mode op à importer"
```

```
Private Const _DESC_IMPORTATION_DOSS_C = "Sélectionner la destination du mode op à exporter"
```

```
Private Const _DESC_EXPORTATION = "Sélectionner le mode op à exporter"
```

```
Private Const _DESC_IMPORTATION_DOSS_B = "Sélectionner la destination de la sauvegarde du  
mode opératoire à importer (cette version ne sera pas utilisable par la production)"
```

```
Private Const _DESC_IMPORTATION_DOSS_E = "Sélectionner la destination du Duplicata  
utilisable en production"
```

```
Private Const _DESC_ARCHIVAGE_DOSS_E = "Sélectionner le Duplicata crypté à archiver"
```

```
Private Const _DESC_ARCHIVAGE_DOSS_F = "Sélectionner la destination du duplicata à archiver"
```

```
'CONSTANTE DE MESSAGE ERREUR / OPERATION
```

```
Private Const _MSG_FIN_OPERATION = "-----Opération terminée-----"
```

```
Private Const _MSG_ERR_DAO = "DAO ERROR"
```

```
Private Const _MSG_ERR_WREADER = "WREADER ERROR"
```

```
Private Const _MSG_ERR_WACTION = "WACTION ERROR"
```

```
Private Const _MSG_ERR_GENERALE = "GENERAL ERROR"
```

```
Private Const _MSG_ERR_EX_1 = "erreur lors du remplacement du mot {0} par {1} en entête"
```

```
Private Const _MSG_ERR_EX_2 = "Importation annulée car le mot clef {0} n'est pas présent  
dans l'entête du document Word"
```

```
Private Const _MSG_ERR_EX_3 = "Une importation nécessitant un archivage d'une version  
antérieure doit obligatoirement avoir un mode op archivé, l'importation est donc annulée"
```

```
Private Const _MSG_ERR_EX_4 = "Mode de consultation indéterminé"
```

```
Private Const _MSG_ERR_EX_5 = "Mode d'exportation indéterminé"
```

```
Private Const _MSG_ERR_EX_6 = "Erreur lors du processus d'exportation"
```

```
Private Const _MSG_ERR_CS_1 = "Erreur lors de la consultation"
```

```
Private Const _MSG_ERR_GEN_1 = "Erreur lors de l'ouverture de la boîte de dialogue  
d'ouverture de fichier"
```

```
Private Const _MSG_ERR_IMP_1 = "Erreur lors du processus d'impression"
```

```
Private Const _MSG_ERR_AR_1 = "Erreur lors du processus d'archivage"
```

```
Private Const _MSG_ERR_IM_1 = "Erreur du processus d'archivage lors d'une importation"
```

```
Private Const _MSG_ERR_IM_2 = "Erreur lors du processus d'importation"
```

```
'CONSTANTE GENERALE DE TEXTE
```

```
Private Const _GEN_BDD_1 = "Enregistrement dans la base de donnée"
```


```
Private Const _GEN_BDD_2 = "Enregistrement AT n°{0}"
```

```
Private Const _GEN_Copy = "Fichier {0} vers {1}"
```

```
Private Const _GEN_CopyBackup = "Fichier {0} vers {1} et copie {2}"
```

```
Private Const _GEN_INFO_SIGNETS = "vous pouvez renseigner les signets"
```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>33/156</b>

```

Private Const _GEN_INFO_FILIGRANE = "Création du filigrane"
Private Const _GEN_INFO_NETTBASPAGE = "Nettoyage du bas de page"
Private Const _GEN_INFO_COPYTO = "Copie dans le dossier {0}"
Private Const _GEN_INFO_REEMPLACEMENT_ET = "Remplacement {0} par {1}"
Private Const _GEN_INFO_DEL_SOURCE = "Suppression du fichier source"
Private Const _GEN_INFO_PDF = "Création du PDF"
Private Const _GEN_INFO_AJOUT_BASPAGE = "Ajout information en bas de page"

Private Const _GEN_ARCHIVAGE_1 = "ARCHIVAGE D'UN MODE OPERATOIRE"
Private Const _GEN_ARCHIVAGE_2 = "Archivage d'une version antérieure"
Private Const _GEN_ARCHIVAGE_3 = "Archivage : annulé"
Private Const _GEN_ARCHIVAGE_4 = "Archivage : {0}"

Private Const _GEN_IMPORTATION_1 = "Importation : annulée"
Private Const _GEN_IMPORTATION_2 = "Ajout de la date d'importation"
Private Const _GEN_IMPORTATION_3 = "Importation : {0}"
Private Const _GEN_IMPORTATION_4 = "Archivage de version antérieure"
Private Const _GEN_IMPORTATION_5 = "Avez Vous une version antérieure à archiver?"
Private Const _GEN_IMPORTATION_6 = "IMPORTATION D'UN MODE OPÉRATOIRE"
Private Const _GEN_IMPORTATION_7 = "(c) {0} --> (b) {1} --> (e) {2}"

Private Const _GEN_IMPRESSION_1 = "IMPRESSION POUR PRODUCTION"
Private Const _GEN_IMPRESSION_2 = "Impression : annulée"
Private Const _GEN_IMPRESSION_3 = "Impression : {0}"
Private Const _GEN_IMPRESSION_4 = "Ouverture de la boîte de dialogue d'impression"
Private Const _GEN_IMPRESSION_5 = "Enregistrement de l'impression dans l'audit trails"
Private Const _GEN_IMPRESSION_6 = "Imprimante : {0}"
Private Const _GEN_IMPRESSION_7 = "Impression en cours sur imprimante {0}"

Private Const _GEN_CONSULTATION_1 = "CONSULTATION -> {0}"
Private Const _GEN_CONSULTATION_2 = "Consultation : annulée"
Private Const _GEN_CONSULTATION_3 = "Consultation : {0}"


Private Const _GEN_EXPORTATION_1 = "EXPORTATION -> {0}"
Private Const _GEN_EXPORTATION_2 = "Exportation : annulée"
Private Const _GEN_EXPORTATION_3 = "Exportation : {0}"

Private _Rouge As New Color(0.8, 255, 0, 0)
Private _monFont = New Font("arial", 12, FontStyle.Regular)
Private Const _Police As Single = 10
Private Const _Style As Integer = FontStyle.Bold
#End Region

Private Enum TypeModeOp As Byte
    Archive = 0
    Officiel = 1
End Enum

Public Sub doAction(ByVal monAction As Outils.Action)
    Try
        Select Case monAction
            Case Outils.Action.ConsultationOfficiel
                Consultation(CTypeModeOp.Officiel)
            Case Outils.Action.ConsultationArchive
                Consultation(CTypeModeOp.Archive)
            Case Outils.Action.Impression
                Impression()
            Case Outils.Action.Importation
                Importation()
        End Select
    Catch
    End Try
End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>34/156</b>

```

        Case Outils.Action.Archivage
            Archivage()
        Case Outils.Action.ExportationOfficiel
            Exportation(TypeModeOp.Officiel)
        Case Outils.Action.ExportationArchive
            Exportation(TypeModeOp.Archive)
        Case Else
            Throw New WActionException("--Action Indéterminée--")
        End Select
    Catch ex As DAOException
        Info("Source : " & ex.getErrSource, True)
        Info(ex.Message)
        Info(_MSG_ERR_DAO)
    Catch ex As WReaderException
        Info(ex.Message, True)
        Info("Procédure : " & ex.getProcedureOrigineErreur)
        Info("document ayant généré l'erreur: " & ex.getNomDocumentWordErreur)
        Info(_MSG_ERR_WREADER)
    Catch ex As WActionException
        Info("Source : " & ex.getErreurSource, True)
        Info(ex.Message)
        Info(_MSG_ERR_WACTION)
    Catch ex As Exception
        Info(ex.Message, True)
        Info(_MSG_ERR_GENERALE)
    Finally
        WReader.Dispose()
        Info(_MSG_FIN_OPERATION)
    End Try
End Sub

#Region "Traitement WAction"
Private Sub Exportation(ByVal TypeExport As TypeModeOp)
    Dim monDossierProd As Outils.DossierProd
    Dim FileExp As BoxOpenFile
    Dim FileC As BoxSaveFile


    Info(String.Format(_GEN_EXPORTATION_1, TypeExport.ToString), True)

    If TypeExport = TypeModeOp.Officiel Then
        monDossierProd = Outils.DossierProd.DossierE
    ElseIf TypeExport = TypeModeOp.Archive Then
        monDossierProd = Outils.DossierProd.DossierF
    Else
        Throw New WActionException(_MSG_ERR_EX_5)
    End If

    Try
        'FICHIER A EXPORTER
        FileExp = New BoxOpenFile(Configuration.GetInstance.getFullProdDir(monDossierProd))
        'le word à chercher sera crypté
        FileExp.listeExtention = {Outils.EXT_FICHIER_CRYPTER}
        'description de la boîte de dialogue
        FileExp.DialogBoxTexteDescription = _DESC_EXPORTATION
        FileExp.DialogBoxPoliceDescription = _monFont

        'affichage de la boîte de dialogue
        FileExp.ShowDialog()
        If IsNothing(FileExp.getResultatSimple) Then

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>35/156</b>

```

        Info(_GEN_EXPORTATION_2)
        Exit Sub
    End If
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_GEN_1, ex)
End Try

Try
    'EMPLACEMENT D'EXPORTATION
    FileC = New
BoxSaveFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierC),
System.IO.Path.GetFileNameWithoutExtension(FileExp.getResultatSimple))
    FileC.DialogBoxTexteDescription = _DESC_IMPORTATION_DOSS_C
    FileC.DialogBoxPoliceDescription = _monFont
    FileC.ShowDialog()
    If IsNothing(FileC.getResultatSimple) Then
        Info(_GEN_EXPORTATION_2)
        Exit Sub
    End If
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_GEN_1, ex)
End Try

Info(String.Format(_GEN_EXPORTATION_3, FileExp.getResultatSimple))
With WReader.GetMyWord
    .OpenWord(FileExp.getResultatFull, WReader.method.open)


    'remplacement en entete
    If TypeExport = TypeModeOp.Archive Then
        Info(String.Format(_GEN_INFO_REEMPLACEMENT_ET, Outils.ET_PERIME,
Outils.ET_ORIGINAL))
        If Not .ReplaceTexteEntete(Outils.ET_PERIME, Outils.ET_ORIGINAL) Then
            Throw New WActionException(String.Format(_MSG_ERR_EX_1, Outils.ET_PERIME,
Outils.ET_ORIGINAL))
        End If
    Else
        Info(String.Format(_GEN_INFO_REEMPLACEMENT_ET, Outils.ET_DUPLICATA,
Outils.ET_ORIGINAL))
        If Not .ReplaceTexteEntete(Outils.ET_DUPLICATA, Outils.ET_ORIGINAL) Then
            Throw New WActionException(String.Format(_MSG_ERR_EX_1, Outils.ET_DUPLICATA,
Outils.ET_ORIGINAL))
        End If
    End If

    Info(_GEN_INFO_AJOUT_BASPAGE)
    .AjoutTexteBasPage(String.Format(_FOOTER_VERSION, "Exportation", Now))

    Info(String.Format(_GEN_INFO_COPYTO,
Configuration.GetInstance.getSimpleProdDir(Outils.DossierProd.DossierC)))
    .CopyTo(FileC.getResultatFull)

    Info(_GEN_BDD_1)
    Dim at As New auditrails(FileExp.getResultatRelatif,
        String.Format(_GEN_EXPORTATION_3, TypeExport.ToString),
        String.Format(_GEN_Copy, FileExp.getResultatSimple,
FileC.getResultatSimple),
        Now,
        Initialisation.__User.getUserId,
        Operations.Exportation)

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>36/156</b>

```

DAOFactory.getAudittrails.dbInsert(at)
Info(String.Format(_GEN_BDD_2, at.idAudittrails))

End With
End Sub

Private Sub Archivage(Optional ByVal viaImportation As Boolean = False)
    Dim FileE As BoxOpenFile
    Dim FileF As BoxSaveFile

    If viaImportation Then
        Info(_GEN_ARCHIVAGE_2)
    Else
        Info(_GEN_ARCHIVAGE_1, True)
    End If


    Try
        'MODE OP A ARCHIVER
        FileE = New
BoxOpenFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierE))
        'le word à chercher sera crypté
        FileE.listeExtention = {Outils.EXT_FICHER_CRYPTER}
        'description de la boîte de dialogue
        FileE.DialogBoxTexteDescription = _DESC_ARCHIVAGE_DOSS_E
        FileE.DialogBoxPoliceDescription = _monFont
        'affichage de la boîte de dialogue
        FileE.ShowDialog()

        If IsNothing(FileE.getResultatSimple) Then
            If viaImportation Then
                Throw New WActionException(_MSG_ERR_EX_3)
            Else
                Info(_GEN_ARCHIVAGE_3)
            End If
        End If
    Catch ex As Exception
        Throw New WActionException(_MSG_ERR_GEN_1, ex)
    End Try

    Try
        'DOSSIER ENREGISTREMENT ARCHIVE
        FileF = New
BoxSaveFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierF),
FileE.getResultatSimple)
        FileF.DialogBoxTexteDescription = _DESC_ARCHIVAGE_DOSS_F
        FileF.DialogBoxPoliceDescription = _monFont
        FileF.ShowDialog()
    Catch ex As Exception
        Throw New WActionException(_MSG_ERR_GEN_1, ex)
    End Try

    If IsNothing(FileF.getResultatFull) Then
        If viaImportation Then
            Throw New WActionException(_MSG_ERR_EX_3)
        Else
            Info(_GEN_ARCHIVAGE_3)
        End If
    Else
        Info(String.Format(_GEN_ARCHIVAGE_4, FileE.getResultatSimple))
    End If

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>37/156</b>

```

With WReader.GetMyWord
    .OpenWord(FileE.getResultatFull, WReader.method.open)

    'remplacement mot en entête
    Info(String.Format(_GEN_INFO_REPLACEMENT_ET, Outils.ET_DUPLICATA,
Outils.ET_PERIME))
    If Not .RemplaceTexteEntete(Outils.ET_DUPLICATA, Outils.ET_PERIME) Then
        Throw New WActionException(String.Format(_MSG_ERR_EX_1, Outils.ET_DUPLICATA,
Outils.ET_PERIME))
    End If

    'ajout texte en bas de page
    Info(_GEN_INFO_AJOUT_BASPAGE)
    .AjoutTexteBasPage(String.Format(_FOOTER_VERSION,
Outils.Action.Archivage.ToString, Now()), " #")

    'copy vers dossier archive
    Info(String.Format(_GEN_INFO_COPYTO,
Configuration.GetInstance.getSimpleProdDir(Outils.DossierProd.DossierF)))
    .CopyTo(FileF.getResultatFull, True)


    Info(_GEN_BDD_1)
    Dim at As New auditrails(FileE.getResultatRelatif,
        _GEN_ARCHIVAGE_1,
        String.Format(_GEN_Copy, FileE.getResultatSimple,
FileF.getResultatSimple),
        Now,
        Initialisation.__User.getUserId,
        Operations.Archivage)
    DAOFactory.getAuditrails.dbInsert(at)
    Info(String.Format(_GEN_BDD_2, at.idAuditrails))

    'Suppression de la source
    Info(_GEN_INFO_DEL_SOURCE)
    System.IO.File.Delete(FileE.getResultatFull)
End With
End If
End Sub

''' <summary>
''' Permet l'importation d'un mode op pour utilisation en production dossier E
''' et copie dans un dossier de sauvegarde B en lecture seule
''' </summary>
''' <remarks>Le fichier source est détruit à l'issue de cette manipulation</remarks>
Private Sub Importation()
    Info(_GEN_IMPORTATION_6, True)
    Dim NeedArchivage As DialogResult
    Dim FileC As BoxOpenFile
    Dim FileB As BoxSaveFile
    Dim FileE As BoxSaveFile

    Try
        'DOSSIER IMPORTATION
        FileC = New
BoxOpenFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierC))
        'le word à chercher sera sous format Word
        FileC.listeExtention = Outils.EXT_FICHER_WORD.Split("|")
        'description de la boîte de dialogue
        FileC.ShowDialogTexteDescription = _DESC_IMPORTATION

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>38/156</b>

```

FileC.ShowDialogBoxPoliceDescription = _monFont


'affichage de la boîte de dialogue
FileC.ShowDialog()
'récupération du résultat
If IsNothing(FileC.getResultatFull) Then
    Info(_GEN_IMPORTATION_1)
    Exit Sub
End If
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_GEN_1, ex)
End Try

Try
    'DOSSIER ENREGISTREMENT SAUVEGARDE
    FileB = New
BoxSaveFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierB),
FileC.getResultatSimple)
    FileB.ShowDialogBoxTexteDescription = _DESC_IMPORTATION_DOSS_B
    FileB.ShowDialogBoxPoliceDescription = _monFont
    FileB.ShowDialog()
    If IsNothing(FileB.getResultatSimple) Then
        Info(_GEN_IMPORTATION_1)
        Exit Sub
    End If
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_GEN_1, ex)
End Try

Try
    'DOSSIER ENREGISTREMENT OFFICIEL
    FileE = New
BoxSaveFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierE),
FileC.getResultatSimple, Outils.EXT_SIMPLE_CRP, BoxSaveFile.ext.AdditionneExtention)
    FileE.ShowDialogBoxTexteDescription = _DESC_IMPORTATION_DOSS_E
    FileE.ShowDialogBoxPoliceDescription = _monFont
    FileE.ShowDialog()
    If IsNothing(FileE.getResultatSimple) Then
        Info(_GEN_IMPORTATION_1)
        Exit Sub
    End If
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_GEN_1, ex)
End Try

Try
    NeedArchivage = MessageBox.Show(_GEN_IMPORTATION_5, _GEN_IMPORTATION_4,
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question)
    If NeedArchivage = DialogResult.Cancel Then
        Info(_GEN_IMPORTATION_1)
        Exit Sub
    ElseIf NeedArchivage = DialogResult.Yes Then
        'Archivage d'une version ultérieure
        'si erreur procédure annulée
        Archivage(True)
    End If
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_IM_1, ex)
End Try

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>39/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

Info(String.Format(_GEN_IMPORTATION_3, FileC.getResultatSimple))
With WReader.GetMyWord
    .OpenWord(FileC.getResultatFull, WReader.method.open)

    If Not .RechercheEnTete(Outils.ET_ORIGINAL) Then
        Throw New WActionException(String.Format(_MSG_ERR_EX_2, Outils.ET_ORIGINAL))
    End If

    Info(_GEN_INFO_NETTBASPAGE)
    'le bas de page doit être vide
    .NettoyageTexteBasPage()

    Info(_GEN_IMPORTATION_2)
    'Ajout de la date d'importation
    .AjoutTexteBasPage(String.Format(_FOOTER_VERSION,
Outils.Action.Importation.ToString(), Now()))

    Info(String.Format(_GEN_INFO_COPYTO,
Configuration.GetInstance.getSimpleProdDir(Outils.DossierProd.DossierB)))
    .CopyTo(FileB.getResultatFull)

    Info(String.Format(_GEN_INFO_REPLACEMENT_ET, Outils.ET_ORIGINAL,
Outils.ET_DUPLICATA))
    If Not .RemplaceTexteEntete(Outils.ET_ORIGINAL, Outils.ET_DUPLICATA) Then
        Throw New WActionException(String.Format(_MSG_ERR_EX_1, Outils.ET_ORIGINAL,
Outils.ET_DUPLICATA))
    End If

    Info(String.Format(_GEN_INFO_COPYTO,
Configuration.GetInstance.getSimpleProdDir(Outils.DossierProd.DossierE)))
    'copyTo + cryptage = doc word déchargé de la mémoire via myDoc.close()
    .CopyTo(FileE.getResultatFull, True)


    Info(_GEN_BDD_1)
    Dim at As New auditrails(FileC.getResultatRelatif,
        _GEN_IMPORTATION_6,
        String.Format(_GEN_IMPORTATION_7,
            FileC.getResultatSimple,
            FileB.getResultatSimple,
            FileE.getResultatSimple),
        Now,
        Initialisation.__User.getUserId,
        Operations.Importation)

    DAOFactory.getAuditrails.dbInsert(at)
    Info(String.Format(_GEN_BDD_2, at.idAuditrails))

    Info(_GEN_INFO_DEL_SOURCE)
    System.IO.File.Delete(FileC.getResultatFull)
End With
End Sub

''' <summary>
''' Permet l'impression d'un mode op pour utilisation en production
''' </summary>
''' <remarks></remarks>
Private Sub Impression()
    Info(_GEN_IMPRESSION_1, True)

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>40/156</b>

```

Dim OpenFileDialog As BoxOpenFile

Try
    OpenFileDialog = New
BoxOpenFile(Configuration.GetInstance.getFullProdDir(Outils.DossierProd.DossierE))

    'le word à cherché sera crypté
    OpenFileDialog.listeExtention = {Outils.EXT_FICHER_CRYPTER}
    'description de la boîte de dialogue
    OpenFileDialog.DialogBoxTexteDescription = _DESC_IMPRESSION
    OpenFileDialog.DialogBoxPoliceDescription = _monFont
    'affichage de la boîte de dialogue
    OpenFileDialog.ShowDialog()
Catch ex As Exception
    Throw New WActionException(_MSG_ERR_GEN_1, ex)
End Try

If IsNothing(OpenFileDialog.getResultatSimple) Then
    Info(_GEN_IMPRESSION_2)
Else
    Info(String.Format(_GEN_IMPRESSION_3, OpenFileDialog.getResultatSimple))
    With WReader.GetMyWord
        Info(_GEN_INFO_SIGNETS)
        .OpenWord(OpenFileDialog.getResultatFull, WReader.method.add)

        Dim WPrinter As New VueImpression(.getFields,
Configuration.GetInstance.getListePhraseAT, .getPages)
        Info(_GEN_IMPRESSION_4)
        WPrinter.ShowDialog()

        If WPrinter.isValidForPrinting Then
            Info(_GEN_IMPRESSION_5)


            Dim PT_at As New auditrails(OpenFileDialog.getRepertoireBaseRelatif,
WPrinter.getAuditTrails,
OpenFileDialog.getResultatSimple,
Now(),
Initialisation.__User.getUserId,
Operations.Impression)
            Dim PT_imp As New Impression(WReader.GetMyWord.getLot,
WPrinter.getNomPrinter,
getPageAsString(WPrinter.getPageAImprimer))
            Dim PT_signet As New List(Of Signets)
            For Each element As Signets In .getFields
                PT_signet.Add(New Signets(element.getClefSignet,
element.getValeurSignet, element.getCodeSignet))
            Next
            DAOFactory.getATPrinter.dbInsertATPrinter(PT_at, PT_imp, PT_signet)

            .AjoutTexteBasPage(String.Format(_FOOTER_IMPRESSION, PT_at.idAuditrails), "
#")

            Info(String.Format(_GEN_IMPRESSION_7, WPrinter.getNomPrinter))
            .PrintDoc(WPrinter.getPageAImprimer)
        Else
            Info(_GEN_IMPRESSION_2)
        End If
    End With
End If

```



	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>41/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

End Sub

```
''' <summary>
''' Permet la consultation d'un mode op
''' </summary>
''' <param name="TConsultation">type de consultation</param>
''' <remarks></remarks>
```

```
Private Sub Consultation(ByVal TConsultation As TypeModeOp)
    Info(String.Format(_GEN_CONSULTATION_1, TConsultation.ToString), True)
    Dim monDossierProd As Outils.DossierProd
    Dim monFiligrane As String
    Select Case TConsultation
        Case TypeModeOp.Archive
            monDossierProd = Outils.DossierProd.DossierF
            monFiligrane = _FILIGRANE_PERIME
        Case TypeModeOp.Officiel
            monDossierProd = Outils.DossierProd.DossierE
            monFiligrane = _FILIGRANE_NOTUSE
        Case Else
            Throw New WActionException(_MSG_ERR_EX_4)
    End Select
```

```
    Dim OpenFileDialog As New
    BoxOpenFile(Configuration.GetInstance.getFullProdDir(monDossierProd))
```

```
    'le word à chercher sera crypté
    OpenFileDialog.listeExtention = {Outils.EXT_FICHER_CRYPTER}
    'description de la boîte de dialogue
    OpenFileDialog.ShowDialogTexteDescription = _DESC_CONSULTATION
    OpenFileDialog.ShowDialogPoliceDescription = _monFont
    'affichage de la boîte de dialogue
    OpenFileDialog.ShowDialog()
```


```
    If IsNothing(OpenFileDialog.getResultatSimple) Then
        Info(_GEN_CONSULTATION_2)
    Else
        Info(String.Format(_GEN_CONSULTATION_3, OpenFileDialog.getResultatSimple))
        With WReader.GetMyWord
            Info(_GEN_INFO_SIGNETS)
            .OpenWord(OpenFileDialog.getResultatFull, WReader.method.add)
            Info(_GEN_INFO_FILIGRANE)
            .Filigrane(monFiligrane, _Rouge)
            Info(_GEN_INFO_PDF)
            .ExportAsPDF()
        End With
    End If
```

End Sub

#End Region

```
'appel de la zone d'affichage de la vue principale
Private Sub Info(ByVal monTexte As String, Optional ByVal Serapation As Boolean = False)
    vuePrincipale.getVP.Info(monTexte, Serapation)
End Sub
```

```
Private Function getPagesAsString(ByVal liste As List(Of Integer)) As String
    If liste.Count = 0 Then
        Return "-"
    Else
        Dim txt As New System.Text.StringBuilder
```


	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>42/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

With txt
    For Each _Int As Integer In liste
        .Append(_Int).Append(",")
    Next
End With
Return txt.ToString.Remove(txt.ToString.Length - 1)
End If
End Function

End Module

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>43/156</b>
--	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

### 9.3 WActionException

```

Public Class WActionException
    Inherits Exception

    Private Const _NOSOURCE = "pas d'erreur source"
    Private Const _SOURCEWREADER = "WReader est la source de l'erreur : {0}"
    Private _ErreurSource As String

    Public ReadOnly Property getErreurSource As String
        Get
            Return _ErreurSource
        End Get
    End Property


    Sub New(ByVal errmsg As String)
        MyBase.New(errmsg)
        _ErreurSource = _NOSOURCE
    End Sub

    ''' <summary>
    ''' Permet l'identification précise d'une erreur généré par WAction
    ''' </summary>
    ''' <param name="ex">Exception d'origine</param>
    ''' <param name="errmsg">Message d'erreur</param>
    ''' <remarks></remarks>
    Sub New(ByVal errmsg As String, ByVal ex As Exception)
        MyBase.New(errmsg)
        If IsNothing(ex) Then
            _ErreurSource = _NOSOURCE
        Else
            _ErreurSource = ex.Message
        End If
    End Sub

    Sub New(ByVal errmsg As String, ByVal ex As WReaderException)
        MyBase.New(errmsg)
        If IsNothing(ex) Then
            _ErreurSource = _NOSOURCE
        Else
            _ErreurSource = String.Format(_SOURCEWREADER, ex.Message)
        End If
    End Sub

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>44/156</b>

## 9.4 WReader

```
Imports Word = Microsoft.Office.Interop.Word
Imports System.Reflection
Imports System.IO
Imports System.Security.Cryptography
Imports System.Text

Public Class WReader
    Private Shared _Instance As WReader = Nothing
    Private Shared ReadOnly mylock As New Object()

    Private Shared _myWord As Word.Application
    Private Shared _myDoc As Word.Document
    Private Shared _FullName As String = ""

    Private _ListeSignet As New List(Of Signets)
    Private Const _ERR_NOINSTANCE As String = "Il n'y a aucun document word ouvert"
    Private Const _sKey As String = "f4ty52uG"

    ''' <summary>
    ''' Détermine la méthode d'ouverture du document Word
    ''' </summary>
    ''' <remarks></remarks>
    Public Enum method
        ''' <summary>
        ''' utilise la méthode Documents.Open sans ouverture des signets
        ''' </summary>
        ''' <remarks></remarks>
        open = 0
        ''' <summary>
        ''' utilise la méthode Documents.Add avec ouverture des signets
        ''' </summary>
        ''' <remarks></remarks>
        add = 1
    End Enum

#Region "Property"
    Public Shared ReadOnly Property getDocName() As String
        Get
            Return Path.GetFileName(_FullName)
        End Get
    End Property

    Private Property visible As Boolean
        Get
            Return _myWord.Visible
        End Get
        Set(ByVal value As Boolean)
            _myWord.Visible = value
        End Set
    End Property

    Public ReadOnly Property getInfos() As String
        Get
            Dim infos As String = "Information : " & getDocName & vbNewLine
            infos += "Nb Section : " & _myDoc.Sections.Count & vbNewLine
        End Get
    End Property
End Class
```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>45/156</b>

```

        For Each sec As Word.Section In _myDoc.Sections
            infos += "Section (" & sec.Index & ") Nb Header : " & sec.Headers.Count & " Nb
Fields : " & sec.Range.Fields.Count & vbNewLine
            infos += "Section (" & sec.Index & ") Nb Footer : " & sec.Footers.Count &
vbNewLine
        Next

        Return infos
    End Get
End Property


''' <summary>
''' Détermine si un document word est ouvert
''' </summary>
''' <value></value>
''' <returns>True si document ouvert</returns>
''' <remarks></remarks>
Public ReadOnly Property isOpen As Boolean
    Get
        Return Not NoInstance()
    End Get
End Property

''' <summary>
''' Retourne le nombre de page du document word
''' </summary>
''' <value></value>
''' <returns>Object renvoyé par la méthode
Doc.Range.Information(wdNumberOfPagesInDocument)</returns>
''' <remarks></remarks>
Public ReadOnly Property getPages() As Object
    Get
        If NoInstance() Then
            Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
        End If
        Try
            'variable vba
            Dim wdNumberOfPagesInDocument As Integer = 4
            Dim numPages As Integer = -1
            'pour avoir le bon nombre de page avant impression
            'nécessite d'avoir appelé la méthode extractionFields()
            'pour éviter d'avoir de nouveau les boites de dialogues
            _myDoc.PrintPreview()
            numPages = _myDoc.Range.Information(wdNumberOfPagesInDocument)
            _myDoc.ClosePrintPreview()

            Return numPages
        Catch ex As Exception
            Throw New WReaderException("Erreur lors de la détermination du nombre de page du
document word", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
        End Try
    End Get
End Property

''' <summary>
''' Liste des signets renseignés par l'utilisateur
''' </summary>
Public ReadOnly Property getFields As List(Of Signets)

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>46/156</b>

```

    Get
        If NoInstance() Then
            Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
        End If
        Try
            Return _ListeSignet
        Catch ex As Exception
            Throw New WReaderException("Erreur lors de la récupération de la liste des
signets", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
        End Try
    End Get
End Property

Public ReadOnly Property getLot As String
    Get
        Return extractLotSignet()
    End Get
End Property


#End Region

#Region "Constructeur / Ouverture Doc Word"
''' <summary>
''' Méthode d'ouverture des modes opératoires
''' </summary>
''' <param name="fichier">Chemin fichier Word (ou crp)</param>
''' <remarks></remarks>
Public Sub OpenWord(ByVal fichier As String, ByVal methode As method)
    Try
        'ferme le word en cours
        If Not NoInstance() Then
            'Fermeture ancien word
            Close()
        End If
        Catch ex As Exception
            Throw New WReaderException("Erreur lors du processus de fermeture de l'ancien Word",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
        End Try

        'vérification des données reçues
        If IsNothing(fichier) Then
            Throw New WReaderException("il n'y a pas de nom de fichier à ouvrir",
System.Reflection.MethodBase.GetCurrentMethod().Name)
        ElseIf Not File.Exists(fichier) Then
            Throw New WReaderException("Le fichier demandé n'existe pas : " & vbNewLine &
fichier, System.Reflection.MethodBase.GetCurrentMethod().Name)
        Else
            _FullName = fichier
        End If

        'détermine si word crypté ou non puis ouverture
        If Path.GetExtension(fichier).ToLower = Outils.EXT_SIMPLE_CRP Then
            Try
                'décryptage
                Dim monCRP As String = Decrypter(fichier)
                'Ouverture du word décrypté
                ouverture(monCRP, methode)
            Catch ex As Exception

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>47/156</b>

```

        Throw New WReaderException("Erreur lors du processus de décryptage",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
Else
    Try
        'Ouverture d'un doc word
        ouverture(fichier, methode)
    Catch ex As Exception
        Throw New WReaderException("Erreur lors de l'ouverture du document Word",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
End If

Try
    'déprotection du document
    If _myDoc.ProtectionType <> Word.WdProtectionType.wdNoProtection Then
        If Not Unlocked(Outils.LISTE_MDP_PRODUCTION.Split("|")) Then
            Throw New WReaderException("Le mot de passe du document Word est inconnu",
"Unlocked")
        End If
    End If
Catch ex As Exception
    Throw New WReaderException("Erreur lors du traitement post ouverture du document
Word", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
End Try

Try
    If _myWord.Visible = True Then
        'le doc est non visible
        _myWord.Visible = False
    End If
Catch ex As Exception
    Throw New WReaderException("Erreur lors du masquage du Word",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
End Try


Try
    'récupération des infos des signets
    extractionFields()
Catch ex As Exception
    Throw New WReaderException("Erreur lors du processus d'extraction des signets",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
End Try

End Sub

Private Sub ouverture(ByVal fichier As String, ByVal methode As method)
    If methode = method.add Then
        _myDoc = _myWord.Documents.Add(fichier, True, True, False)
    Else
        _myDoc = _myWord.Documents.Open(fichier)
        _myWord.Visible = False
    End If
End Sub

''' <summary>
''' Récupération du fichier Word ouvert grace à la méthode openWord()
''' </summary>

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>48/156</b>

```

''' <returns></returns>
''' <remarks></remarks>
Public Shared Function GetMyWord() As WReader
    SyncLock (mylock)
        Try
            If IsNothing(_Instance) Then
                _Instance = New WReader()
            End If
            Return _Instance
        Catch ex As Exception
            Throw New WReaderException("Erreur lors de la récupération de l'instance de la
classe WReader", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
        End Try
    End SyncLock
End Function
Private Sub New()
    'Ouverture Word
    Try
        _myWord = New Word.Application
    Catch ex As Exception
        Throw New WReaderException("Erreur lors de la création de l'instance de
l'application Word", System.Reflection.MethodBase.GetCurrentMethod().Name)
    End Try
End Sub
#End Region

#Region "Destructeur"
''' <summary>
''' Ferme le document Word actif
''' </summary>
''' <remarks></remarks>
Private Shared Sub Close()
    On Error Resume Next
    _myDoc.Close(False)
    Err.Clear()
End Sub
''' <summary>
''' libère les ressources de la classe
''' </summary>
''' <remarks></remarks>
Public Shared Sub Dispose()
    On Error Resume Next
    Close()
    _myWord.Quit()
    Err.Clear()
    _Instance = Nothing
End Sub
#End Region


#Region "Methode Externe"
Public Sub CopyTo(ByVal destination As String, Optional ByVal crypter As Boolean = False)
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

    If IsNothing(destination) Then Exit Sub

    Try

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>49/156</b>

```

        'déverrouille les signets avant enregistrement
    extractionFields(False)
    If crypter Then
        Cryptage(destination)
    Else
        _myDoc.SaveAs2(destination)
    End If
    Catch ex As Exception
        Throw New WReaderException("Erreur lors de la création d'une copie (" & destination
& ")", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
End Sub

''' <summary>
''' Remplace un texte par un autre dans l'entete du document
''' </summary>
''' <param name="TexteAREmplacer"></param>
''' <param name="RemplacerPar"></param>
''' <returns></returns>
''' <remarks></remarks>
Public Function RemplaceTexteEntete(ByVal TexteAREmplacer As String, ByVal RemplacerPar As
String) As Boolean
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If


    If IsNothing(TexteAREmplacer) Or IsNothing(RemplacerPar) Then Return False

    Dim operation As Boolean = False
    On Error GoTo no
        _myWord.ActiveWindow.ActivePane.View.SeekView = 1
        operation = _myWord.Selection.Find.Execute(FindText:=TexteAREmplacer, Forward:=True,
ReplaceWith:=RemplacerPar)
        _myWord.ActiveWindow.ActivePane.View.SeekView = 0
no:
    Return operation
End Function

''' <summary>
''' Recherche la présence d'un mot clef en entete du document
''' </summary>
''' <param name="motClef">mot à chercher</param>
''' <returns>True si mot présent</returns>
''' <remarks>Meme technique de recherche que la méthode de remplacement</remarks>
Public Function RechercheEnTete(ByVal motClef As String) As Boolean
    Dim rep As Boolean = False
    _myWord.ActiveWindow.ActivePane.View.SeekView = 1
    rep = _myWord.Selection.Find.Execute(motClef)
    _myWord.ActiveWindow.ActivePane.View.SeekView = 0
    Return rep
End Function

''' <summary>
''' Conversion du word en PDF
''' </summary>
''' <remarks></remarks>
Public Sub ExportAsPDF()
    If NoInstance() Then

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>50/156</b>

```

        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If
    Try
        Dim PDFFileName As String = Path.GetTempFileName & ".pdf"
        _myDoc.ExportAsFixedFormat(PDFFileName, Word.WdExportFormat.wdExportFormatPDF, True)
    Catch ex As Exception
        Throw New WReaderException("erreur lors de la création d'un PDF à partir du Word",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try

End Sub

''' <summary>
''' Relance les boites de dialogue de renseignement des signets
''' </summary>
''' <returns>Vrai si pas d'erreur</returns>
''' <remarks></remarks>
Public Function ActiveFields() As Boolean
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

    Dim retour As Boolean = True


    Try
        _myDoc.Range.Fields.Locked = False
        'Update() retourne 0 si ok et 1 si erreur
        retour = retour And Not _myDoc.Range.Fields.Update()
        _myDoc.Range.Fields.Locked = True

        _myDoc.Sections(1).Headers(1).Range.Fields.Locked = False
        retour = retour And Not _myDoc.Sections(1).Headers(1).Range.Fields.Update()
        _myDoc.Sections(1).Headers(1).Range.Fields.Locked = True
    Catch ex As Exception
        Throw New WReaderException("erreur lors de la réactivation manuelle des signets via
boites de dialogues word", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    Finally
        visible = False
        extractionFields()
    End Try

    Return retour
End Function

''' <summary>
''' Imprime un document Word selon un interval de pages
''' </summary>
''' <param name="FromPage">1ere page de l'interval à imprimer</param>
''' <param name="ToPage">dernière page de l'interval à imprimer</param>
''' <remarks></remarks>
'''
Public Overloads Sub PrintDoc(ByVal FromPage As Integer, ByVal ToPage As Integer)
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>51/156</b>

```

Try
    If FromPage < 0 Or ToPage < 0 Then Exit Sub
    If FromPage > ToPage Then Exit Sub
Catch ex As Exception
    Throw New WReaderException("erreur avec les numéros de page envoyés à l'impression",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
End Try

Try
    For i = FromPage To ToPage

        Next
    Catch ex As Exception
        Throw New WReaderException("erreur lors de l'impression du document (int, int)",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
End Sub


''' <summary>
''' Imprime une page unique
''' </summary>
''' <param name="Page">Numéro de la page à imprimer</param>
''' <remarks></remarks>
Public Overloads Sub PrintDoc(ByVal Page As Integer)
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

    Try
        'impression Recto seul
        _myDoc.PrintOut(Background:=False, Range:=4, Pages:=CStr(Page))
    Catch ex As Exception
        Throw New WReaderException("erreur lors de l'impression du document (int)",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
End Sub

''' <summary>
''' Imprime à partir d'une liste de numéro de page
''' </summary>
''' <param name="Pages">Liste de numéro de pages</param>
''' <remarks></remarks>
Public Overloads Sub PrintDoc(ByVal Pages As List(Of Integer))
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

    Try
        For Each Page As Integer In Pages
            PrintDoc(Page)
        Next
    Catch ex As Exception
        Throw New WReaderException("erreur lors de l'impression du document (List of int)",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>52/156</b>

```

''' <summary>
''' Ajoute un filigrane dans le doc word
''' </summary>
''' <param name="TexteFiligrane">Texte du filigrane</param>
''' <param name="Couleur">Objet de type Transparence/Couleur </param>
''' <remarks></remarks>
Public Sub Filigrane(ByVal TexteFiligrane As String, ByVal Couleur As Color)
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

    If IsNothing(TexteFiligrane) Or IsNothing(Couleur) Then Exit Sub

    Try
        For Each section As Word.Section In _myDoc.Sections
            section.Range.Select()


            _myDoc.ActiveWindow.ActivePane.View.SeekView =
Word.WdSeekView.wdSeekCurrentPageHeader
            _myWord.Selection.HeaderFooter.Shapes.AddTextEffect(1, TexteFiligrane, "Times
New Roman", 1, False, False, 0, 0).Select()

            With _myWord.Selection.ShapeRange
                .Name = "WA_" & section.Range.Start & section.Range.ID & "." & section.Index
                .TextEffect.NormalizedHeight = False
                .Line.Visible = False
                .Fill.Visible = True
                .Fill.Solid()
                .Fill.ForeColor.RGB = RGB(Couleur.getRouge, Couleur.getVert,
Couleur.getBleu)

                .Fill.Transparency = Couleur.getAlpha
                .Rotation = 330
                .LockAspectRatio = True
                .Height = CentimetersToPoints(4.46)
                .Width = CentimetersToPoints(22.32)
                .WrapFormat.AllowOverlap = True
                .WrapFormat.Side = 3
                .WrapFormat.Type = 3
                .RelativeHorizontalPosition = 0
                .RelativeVerticalPosition = 0
                .Top = -999995
                .Left = -999995
            End With
            _myDoc.ActiveWindow.ActivePane.View.SeekView = 0
        Next
        Catch ex As Exception
            Throw New WReaderException("erreur lors de l'ajout d'un filigrane au document word",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
        End Try
    End Sub

''' <summary>
''' Nettoyage du bas de page du document word
''' </summary>
''' <remarks></remarks>
Public Sub NettoyageTexteBasPage()
    If NoInstance() Then

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>53/156</b>

```

        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If

    Try
        With _myDoc.PageSetup
            .FirstPageTray = 0
            .OtherPagesTray = 0
            .OddAndEvenPagesHeaderFooter = False
            .DifferentFirstPageHeaderFooter = False
            .TwoPagesOnOne = False
        End With

        For Each sec As Word.Section In _myDoc.Sections
            For Each foot As Word.HeaderFooter In sec.Footers
                With foot.Range
                    .Delete()
                End With
            Next
        Next
    Catch ex As Exception
        Throw New WReaderException("erreur lors du traitement du bas de page du document
word", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try

End Sub


''' <summary>
''' Ajout d'un texte en bas de page du document word
''' </summary>
''' <param name="Texte">texte à ajouter</param>
''' <param name="Separtexte">String à ajouter entre l'ancien et le nouveau texte</param>
''' <remarks>Utiliser la méthode de nettoyage du bas de page avant d'utiliser cette
méthode</remarks>
Public Sub AjoutTexteBasPage(ByVal Texte As String, Optional ByVal Separtexte As String =
"")
    If NoInstance() Then
        Throw New WReaderException(_ERR_NOINSTANCE,
System.Reflection.MethodBase.GetCurrentMethod().Name)
    End If
    If IsNothing(Texte) Then Exit Sub
    Try
        Dim SecS, SecE As Integer

        'on récupère la taille du 1er bas de page
        With _myDoc.Sections(1).Footers(1).Range
            SecS = .Start
            SecE = .End
        End With

        For Each sec As Word.Section In _myDoc.Sections
            If sec.Footers.Count >= 1 Then
                With sec.Footers(1).Range
                    'Si ce bas de page est différent du 1er, c'est que le texte à déjà été
ajouté

                    If SecS = .Start And SecE = .End Then
                        .Text = Replace(Replace(.Text & Separtexte & Texte, Chr(10), ""),
Chr(13), "") 'remplace .Text & vbCrLf & Texte
                        .Font.Size = 8.0
                    End If
                End With
            End If
        Next
    End Try
End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>54/156</b>

```


        End If
    End With
End If
Next
Catch ex As Exception
    Throw New WReaderException("erreur lors de l'ajout d'information en bas de page",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
End Try
End Sub
#End Region

#Region "METHODE INTERNE"
''' <summary>
''' Méthode de conversion centimetre en point
''' </summary>
''' <param name="value">valeur en centimetre</param>
''' <returns>valeur en point</returns>
''' <remarks></remarks>
Private Function CentimetersToPoints(ByRef value As Long) As Long
    Return value * 28.3464574
End Function

''' <summary>
''' Méthode de déprotection en utilisant une liste de mot clef
''' généralement ce sont les password de production
''' </summary>
''' <param name="ListeMDP">Liste des mots de passes potentiellement utilisés</param>
''' <remarks></remarks>
Private Function Unlocked(ByVal ParamArray ListeMDP() As String) As Boolean
    'traitement fichier inconnu
    On Error Resume Next
    For Each mdp As String In ListeMDP
        _myDoc.Unprotect(mdp)
        If Err.Number <> 0 Then
            Err.Clear()
        Else
            Return True
        End If
    Next
    Return False
End Function

''' <summary>
''' Récupère les signets renseignés par l'utilisateur
''' Généralement renseigné à l'ouverture du document grace à la méthode _myWord.document.add
''' </summary>
''' <param name="Locked">Définie si les signets sont locké ou non à l'issue de la
récupération</param>
''' <remarks></remarks>
Private Sub extractionFields(Optional ByRef Locked As Boolean = True)
    Try
        _ListeSignet.Clear()
        'récupération des signets des anciens mode opératoire
        For Each element As Word.Field In _myDoc.Sections(1).Headers(1).Range.Fields
            If element.Type = Word.WdFieldType.wdFieldFillIn Then
                Dim sgt As New Signets(extractClefSignet(element.Code.Text),
element.Result.Text, element.Code.Text)
                If Not _ListeSignet.Contains(sgt) Then
                    _ListeSignet.Add(sgt)
                End If
            End If
        Next
    Catch ex As Exception
        'gestion des erreurs
    End Try
End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>55/156</b>

```

        End If
        element.Locked = True
    End If
Next

'recupération dans le corps du document
For Each element As Word.Field In _myDoc.Range.Fields
    If element.Type = Word.WdFieldType.wdFieldSet Then
        Dim sgt As New Signets(extractClefSignet(element.Code.Text),
element.Result.Text, element.Code.Text)
        If Not _ListeSignet.Contains(sgt) Then
            _ListeSignet.Add(sgt)
        End If
        element.Locked = Locked 'pour bloquer l'ouverture type popup intempestif
    End If
Next


For Each sec As Word.Section In _myDoc.Sections
    For Each element As Word.Field In sec.Range.Fields
        If element.Type = Word.WdFieldType.wdFieldSet Or element.Type =
Word.WdFieldType.wdFieldFillIn Then
            Dim sgt As New Signets(extractClefSignet(element.Code.Text),
element.Result.Text, element.Code.Text)
            If Not _ListeSignet.Contains(sgt) Then
                _ListeSignet.Add(sgt)
            End If
            element.Locked = Locked 'pour bloquer l'ouverture type popup intempestif
        End If
    Next
Next
Catch ex As Exception
    Throw New WReaderException("erreur lors de la récupération de la valeur des signets
renseigné par l'utilisateur", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
End Try
End Sub

Private Function extractClefSignet(ByVal codeSignet As String) As String
    Dim tab() = codeSignet.Split(Chr(34))
    If tab.Count = 3 Then
        Return tab(1).Trim({Chr(34), " "})
    Else
        Return "-"
    End If
End Function

Private Function extractLotSignet() As String
    For Each element As Signets In _ListeSignet
        If LCase(element.getCodeSignet) Like "*" & LCase("Lot") & "*" Then
            Return element.getValeurSignet
        End If
    Next
    Return "-"
End Function

''' <summary>
''' Crypte un fichier word en CRP
''' </summary>
''' <param name="destinationCRP">nom du fichier de destination</param>
''' <remarks>l'instance du document est perdu à l'issue de cette méthode</remarks>

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>56/156</b>

```

Private Sub Cryptage(ByRef destinationCRP As String)
    Dim fichierInput As String = Path.GetTempFileName
    Try
        _myDoc.SaveAs2(fichierInput)
        'close pour permettre l'accès à fsInput
        _myDoc.Close()
    Catch ex As Exception
        Throw New WReaderException("erreur lors de la création du fichier temporaire lors de
l'opération de cryptage", System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try

    Dim fsInput As FileStream = New FileStream(fichierInput, FileMode.Open, FileAccess.Read)
    Dim fsEncrypted As New FileStream(destinationCRP, FileMode.Create, FileAccess.Write)
    Dim DES As New DESCryptoServiceProvider()
    DES.Key = ASCIIEncoding.ASCII.GetBytes(_sKey)
    DES.IV = ASCIIEncoding.ASCII.GetBytes(_sKey)
    Dim desencrypt As ICryptoTransform = DES.CreateEncryptor()
    Dim cryptostream As CryptoStream = New CryptoStream(fsEncrypted, desencrypt,
CryptoStreamMode.Write)
    Try
        'Lit le texte du fichier source dans le tableau d'octets
        Dim bytearrayinput(fsInput.Length - 1) As Byte
        fsInput.Read(bytearrayinput, 0, bytearrayinput.Length)


        'écrit le fichier crypté à l'aide de DES
        cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length)
    Catch ex As Exception
        Throw New WReaderException("erreur lors du cryptage du document word temporaire",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    Finally
        Cryptostream.Close()
        fsInput.Dispose()
        'suppression du fichier temporaire
        File.Delete(fichierInput)
    End Try
End Sub

''' <summary>
''' Décrypte un fichier word CRP dans un dossier temporaire (GetTempPath)
''' </summary>
''' <param name="WordCrypter">Le chemin du fichiers crypté</param>
''' <returns>Le chemin du fichier décrypté</returns>
''' <remarks></remarks>
Private Function Decrypter(ByVal WordCrypter As String) As String

    Dim Destination As String = Path.GetTempPath &
Path.GetFileNameWithoutExtension(WordCrypter)
    Dim DES As New Security.Cryptography.DESCryptoServiceProvider()
    DES.Key() = ASCIIEncoding.ASCII.GetBytes(_sKey)
    DES.IV = ASCIIEncoding.ASCII.GetBytes(_sKey)
    Dim fsread As New FileStream(WordCrypter, FileMode.Open, FileAccess.Read)
    Dim desdecrypt As Security.Cryptography.ICryptoTransform = DES.CreateDecryptor()
    Dim cryptostreamDecr As New Security.Cryptography.CryptoStream(fsread, desdecrypt,
Security.Cryptography.CryptoStreamMode.Read)
    Dim fsDecrypted As New FileStream(Destination, FileMode.Create, FileAccess.Write)
    Dim buffer As New List(Of Byte)
    Dim byt As Integer
    Try
        Do

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>57/156</b>

```

        byt = cryptostreamDecr.ReadByte()
        If byt = -1 Then
            Exit Do
        Else
            buffer.Add(byt)
        End If
    Loop
Catch ex As Exception
    Throw New WReaderException("Erreur lors du processus de décryptage",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
Finally
    fsDecrypted.Write(buffer.ToArray, 0, buffer.Count)
    fsDecrypted.Flush()
    fsDecrypted.Close()
    fsread.Close()

End Try


Return Destination

End Function

''' <summary>
''' Vérifie qu'un fichier word est bien ouvert
''' </summary>
''' <returns>true si aucune instance sinon false</returns>
''' <remarks></remarks>
Private Function NoInstance() As Boolean
    Try
        Return _myWord.Documents.Count <= 0
    Catch ex As Exception
        Throw New WReaderException("erreur lors du controle d'instance",
System.Reflection.MethodBase.GetCurrentMethod().Name, ex)
    End Try
End Function
#End Region

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>58/156</b>

## 9.5 WReaderException

```
Public Class WReaderException
    Inherits Exception
```

```
    Private _errModule As String
    Private _DocumentName As String = Nothing
    Private Const _ERR_NOWORDDOC = "nom du document word non déterminé"
```


```
    Public ReadOnly Property getNomDocumentWordErreur() As String
        Get
            Return If(IsNothing(_DocumentName), _ERR_NOWORDDOC, _DocumentName)
        End Get
    End Property
```

```
    Public ReadOnly Property getProcedureOrigineErreur() As String
        Get
            Return _errModule
        End Get
    End Property
```

```
''' <summary>
''' Permet l'identification précise d'une erreur généré par WReader
''' </summary>
''' <param name="ex">Exception d'origine</param>
''' <param name="errmsg">Message d'erreur</param>
''' <param name="errModule">System.Reflection.MethodBase.GetCurrentMethod().Name</param>
''' <remarks></remarks>
```

```
Sub New(ByVal errmsg As String, ByVal errModule As String, Optional ByVal ex As Exception =
Nothing)
    MyBase.New(If(IsNothing(ex), errmsg, ex.Message))
    _errModule = errModule
    _DocumentName = WReader.getDocName
    WReader.Dispose()
End Sub
```

```
End Class
```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>59/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

## 9.6 Color

Public Class Color

```
Private _Alpha As Double
Private _Rouge As Byte
Private _Vert As Byte
Private _Bleu As Byte
```

```
Public ReadOnly Property getAlpha() As Double
    Get
```

```
        Return _Alpha
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property getRouge() As Byte
```

```
    Get
```

```
        Return _Rouge
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property getVert() As Byte
```

```
    Get
```

```
        Return _Vert
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property getBleu() As Byte
```

```
    Get
```

```
        Return _Bleu
```

```
    End Get
```

```
End Property
```

```
Sub New(Optional ByVal Alpha As Double = 1.0, Optional ByVal Rouge As Byte = 0, Optional
ByVal Vert As Byte = 0, Optional ByVal Bleu As Byte = 0)
```

```
    If Alpha > 1.0 Then
```

```
        _Alpha = 1.0
```

```
    ElseIf Alpha < 0.0 Then
```

```
        _Alpha = 0.0
```

```
    Else
```

```
        _Alpha = Alpha
```

```
    End If
```


```
    _Rouge = Rouge
```

```
    _Vert = Vert
```

```
    _Bleu = Bleu
```

```
End Sub
```

End Class

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>60/156</b>

## 9.7 Configuration

Imports System.IO

```

''' <summary>
''' Lecture et enregistrement des variables du fichier ini
''' Variables d'environnements de l'interface
''' </summary>
''' <remarks></remarks>
Public Class Configuration

    'instance du singleton
    Private Shared _Instance As Configuration = Nothing

    'nom du fichier ini à la racine de l'interface
    Private _MonFichier As String

    'liste des combinaisons clef / valeur récupéré du fichier ini
    Private _KeyValueList As Hashtable


    'Liste des phrases d'audit trails
    Private _maListePAT As New List(Of String)

    'mot clef list
    Private Const _PAT = "PAT"

    'Caractère spéciaux du fichier ini
    Private Const _COMMENTAIRE As String = ";"
    Private Const _REGION As String = "["
    Private Const _ATTRIBVAL As String = "="

    ''' <summary>
    ''' Répertoire de l'application en cours
    ''' </summary>
    Public ReadOnly Property getBaseDir() As String
        Get
            Return Directory.GetCurrentDirectory
        End Get
    End Property
    ''' <summary>
    ''' Répertoire de l'application en cours avec répertoire de base
    ''' </summary>
    Public ReadOnly Property getWorkDir() As String
        Get
            Dim WorkDir As New System.Text.StringBuilder(Directory.GetCurrentDirectory)
            With WorkDir
                .Append("\")
                .Append(Me.GetValueFromKey(Outils.INI_KEY_REPBASE))
            End With
            Return WorkDir.ToString
        End Get
    End Property
    Public ReadOnly Property getFullProdDir(ByVal dossierProd As Outils.DossierProd) As String
        Get
            Dim FullDir As New System.Text.StringBuilder(Directory.GetCurrentDirectory)
            With FullDir
                .Append("\")
                .Append(Me.GetValueFromKey(Outils.INI_KEY_REPBASE))

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>61/156</b>

```

        .Append("\")
        .Append(Me.GetValueFromKey(dossierProd.ToString))
    End With
    Return FullDir.ToString
End Get
End Property
Public ReadOnly Property getSimpleProdDir(ByVal dossierProd As Outils.DossierProd) As String
    Get
        Return Me.GetValueFromKey(dossierProd.ToString)
    End Get
End Property
Public ReadOnly Property getListePhraseAT() As List(Of String)
    Get
        Return _maListePAT
    End Get
End Property

Private Sub New()
    'récupère le nom du fichier décrit dans les constantes
    _MonFichier = Outils.NOM_FICHIER_INI
    _KeyValueList = getData()
End Sub

Public Shared Function getInstance() As Configuration

    If _Instance Is Nothing Then
        _Instance = New Configuration()
    End If


    Return _Instance

End Function

''' <summary>
''' Récupère les combinaisons clef / valeur du fichier ini
''' </summary>
''' <returns>une liste de combinaison clef / valeur</returns>
''' <remarks></remarks>
Private Function getData() As Hashtable
    Dim listeArgs As New Hashtable

    If File.Exists(_MonFichier) Then
        For Each Ligne As String In File.ReadAllLines(_MonFichier)
            '3 car c le minimum pour une combi C=V
            If Ligne.Length >= 3 Then
                'si présence de = et que la ligne n'est pas un commentaire
                If Ligne.Contains(_ATTRIBVAL) And Not Ligne(1).Equals(_COMMENTAIRE) Then
                    Dim maClef As String = Ligne.Split(_ATTRIBVAL)(0)
                    Dim maValeur As String = Ligne.Split(_ATTRIBVAL)(1)
                    Try
                        'si la clef est égale à PAT
                        If maClef = _PAT Then
                            If maValeur <> String.Empty Then
                                _maListePAT.Add(maValeur)
                            End If
                        Else
                            'si la clef n'existe pas déjà dans le hashtable
                            If Not listeArgs.ContainsKey(maClef) Then
                                listeArgs.Add(maClef, maValeur)
                            End If
                        End Try
                    End If
                End If
            End If
        Next
    End If
End Function

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>62/156</b>

```


        End If
    End If
    Catch ex As Exception
        MessageBox.Show("Une donnée dans le fichier de configuration (*.ini)
est corrompue", "Erreur combinaison mot clef - valeur", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
    End Try
End If
End If
Next
End If

Return listeArgs
End Function

''' <summary>
''' Retourne la valeur d'une clef passé en paramètre
''' </summary>
''' <param name="Clef"></param>
''' <returns>soit la valeur, soit nothing</returns>
''' <remarks></remarks>
Public Function GetValueFromKey(ByVal Clef As String) As String
    Return _KeyValueList(Clef)
End Function

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>63/156</b>

## 9.8 Outils

```
''' <summary>
''' Contient toutes les constantes et les énumérations
''' Contient également toutes les méthodes statiques
''' </summary>
''' <remarks></remarks>
```

```
Public Class Outils
```

```
#Region "Énumération"
```

```
''' <summary>
''' Détermine la vue à afficher
''' </summary>
''' <remarks></remarks>
```

```
Enum View As Integer
```

```
    Principale = 0
```

```
    autre = 1
```

```
End Enum
```

```
''' <summary>
''' Niveau d'accès aux dossiers
''' </summary>
''' <remarks></remarks>
```

```
Enum AccessFolder As Integer
```

```
    Probleme = -2
```

```
    Ignore = -1
```

```
    None = 0
```

```
    Lecture = 1
```

```
    Ecriture = 2
```

```
End Enum
```

```
''' <summary>
''' Label des 6 dossiers Prod du fichier INI
''' </summary>
''' <remarks></remarks>
```

```
Enum DossierProd As Integer
```

```
    DossierA = 1
```

```
    DossierB = 2
```

```
    DossierC = 3
```

```
    DossierD = 4
```

```
    DossierE = 5
```

```
    DossierF = 6
```

```
End Enum
```

```
Enum Action As Byte
```

```
    ConsultationOfficiel = 0
```

```
    ConsultationArchive = 1
```

```
    Impression = 2
```

```
    Importation = 3
```

```
    ExportationOfficiel = 4
```


```
    ExportationArchive = 5
```

```
    Archivage = 6
```

```
End Enum
```

```
#End Region
```

```
#Region "Constantes"
```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>64/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

'Constante des entetes des mode op de production
Public Const ET_DUPLICATA As String = "DUPLICATA"
Public Const ET_ORIGINAL As String = "ORIGINAL"
Public Const ET_PERIME As String = "PÉRIMÉ"

'Constantes du fichier ini [configuration]
Public Const INI_KEY_REPBASE As String = "BaseRep"
Public Const INI_KEY_REPTMP As String = "Tmp"
Public Const INI_IGNORE_CHAR As Char = "?"

''' <summary>
''' Clef de cryptage des fichiers word
''' </summary>
''' <remarks></remarks>
Public Const CLEF_CRYPTAGE As String = "f4ty52uG"

''' <summary>
''' Liste des mots de passe dans les fichiers Word Chimie avant importation dans l'interface
''' </summary>
''' <remarks></remarks>
Public Const LISTE_MDP_PRODUCTION As String =
"productions|dupli|produit|duplis|Production|Productions|Dupli|Produit|Duplis|Produits|productio
n|PRODUCTION|DUPLI|PRODUIT"

''' <summary>
''' extentions pour les fichiers cryptés
''' </summary>
''' <remarks></remarks>
Public Const EXT_FICHER_CRYPTER As String = ".crp"

''' <summary>
''' extentions pour les fichiers word
''' </summary>
''' <remarks></remarks>
Public Const EXT_FICHER_WORD As String = ".doc|.docx|.dotx|.dotm|.dot"


''' <summary>
''' Nom du fichier de configuration attendu pour initialiser l'interface
''' </summary>
''' <remarks></remarks>
Public Const NOM_FICHER_INI As String = "InterfaceModeOp2.ini"

Public Const EXT_SIMPLE_CRP As String = ".crp"
#End Region

End Class

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>65/156</b>

## 9.9 ArchDossProd

Imports System.IO

Imports InterfaceModeOp2.Outils

Public Class ArchDossProd

Private Const \_NBDossier As Integer = 6

Private \_mapDroit As New Hashtable

''' <summary>

''' Retourne l'accès d'un dossier qui a été déterminé au préalable

''' </summary>

''' <param name="monDossier"></param>

''' <value></value>

''' <returns></returns>

''' <remarks></remarks>

Public ReadOnly Property getAccess(ByVal monDossier As Outils.DossierProd) As AccessFolder  
Get

Return \_mapDroit(monDossier.ToString)

End Get

End Property

''' <summary>

''' Détermination en fonction des droits directs sur les dossiers Prod

''' </summary>

''' <remarks></remarks>

Sub New()

If Directory.Exists(Configuration.GetInstance.getBaseDir) Then

Dim ListeDossier = [Enum].GetValues(GetType(Outils.DossierProd))

For Each dossier As Outils.DossierProd In ListeDossier

\_mapDroit.Add(dossier.ToString,

getAccessFromFolder(Configuration.GetInstance.getFullProdDir(dossier)))

Next

End If

End Sub

''' <summary>

''' Prends le format ?01?22 par exemple

''' </summary>

''' <param name="CodeIni">le code en lui meme (?00?11)</param>

''' <remarks></remarks>

Sub New(ByVal CodeIni As String)

If Not IsNothing(CodeIni) Then

If CodeIni.Length = \_NBDossier Then

For i As Outils.DossierProd = 1 To \_NBDossier

\_mapDroit.Add(i.ToString, ToAccessFolder(CodeIni(i - 1)))

Next

End If

End If

End Sub


''' <summary>

''' Réalise des test en accès / lecture / écriture sur un dossier passé en paramètre

''' </summary>

''' <param name="nomDossier"></param>

''' <returns></returns>

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>66/156</b>

```

''' <remarks></remarks>
Private Function getAccessFromFolder(ByVal nomDossier As String) As AccessFolder
    Const NOM_TEST As String = "\test.txt"
    Const NO_REP As String = "\"

    Dim access As AccessFolder = AccessFolder.Probleme

    On Error GoTo EndOfFile

    If Not Directory.Exists(nomDossier) Then Throw New Exception

    access = AccessFolder.Ignore
    If nomDossier.EndsWith(NO_REP) Then Throw New Exception

    'test de l'accès au dossier
    access = AccessFolder.None
    Directory.GetFiles(nomDossier)


    'test si création d'un fichier est possible
    access = AccessFolder.Lecture
    Dim fsread As New FileStream(nomDossier & NOM_TEST, FileMode.Create)
    fsread.Dispose()
    File.Delete(nomDossier & NOM_TEST)
    access = AccessFolder.Ecriture
EndOfFile:
    Return access
End Function

''' <summary>
''' Converti les caractères provenant d'une string en élément accessFolder
''' </summary>
''' <param name="index">Elément d'une chaîne de caractère du type 00?11?</param>
''' <returns>L'équivalent en type AccessFolder</returns>
''' <remarks></remarks>
Private Function ToAccessFolder(ByVal index As Char) As AccessFolder
    Select Case index
        Case Outils.INI_IGNORE_CHAR
            Return AccessFolder.Ignore
        Case "-1"
            Return AccessFolder.Ignore
        Case "0"
            Return AccessFolder.None
        Case "1"
            Return AccessFolder.Lecture
        Case "2"
            Return AccessFolder.Ecriture
        Case Else
            Return AccessFolder.Probleme
    End Select
End Function

Public Overrides Function ToString() As String
    Return MyBase.ToString()
End Function

''' <summary>
''' Vérifie si les droits en écriture ou lecture sont suffisants pour avoir les droits
passé en paramètre
''' </summary>

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>67/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

''' <param name="Droit">le droit voulu qui est testé</param>
''' <returns>True si l'architecture permet ce droit</returns>
''' <remarks></remarks>
Public Function isEnoughFor(ByVal Droit As Droits) As Boolean
    Dim Arch As New ArchDossProd(Configuration.GetInstance.GetValueFromKey(Droit))
    Dim ListeDossier = [Enum].GetValues(GetType(Outils.DossierProd))

    For Each dossier As Outils.DossierProd In ListeDossier
        Dim DroitToUse As Outils.AccessFolder
        Dim monDroit As Outils.AccessFolder

        DroitToUse = Arch.getAccess(dossier)
        monDroit = Me.getAccess(dossier)

        If DroitToUse <> AccessFolder.Ignore Then
            If monDroit < DroitToUse Then
                Return False
            End If
        End If
    Next

    Return True
End Function


Public Shared Function GetListeDroitUser() As List(Of ArchDossProd)
    Dim maliste As New List(Of ArchDossProd)

    For i As Byte = 0 To 5
        maliste.Add(New ArchDossProd(Configuration.GetInstance.GetValueFromKey(i)))
    Next

    Return maliste
End Function

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>68/156</b>

## 9.10 UserConfig

Public Class UserConfig

```

Private _utilisateur As Utilisateur
'pc utilisateur
Private _nomPC As String
'droit architecture de production
Private _archiDossProd As ArchDossProd
'droit acces interface
Private _DroitReelInterface As Droits = Droits.Guest
'droit défini dans la base de données
Private _DroitBDD As Droits = Droits.Guest

```

#Region "Property"

```


'GETTER
Public ReadOnly Property getUsername() As String
    Get
        Return _utilisateur.getNomUtilisateur
    End Get
End Property
Public ReadOnly Property getUserId() As Double
    Get
        Return _utilisateur.idUtilisateur
    End Get
End Property
Public ReadOnly Property getPC() As String
    Get
        Return _nomPC
    End Get
End Property
Public ReadOnly Property getArchDossProd() As ArchDossProd
    Get
        Return _archiDossProd
    End Get
End Property
Public ReadOnly Property getDroitBDD() As Droits
    Get
        Return _DroitBDD
    End Get
End Property
Public ReadOnly Property getDroitDetermine() As Droits
    Get
        Return _DroitReelInterface
    End Get
End Property

```

```

'SETTER
Public WriteOnly Property setArchDossProd As String
    Set(ByVal value As String)
        _archiDossProd = New ArchDossProd(value)
        DetermineDroitReel()
    End Set
End Property
Public WriteOnly Property setDroitUser As Droits
    Set(ByVal value As Droits)
        _DroitBDD = value
        DetermineDroitReel()
    End Set
End Property

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>69/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

End Set
End Property
Public WriteOnly Property setUserId() As Double
    Set(ByVal value As Double)
        _utilisateur.idUtilisateur = value
    End Set
End Property
#End Region

Sub New()
    'détermination des droits utilisateurs sur dossiers de prod
    _archiDossProd = New ArchDossProd()

    'détermination de l'utilisateur
    _utilisateur = New Utilisateur(Environment.UserName)

    'détermine le nom du pc
    _nomPC = UCase(Environment.MachineName)

    'calcul des droits interface
    DetermineDroitReel()
End Sub

Private Sub DetermineDroitReel()
    If Not IsNothing(_archiDossProd) Then

        If _archiDossProd.isEnoughFor(_DroitBDD) Then
            _DroitReelInterface = _DroitBDD
            Exit Sub
        Else
            _DroitReelInterface = If(_archiDossProd.isEnoughFor(Droits.Guest), Droits.Guest,
Droits.NoUse)
            Exit Sub
        End If


    End If

    _DroitReelInterface = Droits.NoUse

End Sub

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>70/156</b>

## 9.11 Auditrails

Public Class **auditrails**

```

Private _idAuditrails As Integer
Private _nomFichier As String
Private _commentaire As String
Private _infosystem As String
Private _date As DateTime
Private _idUtilisateur As Integer
Private _idOperation As Operations

```

#Region "Property"

```

Public Property idAuditrails As Integer
    Set(ByVal value As Integer)
        _idAuditrails = value
    End Set
    Get
        Return _idAuditrails
    End Get
End Property
Public ReadOnly Property getNomFichierAuditrails As String
    Get
        Return _nomFichier
    End Get
End Property
Public ReadOnly Property getCommentaireAuditrails As String
    Get
        Return _commentaire
    End Get
End Property
Public ReadOnly Property getinfosystemAuditrails As String
    Get
        Return _infosystem
    End Get
End Property
Public ReadOnly Property getDateAuditrails As String
    Get
        Return DateToString(_date)
    End Get
End Property
Public ReadOnly Property getIdUtilisateur As Integer
    Get
        Return _idUtilisateur
    End Get
End Property
Public ReadOnly Property getIdOperation As Operations
    Get
        Return _idOperation
    End Get
End Property

```


#End Region

#Region "Constructeurs"

```

Sub New()
    _idAuditrails = -1
    _nomFichier = Nothing
    _commentaire = Nothing

```

	Libellé Système  <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système  <b>2300</b>	Type doc  <b>FS</b>	N°Ordre  <b>01</b>	Rév.  <b>F</b>	Page  <b>71/156</b>
---	---	------------------------------	---------------------------	--------------------------	----------------------	---------------------------


```

        _infosystem = Nothing
        _date = Nothing
        _idUtilisateur = -1
        _idOperation = -1
    End Sub
    Sub New(ByVal nomFichierAuditrails As String, ByVal commentaireAuditrails As String, ByVal
infosystem As String, _
        ByVal dateAuditrails As DateTime, ByVal idUtilisateur As Integer, ByVal idOperation
As Operations)
        _idAuditrails = -1
        _nomFichier = nomFichierAuditrails
        _commentaire = commentaireAuditrails
        _infosystem = infosystem
        _date = dateAuditrails
        _idUtilisateur = idUtilisateur
        _idOperation = idOperation
    End Sub
    Sub New(ByVal idAuditrails As Integer, ByVal nomFichierAuditrails As String, ByVal
commentaireAuditrails As String, ByVal infosystem As String, _
        ByVal dateAuditrails As DateTime, ByVal idUtilisateur As Integer, ByVal idOperation
As Operations)
        _idAuditrails = idAuditrails
        _nomFichier = nomFichierAuditrails
        _commentaire = commentaireAuditrails
        _infosystem = infosystem
        _date = dateAuditrails
        _idUtilisateur = idUtilisateur
        _idOperation = idOperation
    End Sub
#End Region

Public Overrides Function ToString() As String
    Dim description As New System.Text.StringBuilder("Audit Trails n°")
    With description
        .Append(_idAuditrails).AppendLine()
        .Append("ID utilisateur : ").Append(_idUtilisateur).AppendLine()
        .Append("operation : ").Append(_idOperation.ToString)
    End With
    Return description.ToString()
End Function

End Class

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>72/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

## 9.12 Enumeration

Public Module Enumeration

'pas de fonction CRUD pour ces tables

Public Enum Operations

Undefined = -1  
Impression = 0  
Importation = 1  
Exportation = 2  
Archivage = 3

End Enum

Public Enum Droits

NoUse = -1  
Guest = 0  
User = 1  
KeyUser = 2  
UserAQ = 3  
AdminAQ = 4  
AdminDvlp = 5

End Enum


Public Enum Verification

Undefined = -1  
NoVerif = 0  
EnCours = 1  
VerifOK = 2

End Enum

End Module



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>73/156</b>

### 9.13 HistoDroits

Public Class HistoDroits

```
Private _commentaire As String
Private _date As DateTime
Private _idDroit As Droits
Private _idUtilisateur As Double
```


#Region "Property"

```
Public Property commentaire As String
    Get
        Return _commentaire
    End Get
    Set(ByVal value As String)
        If IsNothing(value) Then value = ""
        _commentaire = value
    End Set
End Property
Public ReadOnly Property getDate As String
    Get
        Return DateToString(_date)
    End Get
End Property
Public ReadOnly Property getIdDroit As Droits
    Get
        Return _idDroit
    End Get
End Property
Public ReadOnly Property getIdUtilisateur As Double
    Get
        Return _idUtilisateur
    End Get
End Property
#End Region
```

#Region "Constructeurs"

```
Sub New()
    _commentaire = Nothing
    _date = Nothing
    _idDroit = -1
    _idUtilisateur = -1
End Sub
Sub New(ByVal commentaire As String, ByVal maDate As DateTime, ByVal idUtilisateur As Double, ByVal idDroit As Droits)
    _commentaire = commentaire
    _date = maDate
    _idDroit = idDroit
    _idUtilisateur = idUtilisateur
End Sub
#End Region
```

```
Public Overrides Function ToString() As String
    Dim str As New System.Text.StringBuilder
    With str
        .Append("Droit ")
        .Append(_idDroit)
        .Append(" de l'utilisateur ")
    End With
End Function
```


	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>74/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

        .Append(_idUser)
        .Append(" du ")
        .Append(_date.ToString)
    End With
    Return str.ToString()
End Function

```

```
End Class
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>75/156</b>

## 9.14 HistoVerification

```


Public Class HistoVerification
    Private _commentaire As String
    Private _date As DateTime
    Private _idVerification As Double
    Private _idAuditrails As Verification

#Region "Property"
    Public Property commentaire As String
        Get
            Return _commentaire
        End Get
        Set(ByVal value As String)
            If IsNothing(value) Then value = ""
            _commentaire = value
        End Set
    End Property
    Public ReadOnly Property getDate As String
        Get
            Return DateToString(_date)
        End Get
    End Property
    Public ReadOnly Property getIdVerification As Verification
        Get
            Return _idVerification
        End Get
    End Property
    Public ReadOnly Property getIdAuditrails As Double
        Get
            Return _idAuditrails
        End Get
    End Property
#End Region

#Region "Constructeurs"
    Sub New()
        _commentaire = Nothing
        _date = Nothing
        _idVerification = -1
        _idAuditrails = -1
    End Sub
    Sub New(ByVal commentaire As String, ByVal maDate As DateTime, ByVal idAuditrails As Double,
    ByVal idVerification As Verification)
        _commentaire = commentaire
        _date = maDate
        _idVerification = idVerification
        _idAuditrails = idAuditrails
    End Sub
#End Region

    Public Overrides Function ToString() As String
        Dim str As New System.Text.StringBuilder
        With str
            .Append("Verification ")
            .Append(_idVerification)
            .Append(" de l'audit trails ")
            .Append(_idAuditrails)
        End With
    End Function

```


	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>76/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

        .Append(" du ")
        .Append(_date.ToString)
    End With
    Return str.ToString()
End Function

```

```
End Class
```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>77/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

## 9.15 Impression


```

Public Class Impression
    Private _idImpression As Integer
    Private _lot As String
    Private _nomImprimante As String
    Private _pagesImpression As String
    Private _idAuditrails As Integer

#Region "Property"
    Public Property idImpression As Integer
        Get
            Return _idImpression
        End Get
        Set(ByVal value As Integer)
            _idImpression = value
        End Set
    End Property
    Public ReadOnly Property getLot As String
        Get
            Return _lot
        End Get
    End Property
    Public ReadOnly Property getNomImprimante As String
        Get
            Return _nomImprimante
        End Get
    End Property
    Public ReadOnly Property getPagesImpression As String
        Get
            Return _pagesImpression
        End Get
    End Property
    Public Property idAuditrails As Integer
        Get
            Return _idAuditrails
        End Get
        Set(ByVal value As Integer)
            _idAuditrails = value
        End Set
    End Property
#End Region

#Region "Constructeurs"
    Sub New()
        _idImpression = -1
        _lot = Nothing
        _nomImprimante = Nothing
        _pagesImpression = Nothing
        _idAuditrails = -1
    End Sub
    Sub New(ByVal lot As String, ByVal nomImprimante As String, ByVal pagesImpression As String,
Optional ByVal idAuditrails As Integer = -1)
        _idImpression = -1
        _lot = lot
        _nomImprimante = nomImprimante
        _pagesImpression = pagesImpression
        _idAuditrails = idAuditrails
    End Sub

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>78/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------


```

End Sub
Sub New(ByVal idImpression As Integer, ByVal lot As String, ByVal nomImprimante As String,
ByVal pagesImpression As String, ByVal idAuditrails As Integer)
    _idImpression = idImpression
    _lot = lot
    _nomImprimante = nomImprimante
    _pagesImpression = pagesImpression
    _idAuditrails = idAuditrails
End Sub
#End Region

Public Overrides Function ToString() As String
    Dim description As New System.Text.StringBuilder("Impression n°")
    With description
        .Append(_idImpression).AppendLine()
        .Append("Audit trails n°").Append(_idAuditrails)
    End With
    Return description.ToString()
End Function

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>79/156</b>

## 9.16 Signets


```

Public Class Signets
    Private _idSignet As Integer
    Private _clef As String
    Private _valeur As String
    Private _code As String
    Private _idImpression As Integer

#Region "Property"
    Public Property idSignet As Integer
        Get
            Return _idSignet
        End Get
        Set(ByVal value As Integer)
            _idSignet = value
        End Set
    End Property
    Public ReadOnly Property getClefSignet As String
        Get
            Return _clef
        End Get
    End Property
    Public ReadOnly Property getValeurSignet As String
        Get
            Return _valeur
        End Get
    End Property
    Public ReadOnly Property getCodeSignet As String
        Get
            Return _code
        End Get
    End Property
    Public Property idImpression As Integer
        Get
            Return _idImpression
        End Get
        Set(ByVal value As Integer)
            _idImpression = value
        End Set
    End Property
#End Region

#Region "Constructeurs"
    Sub New()
        _idSignet = -1
        _clef = Nothing
        _valeur = Nothing
        _code = Nothing
        _idImpression = -1
    End Sub
    Sub New(ByVal description As String, ByVal valeur As String, ByVal code As String, Optional
ByVal idImpression As Integer = -1)
        _idSignet = -1
        _clef = description
        _valeur = valeur
        _code = code
        _idImpression = idImpression
    End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>80/156</b>

```

End Sub
Sub New(ByVal idSignet As Integer, ByVal description As String, ByVal valeur As String,
ByVal code As String, ByVal idImpression As Integer)
    _idSignet = idSignet
    _clef = description
    _valeur = valeur
    _code = code
    _idImpression = idImpression
End Sub
#End Region

Public Overrides Function Equals(ByVal obj As Object) As Boolean

    If TypeOf obj Is Signets Then
        Dim eq1 As New System.Text.StringBuilder()
        eq1.Append(_idImpression).Append(_clef).Append(_valeur)


        Dim eq2 As New System.Text.StringBuilder()
        eq2.Append(obj.idImpression).Append(obj.getClefSignet).Append(obj.getValeurSignet)

        Return eq1.Equals(eq2)
    Else
        Return False
    End If
End Function
Public Overrides Function ToString() As String
    Dim description As New System.Text.StringBuilder("Signet N°")
    With description
        .Append(_idSignet).AppendLine()
        .Append("Impression n°").Append(_idImpression).AppendLine()
        .Append("code : ").Append(_code)
    End With
    Return description.ToString()
End Function

End Class

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>81/156</b>

## 9.17 Utilisateur

Public Class Utilisateur

```
Private _idUtilisateur As Integer
Private _nomUtilisateur As String
```

#Region "Property"

```
Public Property idUtilisateur As Integer
    Get
        Return _idUtilisateur
    End Get
    Set(ByVal value As Integer)
        _idUtilisateur = value
    End Set
End Property
```

```
Public ReadOnly Property getNomUtilisateur As String
    Get
        Return _nomUtilisateur
    End Get
End Property
```

#End Region

#Region "Constructeur"

```
Sub New()
    Me._idUtilisateur = -1
    Me._nomUtilisateur = Nothing
End Sub
```

```
Sub New(ByVal nomUtilisateur As String)
    Me._idUtilisateur = -1
    Me._nomUtilisateur = nomUtilisateur.ToUpper
End Sub
```

```
Sub New(ByVal idUtilisateur As Integer, ByVal nomUtilisateur As String)
    Me._idUtilisateur = idUtilisateur
    Me._nomUtilisateur = nomUtilisateur.ToUpper
End Sub
```


#End Region

#Region "Méthodes Publiques"

```
Public Overrides Function toString() As String
    Dim description As New System.Text.StringBuilder("Utilisateur n°")
    With description
        .Append(Me._idUtilisateur).AppendLine()
        .Append("nom : ").Append(Me._nomUtilisateur)
    End With
    Return description.ToString
End Function
```

#End Region

End Class

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>82/156</b>

## 9.18 AbstractDAO

Imports System.Data.SQLite

Public MustInherit Class AbstractDAO

```
Protected Const ERR_INSERT = "Erreur lors de l'insertion de l'objet {0} : {1}"
Protected Const ERR_GETALL = "Erreur lors de la récupération de la liste des objets {0}"
Protected Const ERR_GETID = "Erreur lors de la récupération de l'objet {0} ayant l'id {1}"
Protected Const ERR_UPDATE = "Erreur lors de la mise à jour de l'objet {0} : {1}"
Protected Const ERR_UPDATE_NOT1 = "l'objet {0} ayant l'id n°{1} n'a pas pu être mis à jour"
Protected Const ERR_DELETE = "Erreur lors de la suppression de l'objet {0} : {1}"
```

```
Protected Const DAO_LASTID = ";select last_insert_rowid()"
```

```
Private Const DAO_INSERT = "DAO_INSERT"
Private Const DAO_GETALL = "DAO_GETALL"
Private Const DAO_GETID = "DAO_GETID {0}"
Private Const DAO_UPDATE = "DAO_UPDATE"
Private Const DAO_DELETE = "DAO_DELETE"
```

```
Protected Function DAOInsert(ByVal RQ_INSERT As String, ByVal ParamArray ParamList() As
SQLiteParameter) As Double
```

```
Dim newID As Double = -1
```

```
'définition de la requete
```

```
Using tr As SQLiteTransaction = DAOFactory.getConnexion.BeginTransaction
```

```
Using requete As SQLiteCommand = New SQLiteCommand()
```

```
Try
```

```
'ajout dans la table
```

```
With requete
```

```
.Transaction = tr
```

```
.CommandText = RQ_INSERT & DAO_LASTID
```

```
.Parameters.AddRange(ParamList)
```

```
.Prepare()
```

```
newID = .ExecuteScalar()
```

```
End With
```

```
Catch ex As Exception
```

```
tr.Rollback()
```

```
Throw New DAOException(getErrMsg(RQ_INSERT, ex.Message, ParamList))
```

```
End Try
```

```
tr.Commit()
```

```
Return newID
```

```
End Using
```

```
End Using
```

```
End Function
```

```
Protected Function DAOGetAll(ByVal RQ_SELECTALL As String) As SQLiteDataReader
```


```
'définition de la requete
```

```
Using requete As SQLiteCommand = New SQLiteCommand(RQ_SELECTALL,
DAOFactory.getConnexion)
```

```
Try
```

```
'execution de la requete
```

```
Dim reader As SQLiteDataReader = requete.ExecuteReader()
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>83/156</b>

```

        'récupération du jeu de résultats
        Return reader

    Catch ex As Exception
        Throw New DAOException(getErrorMsg(RQ_SELECTALL, ex.Message))
    End Try
End Using
End Function

Protected Function DAOGetById(ByRef id As Integer, ByVal RQ_GETID As String) As
SQLiteDataReader
    'définition de la requete
    Using requete As SQLiteCommand = New SQLiteCommand(RQ_GETID, DAOFactory.getConnexion)
        Try

            'définition des paramètres à mettre à jour
            Dim p1 = New SQLiteParameter()
            p1.DbType = DbType.Double
            p1.Value = id

            'intégration des paramètres à la requete
            requete.Parameters.Add(p1)

            'execution de la requete
            requete.Prepare()
            Dim reader As SQLiteDataReader = requete.ExecuteReader()

            Return reader

        Catch ex As Exception
            Throw New DAOException(getErrorMsg(RQ_GETID, ex.Message, id))
        End Try
    End Using
End Function

Protected Sub DAOUpdate(ByVal RQ_UPDATE As String, ByVal ParamArray ParamList() As
SQLiteParameter)
    'définition de la requete
    Using requete As SQLiteCommand = New SQLiteCommand(RQ_UPDATE, DAOFactory.getConnexion)

        Try


            'intégration des paramètres à la requete
            requete.Parameters.AddRange(ParamList)

            'exécution de la requete
            requete.Prepare()
            requete.ExecuteNonQuery()

        Catch ex As Exception
            Throw New DAOException(getErrorMsg(RQ_UPDATE, ex.Message, ParamList))
        End Try
    End Using
End Sub

Protected Function DAODelete(ByVal id As Double, ByVal RQ_DELETE As String) As Boolean
    Dim SuppObj As Boolean = False

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>84/156</b>

'définition de la requete

```
Using requete As SQLiteCommand = New SQLiteCommand(RQ_DELETE, DAOFactory.getConnexion)
```

```
Try
```

'définition des paramètres à mettre à jour

```
Dim p1 = New SQLiteParameter()
```

```
p1.DbType = DbType.Double
```

```
p1.Value = id
```

'intégration des paramètres à la requete

```
requete.Parameters.Add(p1)
```

'exécution de la requete

```
requete.Prepare()
```

```
SuppObj = requete.ExecuteNonQuery()
```

```
Catch ex As Exception
```

```
Throw New DAOException(getErrorMsg(RQ_DELETE, ex.Message, id))
```

```
End Try
```

```
Return SuppObj
```

```
End Using
```

```
End Function
```

```
Protected Function DAOGetByString(ByRef str As String, ByVal RQ_GETByStr As String) As  
SQLiteDataReader
```

'définition de la requete

```
Using requete As SQLiteCommand = New SQLiteCommand(RQ_GETByStr, DAOFactory.getConnexion)
```

```
Try
```

'définition des paramètres à mettre à jour

```
Dim p1 = New SQLiteParameter()
```

```
p1.DbType = DbType.String
```

```
p1.Value = str
```

'intégration des paramètres à la requete

```
requete.Parameters.Add(p1)
```

'execution de la requete

```
requete.Prepare()
```

```
Dim reader As SQLiteDataReader = Nothing
```

```
Try
```

```
reader = requete.ExecuteReader()
```

```
Catch ex As Exception
```

```
Return Nothing
```

```
End Try
```

```
Return reader
```

```
Catch ex As Exception
```

```
Throw New DAOException(getErrorMsg(RQ_GETByStr, ex.Message, str))
```

```
End Try
```


```
End Using
```

```
End Function
```

```
Private Overloads Function getErrorMsg(ByVal requete As String, ByVal errmsg As String,  
Optional ByVal id As Integer = -1)
```

```
Dim msg As New System.Text.StringBuilder(requete)
```

```
With msg
```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>85/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

.AppendLine()

'ajout liste des paramètres si présente
If id <> -1 Then
    .Append("[").Append(id)
    .Append("]").AppendLine()
End If

```

```

.AppendLine(errmsg)

End With
Return msg.ToString
End Function

```

```

Private Overloads Function getErrorMsg(ByVal requete As String, ByVal errmsg As String,
ByVal ParamArray params() As SQLiteParameter) As String
    Dim msg As New System.Text.StringBuilder(requete)
    With msg
        .AppendLine()


        'ajout liste des paramètres si présente
        If Not IsNothing(params) Then
            .Append("[")
            For Each p As SQLiteParameter In params
                .Append(p.Value).Append(", ")
            Next
            .Remove(.Length - 2, 2)
            .Append("]").AppendLine()
        End If

        .Append(errmsg)

    End With
    Return msg.ToString
End Function

End Class

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>86/156</b>
--	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

## 9.19 DAOException


```

Public Class DAOException
    Inherits Exception

    Private Const _ERR_NOSOURCE = "Pas d'erreur source"
    Private _errSource As String

    Public ReadOnly Property getErrSource As String
        Get
            Return _errSource
        End Get
    End Property
    Public Sub New(ByVal errmsg As String, Optional ByVal errExcep As Exception = Nothing)
        MyBase.New(errmsg)
        _errSource = If(IsNothing(errExcep), _ERR_NOSOURCE, errExcep.Message)
    End Sub
End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>87/156</b>

## 9.20 DAOFactory

Imports System.Data.SQLite

Public Class DAOFactory

Private Shared connection As SQLiteConnection = Singleton.GetInstance()

Public Shared ReadOnly Property getConnexion() As SQLiteConnection  
Get  
Return connection  
End Get  
End Property

Public Shared Function getUtilisateur() As DAOUtilisateur  
Return New DAOUtilisateur()  
End Function

Public Shared Function getAuditrails() As DAOAuditrails  
Return New DAOAuditrails  
End Function

Public Shared Function getImpression() As DAOImpression  
Return New DAOImpression  
End Function

Public Shared Function getSignet() As DAOSignet  
Return New DAOSignet  
End Function


Public Shared Function getHistoDroit() As DAO\_HDroit  
Return New DAO\_HDroit  
End Function

Public Shared Function getHistoVerification() As DAO\_HVerif  
Return New DAO\_HVerif  
End Function

Public Shared Function getATPrinter() As DAOSpecificTransaction  
Return New DAOSpecificTransaction  
End Function

Public Shared Function getViews() As DAOViews  
Return New DAOViews  
End Function

End Class

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>88/156</b>
--	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

## 9.21 IDAO

Public Interface IDAO(Of T)

Sub dbInsert(ByRef value As T)

Function dbGetAll() As List(Of T)

Function dbGetById(ByRef id As Integer) As T

Function dbDelete(ByRef value As T) As Boolean


Sub dbUpdate(ByRef value As T)

Function getUpdateParameters(ByVal value As T) As SQLite.SQLiteParameter()

Function getInsertParameters(ByVal value As T) As SQLite.SQLiteParameter()

End Interface



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>89/156</b>

## 9.22 IDAO\_Histo

```

''' <summary>
''' Interface pour les tables/bean commençant par histo_
''' </summary>
''' <typeparam name="Histo">nom de l'objet histo</typeparam>
''' <remarks></remarks>
Public Interface IDAO_Histo(Of Histo)

    Enum ObjectType As Byte
        Objet = 0
        Statut = 1
    End Enum

    ''' <summary>
    ''' Enregistre un nouvel objet de type Histo dans la base de données
    ''' </summary>
    ''' <param name="value">objet de type Histo à enregistrer</param>
    ''' <remarks></remarks>
    Sub dbInsert(ByRef value As Histo)

    ''' <summary>
    ''' récupère l'historique d'un objet
    ''' </summary>
    Function dbGetAllStatutById(ByRef id As Integer) As List(Of Histo)

    ''' <summary>
    ''' récupère le dernier statut d'un objet
    ''' </summary>
    Function dbGetLastStatutById(ByRef id As Integer) As Histo


    ''' <summary>
    ''' Mise à jour du commentaire de l'historique
    ''' </summary>
    ''' <param name="value">Objet de type Histo à mettre à jour</param>
    ''' <remarks></remarks>
    Sub dbUpdate(ByRef value As Histo)

    Function getUpdateParameters(ByVal value As Histo) As SQLite.SQLiteParameter()

    Function getInsertParameters(ByVal value As Histo) As SQLite.SQLiteParameter()

End Interface

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>90/156</b>

## 9.23 Singleton

Imports System.Data.SQLite

Imports System.IO


```
''' <summary>
''' Singleton de connexion base SQLite
''' </summary>
''' <remarks></remarks>
Public Class Singleton
    'Paramètres de connexion
    Private Const DATASOURCE = "Data Source="
    Public Const DBFOLDER = "dbFolder"
    Private Const VERSION = "Version=3"
    Private Const PASSWORD = "Password="
    Private Const PWD = "tXpD56gN"
    Private Const SEPARATOR = ";"
    Private Const RQ_TEST = "select * from enum_droits"

    'Objet de type connexion SQLite
    Private Shared _Instance As Singleton = Nothing
    Private Shared _Connexion As SQLiteConnection = Nothing
    Private Shared _ModeProtection As SByte = -1

    ''' <summary>
    ''' -1:Probleme; 0:NonProtege, 1:protege
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Shared ReadOnly Property getModeProtection() As SByte
        Get
            Return _ModeProtection
        End Get
    End Property

    ''' <summary>
    ''' chaîne de connexion à la base de données
    ''' </summary>
    ''' <value></value>
    ''' <returns>Chaîne de connexion SQLite</returns>
    ''' <remarks></remarks>
    Private ReadOnly Property connString(Optional ByVal bddProtege As Boolean = True) As String
        Get
            Dim DBFILE = Configuration.GetInstance.GetValueFromKey(DBFOLDER)
            If Not bddProtege Then
                Return DATASOURCE & DBFILE & SEPARATOR & VERSION
            Else
                Return DATASOURCE & DBFILE & SEPARATOR & VERSION & SEPARATOR & PASSWORD & PWD
            End If
        End Get
    End Property

    ''' <summary>
    ''' Constructeur en mode singleton
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub New()
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>91/156</b>

```

Try
    'création de la connexion
    _Connexion = New SQLiteConnection(connString())

    'ouverture de la connexion
    _Connexion.Open()

    If Not ConnexionOK() Then
        _Connexion = New SQLiteConnection(connString(False))
        _Connexion.Open()
        _ModeProtection = 0
    Else
        _ModeProtection = 1
    End If

Catch ex As Exception
    Throw New DAOException("Problème lors de la connexion à la base de données", ex)
End Try
End Sub

'Test la connexion en vérifiant si résultat avec enum_droit
'Permet de tester si base protégé par mot de passe
'car si mdp vide, la connexion est ok mais aucune table visible
Private Function ConnexionOK() As Boolean
    Using requete As SQLiteCommand = New SQLiteCommand(RQ_TEST, _Connexion)

        Try
            Dim reader As SQLiteDataReader = requete.ExecuteReader()


            If reader.Read() Then
                Return True
            Else : Return False
            End If

        Catch ex As Exception
            Return False
        End Try
    End Using
End Function

Public Shared Sub protectionBDD(Optional ByVal protect As Boolean = True)
    If protect Then
        _Connexion.ChangePassword(PWD)
        _ModeProtection = 1
    Else
        _Connexion.ChangePassword("")
        _ModeProtection = 0
    End If
End Sub

''' <summary>
''' Récupération de l'instance de connexion
''' </summary>
''' <returns></returns>
''' <remarks></remarks>
Public Shared Function getInstance() As SQLiteConnection
    Try
        If _Connexion Is Nothing Then
            _Instance = New Singleton()
            'Console.WriteLine("Connexion bdd ok : " & vbNewLine & _Connexion.FileName)

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>92/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

        End If
        Return _Connexion
    Catch ex As Exception
        Return Nothing
    End Try
End Function


''' <summary>
''' fermeture de la connexion
''' </summary>
''' <remarks></remarks>
Public Shared Sub close()

    If Not _Connexion Is Nothing Then
        _Connexion.Close()
        _Connexion = Nothing
    End If

End Sub

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>93/156</b>

## 9.24 DAOAuditrails

Imports System.Data.SQLite

Public Class DAOAuditrails

Inherits AbstractDAO

Implements IDAO(Of auditrails)

Public Function dbDelete(ByRef value As auditrails) As Boolean Implements IDAO(Of auditrails).dbDelete

Return MyBase.DAODelete(value.idAuditrails, AT\_DELETE)

End Function

Public Function dbGetAll() As System.Collections.Generic.List(Of auditrails) Implements IDAO(Of auditrails).dbGetAll

'création de la liste

Dim maListeObj As New List(Of auditrails)

Try

'execution de la requete

Dim reader As SQLiteDataReader = MyBase.DAOGetAll(AT\_READ)

'récupération du jeu de résultats

While reader.Read()

maListeObj.Add(New auditrails(reader(0).ToString,  
reader(1).ToString,  
reader(2).ToString,  
reader(3).ToString,  
Convert.ToDateTime(reader(4).ToString),  
reader(5).ToString,  
reader(6).ToString))

End While

'libération des objets

reader.Close()

Return maListeObj

Catch ex As Exception

Throw New DAOException(String.Format(ERR\_GETALL, AT\_ERR\_TYPEOF), ex)

End Try

End Function

Public Function dbGetById(ByRef id As Integer) As auditrails Implements IDAO(Of auditrails).dbGetById

Dim obj As auditrails = Nothing

Try

Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, AT\_READBYID)


If reader.Read() Then

'récupération du résultat

obj = New auditrails(reader(0).ToString,  
reader(1).ToString,  
reader(2).ToString,  
reader(3).ToString,  
Convert.ToDateTime(reader(4).ToString),  
reader(5).ToString,  
reader(6).ToString)

End If

'libération des objets

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>94/156</b>

```

        reader.Close()

    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_GETID, AT_ERR_TYPEOF, id), ex)
    End Try
    Return obj
End Function

Public Sub dbInsert(ByRef value As auditrails) Implements IDAO(Of auditrails).dbInsert
    Try
        value.idAuditrails = MyBase.DAOInsert(AT_CREATE, getInsertParameters(value))
    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_INSERT, AT_ERR_TYPEOF, value.ToString), ex)
    End Try

End Sub

Public Sub dbUpdate(ByRef value As auditrails) Implements IDAO(Of auditrails).dbUpdate

    Try
        MyBase.DAOUpdate(AT_UPDATE, getUpdateParameters(value))

    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_UPDATE, AT_ERR_TYPEOF, value.ToString), ex)
    End Try
End Sub

Public Function getInsertParameters(ByVal value As auditrails) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of auditrails).getInsertParameters
    'définition des paramètres à mettre à jour
    Dim p1 = New SQLiteParameter()
    p1.DbType = DbType.String
    p1.Value = value.getNomFichierAuditrails

    Dim p2 = New SQLiteParameter()
    p2.DbType = DbType.String
    p2.Value = value.getCommentaireAuditrails

    Dim p5 = New SQLiteParameter()
    p5.DbType = DbType.String
    p5.Value = value.getinfosystemAuditrails

    Dim p3 = New SQLiteParameter()
    p3.DbType = DbType.String
    p3.Value = value.getDateAuditrails


    Dim p4 = New SQLiteParameter()
    p4.DbType = DbType.Double
    p4.Value = value.getIdUtilisateur

    Dim p6 = New SQLiteParameter()
    p6.DbType = DbType.Double
    p6.Value = value.getIdOperation

    Return {p1, p2, p5, p3, p4, p6}
End Function

Public Function getUpdateParameters(ByVal value As auditrails) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of auditrails).getUpdateParameters


```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>95/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	-----------------------

```

Dim p5 = New SQLiteParameter()
p5.DbType = DbType.Double
p5.Value = value.idAuditrails
Return {getInsertParameters(value), p5}
End Function
End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>96/156</b>

## 9.25 DAOImpression

```
Imports System.Data.SQLite
```

```
Public Class DAOImpression
```

```
    Inherits AbstractDAO
```

```
    Implements IDAO(Of Impression)
```

```
    Public Function dbDelete(ByRef value As Impression) As Boolean Implements IDAO(Of Impression).dbDelete
```

```
        Return MyBase.DAODelete(value.idImpression, IMP_DELETE)
```

```
    End Function
```

```
    Public Function dbGetAll() As System.Collections.Generic.List(Of Impression) Implements IDAO(Of Impression).dbGetAll
```

```
        'création de la liste
```

```
        Dim maListeObj As New List(Of Impression)
```

```
        Try
```

```
            'execution de la requete
```

```
            Dim reader As SQLiteDataReader = MyBase.DAOGetAll(IMP_READ)
```

```
            'récupération du jeu de résultats
```

```
            While reader.Read()
```

```
                maListeObj.Add(New Impression(reader(0).ToString,
                                                reader(1).ToString,
                                                reader(2).ToString,
                                                reader(3).ToString,
                                                reader(4).ToString))
```

```
            End While
```

```
            'libération des objets
```

```
            reader.Close()
```

```
        Catch ex As Exception
```

```
            Throw New DAOException(String.Format(ERR_GETALL, IMP_ERR_TYPEOF), ex)
```

```
        End Try
```

```
        Return maListeObj
```

```
    End Function
```

```
    Public Function dbGetById(ByRef id As Integer) As Impression Implements IDAO(Of Impression).dbGetById
```

```
        Dim obj As Impression = Nothing
```

```
        Try
```

```
            Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, IMP_READBYID)
```

```
            If reader.Read() Then
```

```
                'récupération du résultat
```


```
                obj = New Impression(reader(0).ToString,
                                      reader(1).ToString,
                                      reader(2).ToString,
                                      reader(3).ToString,
                                      reader(4).ToString)
```

```
            End If
```

```
            'libération des objets
```

```
            reader.Close()
```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>97/156</b>

```

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_GETID, IMP_ERR_TYPEOF, id), ex)
        End Try
        Return obj

    End Function

    Public Sub dbInsert(ByRef value As Impression) Implements IDAO(Of Impression).dbInsert
        Try

            value.idImpression = MyBase.DAOInsert(IMP_CREATE, getInsertParameters(value))

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_INSERT, IMP_ERR_TYPEOF, value.ToString),
ex)
        End Try
    End Sub

    Public Sub dbUpdate(ByRef value As Impression) Implements IDAO(Of Impression).dbUpdate
        Try
            MyBase.DAOUpdate(IMP_UPDATE, getUpdateParameters(value))

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_UPDATE, IMP_ERR_TYPEOF, value.ToString),
ex)
        End Try
    End Sub

    Public Function getInsertParameters(ByVal value As Impression) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of Impression).getInsertParameters
        'définition des paramètres à mettre à jour
        Dim p0 = New SQLiteParameter()
        p0.DbType = DbType.String
        p0.Value = value.getLot


        Dim p1 = New SQLiteParameter()
        p1.DbType = DbType.String
        p1.Value = value.getNomImprimante

        Dim p2 = New SQLiteParameter()
        p2.DbType = DbType.String
        p2.Value = value.getPagesImpression

        Dim p3 = New SQLiteParameter()
        p3.DbType = DbType.String
        p3.Value = value.idAuditrails
        Return {p0, p1, p2, p3}
    End Function

    Public Function getUpdateParameters(ByVal value As Impression) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of Impression).getUpdateParameters
        Dim p4 = New SQLiteParameter()
        p4.DbType = DbType.Double
        p4.Value = value.idImpression
        Return {getInsertParameters(value), p4}
    End Function
End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>98/156</b>

## 9.26 DAOSignet

```
Imports System.Data.SQLite
```

```
Public Class DAOSignet
    Inherits AbstractDAO
    Implements IDAO(Of Signets)

    Public Sub dbInsertMulti(ByRef value As List(Of Signets))
        Dim tr As SQLiteTransaction = DAOFactory.getConnexion.BeginTransaction
        Using tr
            Try
                For Each signet As Signets In value
                    Dim cmd As SQLiteCommand = DAOFactory.getConnexion.CreateCommand
                    With cmd
                        .Transaction = tr
                        .CommandText = SGT_CREATE
                        .Parameters.AddRange(getInsertParameters(signet))
                        .Prepare()
                        .ExecuteNonQuery()
                    End With
                Next
            Catch ex As Exception
                tr.Rollback()
                Throw New DAOException("erreur lors de l'insertion des signets dans la table
signet", ex)
            End Try
            tr.Commit()
        End Using
    End Sub


    Public Function dbDelete(ByRef value As Signets) As Boolean Implements IDAO(Of
Signets).dbDelete
        Return MyBase.DAODelete(value.idSignet, SGT_DELETE)
    End Function

    Public Function dbGetAll() As System.Collections.Generic.List(Of Signets) Implements IDAO(Of
Signets).dbGetAll
        'création de la liste
        Dim maListeObj As New List(Of Signets)

        Try

            'execution de la requete
            Dim reader As SQLiteDataReader = MyBase.DAOGetAll(SGT_READ)
            'récupération du jeu de résultats
            While reader.Read()
                maListeObj.Add(New Signets(reader(0).ToString,
                    reader(1).ToString,
                    reader(2).ToString,
                    reader(3).ToString,
                    reader(4).ToString))
            End While
            'libération des objets
            reader.Close()

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_GETALL, SGT_ERR_TYPEOF), ex)
        End Try
    End Function
End Class
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>99/156</b>

```

End Try
Return maListeObj
End Function

Public Function dbGetById(ByRef id As Integer) As Signets Implements IDAO(Of
Signets).dbGetById
Dim obj As Signets = Nothing

Try

    'définition des paramètres à mettre à jour
Dim p1 = New SQLiteParameter()
p1.DbType = DbType.Double
p1.Value = id

Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, SGT_READBYID)

If reader.Read() Then
    'récupération du résultat
    obj = New Signets(reader(0).ToString,
                        reader(1).ToString,
                        reader(2).ToString,
                        reader(3).ToString,
                        reader(4).ToString)

End If

    'libération des objets
reader.Close()

Catch ex As Exception
    Throw New DAOException(String.Format(ERR_GETID, SGT_ERR_TYPEOF, id), ex)
End Try
Return obj

End Function

Public Sub dbInsert(ByRef value As Signets) Implements IDAO(Of Signets).dbInsert
Try
    value.idSignet = MyBase.DAOInsert(SGT_CREATE, getInsertParameters(value))

Catch ex As Exception
    Throw New DAOException(String.Format(ERR_INSERT, SGT_ERR_TYPEOF, value.ToString),
ex)
End Try
End Sub


Public Sub dbUpdate(ByRef value As Signets) Implements IDAO(Of Signets).dbUpdate
Try

    MyBase.DAOUpdate(SGT_UPDATE, getUpdateParameters(value))

Catch ex As Exception
    Throw New DAOException(String.Format(ERR_UPDATE, SGT_ERR_TYPEOF, value.ToString),
ex)
End Try

End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>100/156</b>

```

Public Function getInsertParameters(ByVal value As Signets) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of Signets).getInsertParameters
    'définition des paramètres à mettre à jour
    Dim p1 = New SQLiteParameter()
    p1.DbType = DbType.String
    p1.Value = value.getClefSignet

    Dim p2 = New SQLiteParameter()
    p2.DbType = DbType.String
    p2.Value = value.getValeurSignet

    Dim p3 = New SQLiteParameter()
    p3.DbType = DbType.String
    p3.Value = value.getCodeSignet

    Dim p4 = New SQLiteParameter()
    p4.DbType = DbType.Double
    p4.Value = value.idImpression

    Return {p1, p2, p3, p4}
End Function


Public Function getUpdateParameters(ByVal value As Signets) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of Signets).getUpdateParameters

    Dim p5 = New SQLiteParameter()
    p5.DbType = DbType.Double
    p5.Value = value.idSignet

    Return {getInsertParameters(value), p5}

End Function
End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>101/156</b>

## 9.27 DAOUtilisateur

```
Imports System.Data.SQLite
Public Class DAOUtilisateur
    Inherits AbstractDAO
    Implements IDAO(Of Utilisateur)

    Public Function dbGetByName(ByRef Name As String) As Utilisateur
        Dim monUtilisateur As Utilisateur = Nothing

        Try

            Dim reader As SQLiteDataReader = MyBase.DAOGetByString(Name, USR_READBYNAME)

            If IsNothing(reader) Then
                Return Nothing
            End If
            'récupération du résultat
            If reader.Read() Then
                monUtilisateur = (New Utilisateur(reader(0).ToString,
                                                    reader(1).ToString))
            End If

            'libération des objets
            reader.Close()

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_GETID, USR_ERR_TYPEOF, Name), ex)
        End Try

        Return monUtilisateur
    End Function

    Public Sub dbInsert(ByRef value As Utilisateur) Implements IDAO(Of Utilisateur).dbInsert
        Try

            value.idUtilisateur = MyBase.DAOInsert(USR_CREATE, getInsertParameters(value))


        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_INSERT, USR_ERR_TYPEOF, value.toString),
ex)
        End Try
    End Sub

    Public Function dbGetAll() As List(Of Utilisateur) Implements IDAO(Of Utilisateur).dbGetAll
        'création de la liste d'utilisateurs
        Dim users As New List(Of Utilisateur)

        Try

            'execution de la requete
            Dim reader As SQLiteDataReader = MyBase.DAOGetAll(USR_READ)
            'récupération du jeu de résultats
            While reader.Read()
                users.Add(New Utilisateur(reader(0).ToString,
                                           reader(1).ToString))
            End While

        End Try
    End Function
End Class
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>102/156</b>

```

        'libération des objets
        reader.Close()

    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_GETALL, USR_ERR_TYPEOF), ex)
    End Try

    Return users
End Function

Public Function dbGetById(ByRef id As Integer) As Utilisateur Implements IDAO(Of
Utilisateur).dbGetById
    Dim monUtilisateur As Utilisateur = Nothing

    Try
        Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, USR_READBYID)

        'récupération du résultat
        If reader.Read() Then
            monUtilisateur = (New Utilisateur(reader(0).ToString,
                                                reader(1).ToString))
        End If

        'libération des objets
        reader.Close()

    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_GETID, USR_ERR_TYPEOF, id), ex)
    End Try

    Return monUtilisateur

End Function

Public Function dbDelete(ByRef value As Utilisateur) As Boolean Implements IDAO(Of
Utilisateur).dbDelete
    Return MyBase.DAODelete(value.idUtilisateur, USR_DELETE)
End Function

Public Sub dbUpdate(ByRef value As Utilisateur) Implements IDAO(Of Utilisateur).dbUpdate
    Dim nbLigneUpdate As Integer = 0

    Try

        MyBase.DAOUpdate(USR_UPDATE, getUpdateParameters(value))


    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_UPDATE, USR_ERR_TYPEOF, value.toString),
ex)
    End Try

End Sub

Public Function getInsertParameters(ByVal value As Utilisateur) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of Utilisateur).getInsertParameters

    'définition des paramètres à mettre à jour
    Dim p1 = New SQLiteParameter()
    p1.DbType = DbType.String

```


	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>103/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```
p1.Value = value.getNomUtilisateur
```

```
Return {p1}
End Function
```

```
Public Function getUpdateParameters(ByVal value As Utilisateur) As
System.Data.SQLite.SQLiteParameter() Implements IDAO(Of Utilisateur).getUpdateParameters
Dim p3 = New SQLiteParameter()
p3.DbType = DbType.Double
p3.Value = value.idUtilisateur

Return {getInsertParameters(value), p3}
End Function
End Class
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>104/156</b>

## 9.28 DAO\_HDroit

```
Imports System.Data.SQLite
```

```
Public Class DAO_HDroit
    Inherits AbstractDAO
    Implements IDAO_Histo(Of HistoDroits)
```

```
    Public Function dbGetAllStatutById(ByRef idObj As Integer) As
System.Collections.Generic.List(Of HistoDroits) Implements IDAO_Histo(Of
HistoDroits).dbGetAllStatutById
        'création de la liste
        Dim maListeObj As New List(Of HistoDroits)
```

```
    Try
        Dim reader As SQLiteDataReader = MyBase.DAOGetById(idObj, HDR_GETALL)

        'récupération du jeu de résultats
        While reader.Read()
            maListeObj.Add(New HistoDroits(reader(0).ToString,
                                           reader(1).ToString,
                                           reader(2).ToString,
                                           reader(3).ToString))
        End While
    End Try
```

```
        'libération des objets
        reader.Close()
    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_GETALL, HDR_ERR_TYPEOF), ex)
    End Try
    Return maListeObj
End Function
```

```
    Public Function dbGetLastStatutById(ByRef id As Integer) As HistoDroits Implements
IDAO_Histo(Of HistoDroits).dbGetLastStatutById
        'création de la liste
        Dim monObj As HistoDroits = Nothing
```


```
    Try
        'définition des paramètres à mettre à jour
        Dim p1 = New SQLiteParameter()
        p1.DbType = DbType.Double
        p1.Value = id

        Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, HDR_GETLAST)
        'récupération du jeu de résultats

        If reader.Read Then
            monObj = New HistoDroits(reader(0).ToString,
                                     reader(1).ToString,
                                     reader(2).ToString,
                                     reader(3).ToString)
        End If
    End Try
```

```
        'libération des objets
        reader.Close()
```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>105/156</b>

```

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_GETALL, HDR_ERR_TYPEOF), ex)
        End Try
        Return monObj
    End Function

    Public Sub dbInsert(ByRef value As HistoDroits) Implements IDAO_Histo(Of
HistoDroits).dbInsert
        Try
            MyBase.DAOInsert(HDR_INSERT, getInsertParameters(value))

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_INSERT, HDR_ERR_TYPEOF, value.ToString),
ex)
        End Try

    End Sub

    Public Sub dbUpdate(ByRef value As HistoDroits) Implements IDAO_Histo(Of
HistoDroits).dbUpdate
        Try
            MyBase.DAOUpdate(HDR_UPDATE, getUpdateParameters(value))

        Catch ex As Exception
            Throw New DAOException(String.Format(ERR_UPDATE, HDR_ERR_TYPEOF, value.ToString),
ex)
        End Try
    End Sub

    Public Function getInsertParameters(ByVal value As HistoDroits) As
System.Data.SQLite.SQLiteParameter() Implements IDAO_Histo(Of HistoDroits).getInsertParameters
        'définition des paramètres à mettre à jour
        Dim p1 = New SQLiteParameter()
        p1.DbType = DbType.String
        p1.Value = value.commentaire

        Dim p2 = New SQLiteParameter()
        p2.DbType = DbType.String
        p2.Value = value.getDate


        Dim p3 = New SQLiteParameter()
        p3.DbType = DbType.Double
        p3.Value = value.getIdUtilisateur

        Dim p4 = New SQLiteParameter()
        p4.DbType = DbType.Double
        p4.Value = value.getIdDroit
        Return {p1, p2, p3, p4}
    End Function

    Public Function getUpdateParameters(ByVal value As HistoDroits) As
System.Data.SQLite.SQLiteParameter() Implements IDAO_Histo(Of HistoDroits).getUpdateParameters
        'définition des paramètres à mettre à jour
        Dim p1 = New SQLiteParameter()
        p1.DbType = DbType.String
        p1.Value = value.commentaire

        Dim p2 = New SQLiteParameter()
        p2.DbType = DbType.Double

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>106/156</b>
--	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```
p2.Value = value.getIdUtilisateur
```

```
Dim p3 = New SQLiteParameter()  
p3.DbType = DbType.Double  
p3.Value = value.getIdDroit
```

```
Dim p4 = New SQLiteParameter()  
p4.DbType = DbType.String  
p4.Value = value.getDate  
Return {p1, p2, p3, p4}
```

```
End Function
```

```
End Class
```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>107/156</b>

## 9.29 DAO\_HVerif

Imports System.Data.SQLite

Public Class DAO\_HVerif

Inherits AbstractDAO

Implements IDAO\_Histo(Of HistoVerification)

Public Function dbGetAllStatutById(ByRef id As Integer) As  
System.Collections.Generic.List(Of HistoVerification) Implements IDAO\_Histo(Of  
HistoVerification).dbGetAllStatutById

'création de la liste

Dim maListeObj As New List(Of HistoVerification)

Try

Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, HVF\_GETALL)

'récupération du jeu de résultats

While reader.Read()

maListeObj.Add(New HistoVerification(reader(0).ToString,  
Convert.ToDateTime(reader(1).ToString),  
reader(2).ToString,  
reader(3).ToString))

End While

'libération des objets

reader.Close()

Catch ex As Exception

Throw New DAOException(String.Format(ERR\_GETALL, HVF\_ERR\_TYPEOF), ex)

End Try

Return maListeObj

End Function

Public Function dbGetLastStatutById(ByRef id As Integer) As HistoVerification Implements  
IDAO\_Histo(Of HistoVerification).dbGetLastStatutById

'création de la liste

Dim monObj As HistoVerification = Nothing

Try

Dim reader As SQLiteDataReader = MyBase.DAOGetById(id, HVF\_GETLAST)

'récupération du jeu de résultats

If reader.Read Then

monObj = New HistoVerification(reader(0).ToString,  
Convert.ToDateTime(reader(1).ToString),  
reader(2).ToString,  
reader(3).ToString)

End If

'libération des objets

reader.Close()


Catch ex As Exception

Throw New DAOException(String.Format(ERR\_GETALL, HVF\_ERR\_TYPEOF), ex)

End Try

Return monObj

End Function

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>108/156</b>

```

Public Sub dbInsert(ByRef value As HistoVerification) Implements IDAO_Histo(Of
HistoVerification).dbInsert
    Try
        MyBase.DAOInsert(HVF_INSERT, getInsertParameters(value))

    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_INSERT, HVF_ERR_TYPEOF, value.ToString),
ex)
    End Try
End Sub

Public Sub dbUpdate(ByRef value As HistoVerification) Implements IDAO_Histo(Of
HistoVerification).dbUpdate
    Dim nbLigneUpdate As Integer = 0

    Try

        MyBase.DAOUpdate(HVF_UPDATE, getUpdateParameters(value))

    Catch ex As Exception
        Throw New DAOException(String.Format(ERR_UPDATE, HVF_ERR_TYPEOF, value.ToString),
ex)
    End Try
End Sub

Public Function getInsertParameters(ByVal value As HistoVerification) As
System.Data.SQLite.SQLiteParameter() Implements IDAO_Histo(Of
HistoVerification).getInsertParameters
    'définition des paramètres à mettre à jour
    Dim p1 = New SQLiteParameter()
    p1.DbType = DbType.String
    p1.Value = value.commentaire

    Dim p2 = New SQLiteParameter()
    p2.DbType = DbType.String
    p2.Value = value.getDate


    Dim p3 = New SQLiteParameter()
    p3.DbType = DbType.Double
    p3.Value = value.getIdAuditrails

    Dim p4 = New SQLiteParameter()
    p4.DbType = DbType.Double
    p4.Value = value.getIdVerification
    Return {p1, p2, p3, p4}
End Function

Public Function getUpdateParameters(ByVal value As HistoVerification) As
System.Data.SQLite.SQLiteParameter() Implements IDAO_Histo(Of
HistoVerification).getUpdateParameters
    'définition des paramètres à mettre à jour
    Dim p1 = New SQLiteParameter()
    p1.DbType = DbType.String
    p1.Value = value.commentaire

    Dim p2 = New SQLiteParameter()
    p2.DbType = DbType.Double
    p2.Value = value.getIdAuditrails

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>109/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

Dim p3 = New SQLiteParameter()
p3.DbType = DbType.Double
p3.Value = value.getIdVerification

Dim p4 = New SQLiteParameter()
p4.DbType = DbType.String
p4.Value = value.getDate
Return {p1, p2, p3, p4}
End Function

```

```
End Class
```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>110/156</b>

### 9.30 DAOspecificTransaction

Imports System.Data.SQLite

Public Class DAOspecificTransaction  
Inherits AbstractDAO

```


Public Sub dbInsertATPrinter(ByRef arg_auditrails As auditrails, ByRef arg_impression As
Impression, ByRef arg_signets As List(Of Signets))

    'pour une transaction SQLite
    Using tr As SQLiteTransaction = DAOFactory.getConnexion.BeginTransaction
        Dim DAO_At As New DAOAuditrails
        Dim DAO_Imp As New DAOImpression
        Dim DAO_sig As New DAOSignet

        Dim cmd_AT As SQLiteCommand = DAOFactory.getConnexion.CreateCommand
        With cmd_AT
            Try
                'insertion dans la table audit trails
                .Transaction = tr
                .CommandText = AT_CREATE & DAO_LASTID
                .Parameters.AddRange(DAO_At.getInsertParameters(arg_auditrails))
                .Prepare()
                arg_auditrails.idAuditrails = .ExecuteScalar()
                'ajout du nouvel id à l'obj impression
                arg_impression.idAuditrails = arg_auditrails.idAuditrails
            Catch ex As Exception
                tr.Rollback()
                Throw New DAOException("erreur lors de l'insertion de l'audit trails", ex)
            End Try
            If arg_auditrails.idAuditrails <= 0 Then
                tr.Rollback()
                Throw New DAOException("le programme n'a pas pu récupérer l'ID de l'audit
trails")
            End If
        End With

        Dim cmd_IMP As SQLiteCommand = DAOFactory.getConnexion.CreateCommand
        With cmd_IMP
            Try
                'insertion dans la table audit trails
                .Transaction = tr
                .CommandText = IMP_CREATE & DAO_LASTID
                .Parameters.AddRange(DAO_Imp.getInsertParameters(arg_impression))
                .Prepare()
                arg_impression.idImpression = .ExecuteScalar()
            Catch ex As Exception
                tr.Rollback()
                Throw New DAOException("erreur lors de l'insertion de l'audit trails", ex)
            End Try
            If arg_impression.idImpression <= 0 Then
                tr.Rollback()
                Throw New DAOException("le programme n'a pas pu récupérer l'ID de
l'enregistrement des infos d'impression")
            End If
        End With
    End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>111/156</b>


```

'injection des signets
For Each signet As Signets In arg_signets
    Dim cmd_signet As SQLiteCommand = DAOFactory.getConnexion.CreateCommand
    With cmd_signet
        Try
            signet.idImpression = arg_impression.idImpression
            .CommandText = SGT_CREATE & DAO_LASTID
            .Parameters.AddRange(DAO_sig.getInsertParameters(signet))
            .Prepare()
            signet.idSignet = .ExecuteScalar

            If signet.idSignet <= 0 Then
                tr.Rollback()
                Throw New DAOException("le programme n'a pas pu récupérer l'ID de
l'enregistrement des signets d'impression")
            End If
        Catch ex As Exception
            tr.Rollback()
            Throw New DAOException("erreur lors de l'insertion des signets dans la
table signet", ex)
        End Try
    End With
Next
tr.Commit()
End Using
End Sub

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>112/156</b>

### 9.31 DAOViews

```

Public Class DAOViews
    Inherits AbstractDAO

    Private Const _ATPRINTER_ALL = "select * from view_ATPrinter"
    Private Const _ATPRINTER_EC = "select * from view_ATPrinter where id_verification < 2"

    Private Const _ATPRINTER_SIGNET = "select * from view_ATPrinter_Signets where id_auditrails
= ?"

    Private Const _ATAUTRE_ALL = "select * from view_AT"
    Private Const _ATAUTRE_EC = "select * from view_AT where id_verification < 2"

    Private Const _LISTEUSER = "select * from view_ListeUtilisateur"
    Private Const _HISTO_USER = "select * from view_ListeDroitUtilisateur where id_utilisateur =
?"
    Private Const _HISTO_VERIF = "select * from view_HistoVerif where id_auditrails = ?"

    ''' <summary>
    ''' Défini toutes les vues que le DataGridView est susceptible d'afficher
    ''' Elle devront être dans le table name du DGV
    ''' </summary>
    Public Enum views As Byte
        PasDeVue = 0
        AT_Printer_ALL = 1
        User_ALL = 2
        AT_IEA_ALL = 3
        AT_Printer_Signet = 4
        Histo_DroitUser = 5
        Histo_VerifAT = 6
        AT_Printer_Encours = 7
        AT_IEA_Encours = 8
    End Enum

    Public Enum reqById As Byte
        GetSignet = 0
        GetHistoDroit = 1
        GetHistoVerif = 2
    End Enum


    Private Sub createDataTable(ByRef maDataTable As DataTable, ByVal monReader As
SQLite.SQLiteDataReader)
        If Not IsNothing(monReader) Then

            'Faire une vérif si colonnes identique car sinon bug....
            For i = 1 To monReader.FieldCount
                maDataTable.Columns.Add(monReader.GetName(i - 1))
            Next

            While (monReader.Read())
                Dim mesRow As New List(Of Object)
                For i = 1 To monReader.FieldCount
                    mesRow.Add(monReader(i - 1).ToString)
                Next
                maDataTable.Rows.Add(mesRow.ToArray)
            End While
        End If
    End Sub

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>113/156</b>

End Sub

```
Public Function getById(ByVal id As Integer, ByVal requete As reqById, ByVal nomTable As String) As DataTable
```

```
    Using maDataTable As New DataTable(nomTable)
```

```
        Dim reader As SQLite.SQLiteDataReader = Nothing
```

```
        If requete = reqById.GetSignet Then
```

```
            reader = MyBase.DAOGetById(id, _ATPRINTER_SIGNET)
```

```
        ElseIf requete = reqById.GetHistoDroit Then
```

```
            reader = MyBase.DAOGetById(id, _HISTO_USER)
```

```
        ElseIf requete = reqById.GetHistoVerif Then
```

```
            reader = MyBase.DAOGetById(id, _HISTO_VERIF)
```

```
        End If
```

```
        createDataTable(maDataTable, reader)
```

```
    Return maDataTable
```

```
End Using
```

```
End Function
```

```
Public Function getAll(ByVal maVue As views, ByVal nomTable As String) As DataTable
```

```
    Using maDataTable As New DataTable(nomTable)
```

```
        Dim reader As SQLite.SQLiteDataReader
```

```
        Select Case maVue
```

```
            Case views.AT_Printer_ALL
```

```
                reader = MyBase.DAOGetAll(_ATPRINTER_ALL)
```

```
            Case views.User_ALL
```

```
                reader = MyBase.DAOGetAll(_LISTEUSER)
```

```
            Case views.AT_IEA_ALL
```

```
                reader = MyBase.DAOGetAll(_ATAUTRE_ALL)
```

```
            Case views.AT_Printer_Encours
```

```
                reader = MyBase.DAOGetAll(_ATPRINTER_EC)
```

```
            Case views.AT_IEA_Encours
```

```
                reader = MyBase.DAOGetAll(_ATAUTRE_EC)
```

```
            Case Else
```

```
                reader = Nothing
```

```
        End Select
```


```
        createDataTable(maDataTable, reader)
```

```
    Return maDataTable
```

```
End Using
```

```
End Function
```

```
End Class
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>114/156</b>

## 9.32 DAORequetes

### Module DAORequetes

```

'Fonctions du CRUD Utilisateur
Public Const USR_CREATE = "PRAGMA foreign_keys = ON;INSERT INTO utilisateur values (null,?)"
Public Const USR_READ = "SELECT * FROM utilisateur"
Public Const USR_READBYID = "SELECT * FROM utilisateur where id_utilisateur = ? "
Public Const USR_READBYNAME = "SELECT * FROM utilisateur where nom_utilisateur = ? "
Public Const USR_UPDATE = "UPDATE utilisateur SET nom_utilisateur=? WHERE id_utilisateur=?"
Public Const USR_DELETE = "DELETE FROM utilisateur WHERE id_utilisateur=?"
Public Const USR_ERR_TYPEOF = "Utilisateur"

'Fonctions du CRUD Signets
Public Const SGT_CREATE = "INSERT INTO signets values (null,?,?,?,?)"
Public Const SGT_READ = "SELECT * FROM signets"
Public Const SGT_READBYID = "SELECT * FROM signets WHERE id_signet=?"
Public Const SGT_UPDATE = "UPDATE signets SET clef_signet=?, valeur_signet=?, code_signet=?,
id_impression=? WHERE id_signet=?"
Public Const SGT_DELETE = "DELETE FROM signets WHERE id_signet=?"
Public Const SGT_ERR_TYPEOF = "Signets"

'Fonctions du CRUD Impression
Public Const IMP_CREATE = "INSERT INTO impression values (null,?,?,?,?)"
Public Const IMP_READ = "SELECT * FROM impression"
Public Const IMP_READBYID = "SELECT * FROM impression WHERE id_impression=?"
Public Const IMP_UPDATE = "UPDATE impression SET lot_impression=?,
nomimprimante_impression=?, pages_impression=?, id_auditrails=? WHERE id_impression=?"
Public Const IMP_DELETE = "DELETE FROM impression WHERE id_impression=?"
Public Const IMP_ERR_TYPEOF = "Impression"


'Fonctions du CRUD Auditrails
Public Const AT_CREATE = "INSERT INTO auditrails values (null,?,?,?,?,?,?)"
Public Const AT_READ = "SELECT * FROM auditrails"
Public Const AT_READBYID = "SELECT * FROM auditrails WHERE id_auditrails=?"
Public Const AT_UPDATE = "UPDATE auditrails SET nomfichier_auditrails=?,
commentaire_auditrails=?, infosystem_auditrails=?, date_auditrails=?, id_utilisateur=? ,
id_operation=? WHERE id_auditrails=?"
Public Const AT_DELETE = "DELETE FROM auditrails WHERE id_auditrails=?"
Public Const AT_ERR_TYPEOF = "Auditrails"

Public Const HVF_GETALL = "select * from histo_verification where id_auditrails = ?"
Public Const HVF_GETLAST = "select * from histo_verification where id_auditrails = ? order
by date_histo_verification DESC limit 1"
Public Const HVF_INSERT = "INSERT into histo_verification values (?, ?, ?, ?)"
Public Const HVF_UPDATE = "UPDATE histo_verification set commentaire_histo_verification = ?
where id_auditrails = ? and id_verification = ? and date_histo_verification = ?"
Public Const HVF_ERR_TYPEOF = "HistoVérification"

Public Const HDR_GETALL = "select * from histo_droits where id_utilisateur = ?"
Public Const HDR_GETLAST = "select * from histo_droits where id_utilisateur = ? order by
date_histo_droit DESC limit 1"
Public Const HDR_INSERT = "INSERT into histo_droits values (?, ?, ?, ?)"
Public Const HDR_UPDATE = "UPDATE histo_droits set commentaire_histo_droit = ? where
id_utilisateur = ? and id_droit = ? and date_histo_droit = ?"
Public Const HDR_ERR_TYPEOF = "HistoDroit"
Public Const ATP_ERR_TYPEOF = "AuditrailsImpression"

```

End Module

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>115/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

### 9.33 DAOService

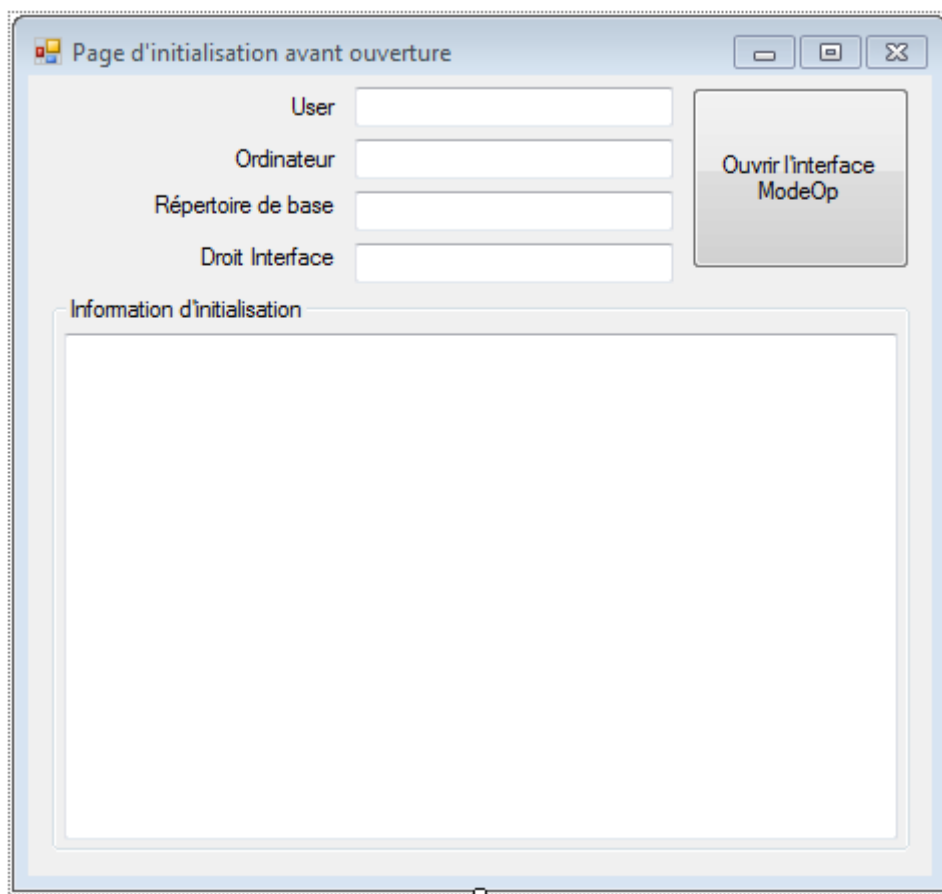
```

Module DAOService
    ''' <summary>
    ''' Renvoie la date au format Text SQLite
    ''' </summary>
    Public Function DateToString(ByVal maDate As DateTime)
        Dim maDateStr As New System.Text.StringBuilder
        With maDateStr
            .Append(Format(maDate.Year, "0000"))
            .Append("-")
            .Append(Format(maDate.Month, "00"))
            .Append("-")
            .Append(Format(maDate.Day, "00"))
            .Append(" ")
            .Append(Format(maDate.Hour, "00"))
            .Append(":")
            .Append(Format(maDate.Minute, "00"))
            .Append(":")
            .Append(Format(maDate.Second, "00"))
        End With
        Return maDateStr.ToString
    End Function
End Module

```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>116/156</b>

### 9.34 Initialisation




```

<Assembly: Reflection.AssemblyVersion("2.0.1.8")>
<Assembly: Reflection.AssemblyFileVersion("2.0.1.8")>
<Assembly: Reflection.AssemblyInformationalVersion("2300 FS 01 F")>
'Version
'Chiffre 1 : Nouvelle version logiciel (changement plateforme, de langage...)
'Chiffre 2 : Changement majeur pouvant entraîner une incompatibilité
'Chiffre 3 : Changement majeur sans risque d'incompatibilité
'Chiffre 4 : Changement mineur sans impact (résolution bug, etc...)
Public Class Initialisation
    'à l'initialisation récupère les données utilisateurs (nom windows)
    Public __User As UserConfig
    Private _BTVisible As Boolean = True

    Private Const _BDD_NOUSER = "l'utilisateur {0} n'est pas présent dans la base de donnée"
    Private Const _BDD_NOFILE = "La base de donnée n'a pas été trouvée"
    Private Const _BDD_NOCONN = "Pas de connexion avec la base de donnée"
    Private Const _BDD_ERRCONN = "Erreur lors de la connexion à la base de donnée"
    Private Const _BDD_GETDROIT = "La BDD renvoie les droits {0} pour l'utilisateur {1}"
    Private Const _INI_CHECKFOLDER = "Erreur lors de la configuration utilisateur"
    Private Const _INI_TEST = "Test lecture / écriture des dossiers de production"
    Private Const _INI_NOFOLDERNAME = "N/A"
    Private Const _INI_OUV = "Ouverture de l'interface"
    Private Const _INI_FERM = "L'interface à été verrouillée"

#Region "Constructeur"

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>117/156</b>

```

Sub New()

    InitializeComponent()

    For Each fichier As String In {"InterfaceModeOp2.ini", "VBTools.dll",
"System.Data.SQLite.dll"}
        If Not System.IO.File.Exists(fichier) Then
            MessageBox.Show(String.Format("Le fichier {0} est introuvable, l'application va
fermer", fichier), "Erreur fatale", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End
    End If
Next

    ConfigurationInterface()

#If DEBUG Then
    'si mode debug, change la config déterminé par la méthode init()
    DebugConfiguration(0, Droits.AdminDvlp, "222222")
#End If

    'les infos sont ajoutées à la page
    ResumeInfo()

End Sub
#End Region

#Region "Configuration"
Private Sub ConfigurationInterface()
    Try
        'détermine les droits de l'utilisateur sur les dossiers de prod
        'en réalisant des tests de lectures écritures sur chaque dossiers
        _User = New UserConfig
    Catch ex As Exception
        _BTVisible = False
        Info(_INI_CHECKFOLDER & vbNewLine & ex.Message, True)
    Exit Sub
End Try

    Try
        'vérification de la présence de la base de données
        'le chemin est défini dans fichier ini
        If Not
System.IO.File.Exists(Configuration.GetInstance.GetValueFromKey(Singleton.DBFOLDER)) Then
            Throw New Exception(_BDD_NOFILE)
        End If
        If IsNothing(DAOFactory.getConnexion()) Then
            Throw New Exception(_BDD_NOCONN)
        End If
        If Singleton.getModeProtection = -1 Then
            Throw New Exception(_BDD_NOCONN)
        End If
    Catch ex As Exception
        _BTVisible = False
        Info(_BDD_ERRCONN & vbNewLine & ex.Message, True)
    Exit Sub
End Try

    Try
        'détermination des droits via bdd

```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>118/156</b>

```


        Dim monUser As Utilisateur =
DAOFactory.getUserUtilisateur.dbGetByName(__User.getUserName)
        If IsNothing(monUser) Then
            'l'utilisateur n'est pas présent dans la bdd
            __User.setDroitUser = Droits.Guest
            Info(String.Format(_BDD_NOUSER, __User.getUserName))
        Else
            'l'utilisateur est présent et à des droits
            __User.setUserId = monUser.idUtilisateur
            __User.setDroitUser =
DAOFactory.getHistoDroit.dbGetLastStatutById(monUser.idUtilisateur).getIdDroit
        End If
        Info(String.Format(_BDD_GETDROIT, __User.getDroitBDD, __User.getUserName))
    Catch ex As Exception
        _BTVisible = False
        Info("" & vbNewLine & ex.Message, True)
        Exit Sub
    End Try
End Sub
''' <summary>
''' 1 : DroitBDD
''' 2 : envTravail
''' les deux 1+2 = 3
''' </summary>
''' <param name="level"></param>
''' <remarks></remarks>
Private Sub DebugConfiguration(ByVal level As Byte, ByVal bdd As Droits, ByVal envT As
String)
    Info()
    If level >= 2 Then
        level -= 2
        'son env. de prod est
        __User.setArchDossProd = envT
        Dim env As New System.Text.StringBuilder()
        For i = 1 To 6
            With env
                .Append("[")
                .Append(__User.getArchDossProd.getAccess(i))
                .Append("]")
            End With
        Next
        Info("Mode Debug force l'environnement de travail à " & env.ToString)
    End If

    If level >= 1 Then
        level -= 1
        'le profil utilisateur est défini sur
        __User.setDroitUser = bdd
        Info("Mode Debug force les droits utilisateur à " & __User.getDroitBDD.ToString)
    End If

End Sub
Private Sub ResumeInfo()
    TXT_Username.Text = __User.getUserName
    TXT_PC.Text = __User.getPC
    TXT_RepBase.Text = Configuration.GetInstance.GetValueFromKey(Outils.INI_KEY_REPBASE)
    TXT_Droit.Text = __User.getDroitDetermine.ToString

    Dim ListeDossier = [Enum].GetValues(GetType(Outils.DossierProd))

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>119/156</b>

```

Info()
For Each dossier As Outils.DossierProd In ListeDossier
    Dim infostr As New System.Text.StringBuilder
    Dim nom As String
    With infostr
        .Append(dossier.ToString)
        .Append(" : ")
        nom = Configuration.GetInstance.GetValueFromKey(dossier.ToString)
        .Append(If(nom = "", _INI_NOFOLDERNAME, nom))
        .Append(" [")
        .Append(__User.getArchDossProd.getAccess(dossier).ToString)
        .Append("]")

        Info(.ToString)

    End With
Next
Info(_INI_TEST)

If __User.getDroitDetermine <= Droits.NoUse Then
    _BTVisible = False
End If

If Singleton.getModeProtection = -1 Then
    _BTVisible = False
End If

Me.BT_Open.Visible = _BTVisible
Info()
Info(If(_BTVisible, _INI_OUV, _INI_FERM))


'l'interface prévient l'adminDVLP que la bdd n'est pas protégé par un mot de passe
If Singleton.getModeProtection = 0 And __User.getDroitDetermine = Droits.AdminDvlp Then
    Info("-----")
    Info("--")
    Info("-- ATTENTION : BDD EN PROTECTION DESACTIVEE --")
    Info("-----")
    Info("--")
End If

End Sub

Private Sub Info(Optional ByVal txt As String = Nothing, Optional ByVal sautLigne As Boolean
= False)
    Dim mesInfos As New System.Text.StringBuilder()
    With mesInfos
        If sautLigne Then .AppendLine()
        If Not IsNothing(txt) Then
            .Append(txt)
        End If
        If sautLigne Then .AppendLine()
        .AppendLine()
        .Append(TXT_InfoINI.Text)
    End With
    TXT_InfoINI.Text = mesInfos.ToString
End Sub
#End Region

#Region "Evènements sur boutons"
''' <summary>

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>120/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

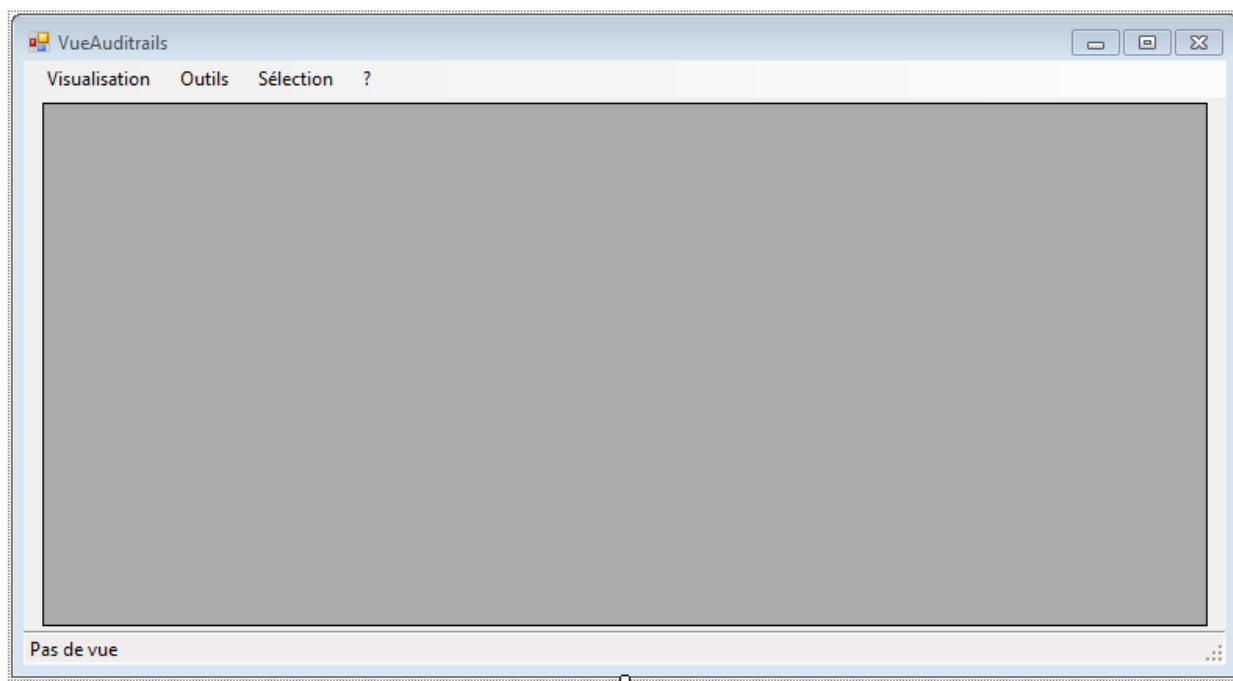
''' Ouverture de la page principale de l'interface
''' </summary>
''' <remarks></remarks>
Private Sub BT_Open_Click() Handles BT_Open.Click
    Me.BT_Open.Visible = False
    Me.Visible = False
    vuePrincipale.getVP.Show()
End Sub
#End Region
End Class

```



VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>121/156</b>

### 9.35 VueAuditrails



```
Imports VBTools.GestionDataGridView
```

```
Public Class VueAuditrails
```

```
Private Const AFFICHAGE_VUE = "Visualisation de la vue : {0}"
Private Const SELECTION = "Sélection de {0} n°{1}"
Private Const CHG_VERIFICATION = "Vérification de {0} : {1}"
Private Const CHG_DROITS = "{0} change les droits : {1}"
Private Const IDENT_IMPRESSION = "impression par {0} le {1}"
```

```
Private Const VUE_ATPRINTER = "Audit trails d'impression pour mode opératoire"
Private Const VUE_ATPRINTER_SIGNETS = "Signet du bon pour utilisation n°{0} pour mode opératoire"
Private Const VUE_HISTODROITSUSER = "Historique des droits de l'utilisateur n°{0}"
Private Const VUE_HISTOVERIFICATION = "Historique des vérification de l'audit trails n°{0}"
Private Const VUE_ATIEA = "Audit trails d'Importation/Exportation/Archivage"
Private Const VUE_USERS = "Liste des utilisateurs"
Private Const VUE_NOVIEW = "Pas de Vue"
```

```
Private _DGVMMainSize As Size
Private _canResize = False
Private _Vue_Encours As DAOViews.views
Private _Vue_Precedente As DAOViews.views = DAOViews.views.PasDeVue
```


```
#Region "Constructeur"
```

```
Sub New()
```

```
' Cet appel est requis par le concepteur.
InitializeComponent()
```

```
With DGV_Main
```

```
'l'utilisateur ne peux pas supprimer/ajouter de données
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>122/156</b>

```

.AllowUserToDeleteRows = False
.AllowUserToAddRows = False
.MultiSelect = True
.ReadOnly = True
.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill


_DGVMainSize = New Size(Me.Width - .Width, Me.Height - .Height)
_canResize = True
TSMI_TOOLS_PRINTSEL.Visible = False
TSMI_TOOLS_VOIREC.Visible = False
changerVue(DAOViews.views.PasDeVue)
End With

End Sub
#End Region

#Region "Evènement Menu"
Private Sub Vue_ATImpression(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TSMI_ATPrinter.Click
changerVue(DAOViews.views.AT_Printer_Encours)
End Sub
Private Sub Vue_Rien(ByVal sender As System.Object, ByVal e As System.EventArgs)
changerVue(DAOViews.views.PasDeVue)
End Sub
Private Sub TSMI_LISTEUSER_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TSMI_LISTEUSER.Click
changerVue(DAOViews.views.User_ALL)
End Sub
Private Sub TSMI_ATAutre_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TSMI_ATAutre.Click
changerVue(DAOViews.views.AT_IEA_Encours)
End Sub
Private Sub Imprimer_DGV(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
TSMI_TOOLS_PRINTTAB.Click
Dim DGV As New PrintDataGridView(DGV_Main)
DGV.textePremierePage = String.Format(IDENT_IMPRESSION,
Initialisation.__User.getUserName, Now)
DGV.Impression(PrintDataGridView.Affichage.All)
End Sub
Private Sub TSMI_TOOLS_PRINTSEL_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_TOOLS_PRINTSEL.Click
Dim DGV As New PrintDataGridView(DGV_Main)
DGV.textePremierePage = String.Format(IDENT_IMPRESSION,
Initialisation.__User.getUserName, Now)
DGV.Impression(PrintDataGridView.Affichage.Selection)
End Sub
Private Sub VuePrécédenteToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_TOOLS_VUEOLD.Click
changerVue(_Vue_Precedente, True)
End Sub
#End Region

#Region "Gestion du menu Selection"
Private Sub ConfigMenuSelection(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles DGV_Main.CellClick
'DGV_Main.Rows.Item(e.RowIndex).Cells(e.ColumnIndex).Value
'sélection d'une ligne complète
If e.ColumnIndex = -1 And e.RowIndex > -1 Then
TSMI_TOOLS_PRINTSEL.Visible = True

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>123/156</b>

```

TSMI_Selection.Visible = True


Select Case _Vue_Encours
    Case DAOViews.views.AT_Printer_Encours
        gestionMenuSelection(True, True, True, False, True)
        TSMI_Selection.Text = String.Format(SELECTION, "l'audit trails
d'impression", DGV_Main.Rows.Item(e.RowIndex).Cells(0).Value)
    Case DAOViews.views.AT_Printer_ALL
        gestionMenuSelection(True, True, True, False, True, True)
        TSMI_Selection.Text = String.Format(SELECTION, "l'audit trails
d'impression", DGV_Main.Rows.Item(e.RowIndex).Cells(0).Value)
    Case DAOViews.views.AT_IEA_ALL
        gestionMenuSelection(True, True, False, False, True, True)
        TSMI_Selection.Text = String.Format(SELECTION, "l'audit trails",
DGV_Main.Rows.Item(e.RowIndex).Cells(0).Value)
    Case DAOViews.views.AT_IEA_Encours
        gestionMenuSelection(True, True, False, False, True)
        TSMI_Selection.Text = String.Format(SELECTION, "l'audit trails",
DGV_Main.Rows.Item(e.RowIndex).Cells(0).Value)
    Case DAOViews.views.User_ALL
        gestionMenuSelection(True, True, False, True, False)
        TSMI_Selection.Text = String.Format(SELECTION, "l'utilisateur",
DGV_Main.Rows.Item(e.RowIndex).Cells(0).Value)
    Case Else
        TSMI_Selection.Visible = False
End Select
Else
    TSMI_Selection.Visible = False
    TSMI_TOOLS_PRINTSEL.Visible = False
End If
End Sub
#End Region

#Region "Redimensionnement de la fenetre"
Private Sub VueAudittrails_Resize(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Resize
    If _canResize Then
        'Gestion du DatagridView1
        With DGV_Main
            .Size = New Size(Me.Width - _DGVMainSize.Width, Me.Height - _DGVMainSize.Height)
        End With
    End If
End Sub
#End Region

#Region "Méthode Interne"
Private Sub changerVue(ByVal maVue As DAOViews.views, Optional ByVal reset As Boolean =
False)
    If reset Then
        _Vue_Precedente = DAOViews.views.PasDeVue
        _Vue_Encours = maVue
    Else
        _Vue_Precedente = _Vue_Encours
        _Vue_Encours = maVue
        Me.TSMI_TOOLS_VOIREC.Visible = False
    End If

    Me.TSMI_TOOLS_PRINTSEL.Visible = False
    TSMI_Selection.Visible = False

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>124/156</b>

```


Select Case maVue
    Case DAOViews.views.AT_Printer_ALL
        DGV_Main.DataSource = DAOFactory.getViews.getAll(maVue, VUE_ATPRINTER)
    Case DAOViews.views.AT_IEA_ALL
        DGV_Main.DataSource = DAOFactory.getViews.getAll(maVue, VUE_ATIEA)
    Case DAOViews.views.User_ALL
        DGV_Main.DataSource = DAOFactory.getViews.getAll(maVue, VUE_USERS)
    Case DAOViews.views.AT_Printer_Signet
        Dim _id As Integer = DGV_Main.SelectedCells(0).Value
        DGV_Main.DataSource = DAOFactory.getViews.getById(_id,
DAOViews.reqById.GetSignet, String.Format(VUE_ATPRINTER_SIGNETS, _id))
    Case DAOViews.views.Histo_DroitUser
        Dim _id As Integer = DGV_Main.SelectedCells(0).Value
        DGV_Main.DataSource = DAOFactory.getViews.getById(_id,
DAOViews.reqById.GetHistoDroit, String.Format(VUE_HISTODROITSUSER, _id))
    Case DAOViews.views.Histo_VerifAT
        Dim _id As Integer = DGV_Main.SelectedCells(0).Value
        DGV_Main.DataSource = DAOFactory.getViews.getById(_id,
DAOViews.reqById.GetHistoVerif, String.Format(VUE_HISTOVERIFICATION, _id))
    Case DAOViews.views.AT_Printer_Encours
        DGV_Main.DataSource = DAOFactory.getViews.getAll(maVue, VUE_ATPRINTER & " -->En
cours de vérification")
    Case DAOViews.views.AT_IEA_Encours
        DGV_Main.DataSource = DAOFactory.getViews.getAll(maVue, VUE_ATIEA & " -->En
cours de vérification")
    Case Else
        DGV_Main.DataSource = Nothing
End Select

If maVue <> DAOViews.views.PasDeVue Then
    'cache les colonnes contenant l'id avec le vrai nom de colonne
    'ID PK
    DGV_Main.Columns(0).Visible = False
    'ID FK
    DGV_Main.Columns(1).Visible = False
End If

Me.lbl_Vue.Text = String.Format(AFFICHAGE_VUE,
                                If(IsNothing(DGV_Main.DataSource),
                                    VUE_NOVIEW,
                                    DGV_Main.DataSource.ToString))

If _Vue_Precedente = DAOViews.views.PasDeVue Then
    Me.TSMI_TOOLS_VUEOLD.Visible = False
Else
    Me.TSMI_TOOLS_VUEOLD.Visible = True
End If
If _Vue_Encours = DAOViews.views.AT_IEA_Encours Or _Vue_Encours =
DAOViews.views.AT_Printer_Encours Then
    Me.TSMI_TOOLS_VOIREC.Visible = True
End If
End Sub
Private Sub gestionMenuSelection(ByVal commentaire As Boolean,
                                ByVal historique As Boolean,
                                ByVal signets As Boolean,
                                ByVal chgDroit As Boolean,
                                ByVal chgVerif As Boolean,
                                Optional ByVal voirEC As Boolean = False)

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>125/156</b>

```

TSMI_Selection_Commentaire.Visible = commentaire
TSMI_Selection_Historique.Visible = historique
TSMI_Selection_Signets.Visible = signets
TSMI_Selection_ChangementDroit.Visible = If(Initialisation.__User.getDroitDetermine =
Droits.UserAQ, False, chgDroit)
TSMI_Selection_Verification.Visible = chgVerif
TSMI_TOOLS_VOIREC.Visible = voirEC
End Sub
#End Region

#Region "Gestion du tri"
Private Sub DGV_Main_CellDoubleClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles DGV_Main.CellDoubleClick
If e.ColumnIndex > -1 And e.RowIndex > -1 Then
For Each element As DataGridViewRow In DGV_Main.Rows
If element.Cells(e.ColumnIndex).Value() <>
DGV_Main.Rows.Item(e.RowIndex).Cells(e.ColumnIndex).Value Then
element.Visible = False
End If
Next
End If
End Sub
#End Region

#Region "Changement Statut de vérification"
Private Sub SetStatut(ByVal newStatut As Verification)
Dim message As String = String.Empty

message = InputBox("Indiquer ici les raisons du changement", "Changement du statut de
vérification")

If Not String.Empty = message Then
DAOFactory.getHistoVerification.dbInsert(New
HistoVerification(String.Format(CHG_VERIFICATION, Environment.UserName.ToUpper, message), Now,
DGV_Main.SelectedCells(0).Value, newStatut))
Else
MessageBox.Show("Annulé")
End If
changerVue(_Vue_Encours)
End Sub
Private Sub TSMI_Selection_Verification_AVerif_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TSMI_Selection_Verification_AVerif.Click
SetStatut(VERIFICATION.NoVerif)
End Sub
Private Sub TSMI_Selection_Verification_EnCours_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TSMI_Selection_Verification_EnCours.Click
SetStatut(Verification.EnCours)
End Sub
Private Sub TSMI_Selection_Verification_VerifOK_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TSMI_Selection_Verification_VerifOK.Click
SetStatut(Verification.VerifOK)
End Sub
#End Region

#Region "Changement de statut Droit utilisateurs"
Private Sub setDroitUser(ByVal drt As Droits)
Dim message As String = String.Empty

```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>126/156</b>

```


message = InputBox("Indiquer ici les raisons du changement", "Changement des droits
utilisateurs")

If Not String.Empty = message Then
    DAOFactory.getHistoDroit.dbInsert(New HistoDroits(String.Format(CHG_DROITS,
Environment.UserName.ToUpper, message), Now, DGV_Main.SelectedCells(0).Value, drt))
Else
    MessageBox.Show("Annulé")
End If
changerVue(_Vue_Encours)
End Sub
Private Sub TSMI_Selection_ChangementDroit_NoUse_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TSMI_Selection_ChangementDroit_NoUse.Click
    setDroitUser(Droits.NoUse)
End Sub
Private Sub TSMI_Selection_ChangementDroit_Guest_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TSMI_Selection_ChangementDroit_Guest.Click
    setDroitUser(Droits.Guest)
End Sub
Private Sub TSMI_Selection_ChangementDroit_User_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TSMI_Selection_ChangementDroit_User.Click
    setDroitUser(Droits.User)
End Sub
Private Sub TSMI_Selection_ChangementDroit_KeyUser_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Selection_ChangementDroit_KeyUser.Click
    setDroitUser(Droits.KeyUser)
End Sub
Private Sub TSMI_Selection_ChangementDroit_UserAQ_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TSMI_Selection_ChangementDroit_UserAQ.Click
    setDroitUser(Droits.UserAQ)
End Sub
Private Sub TSMI_Selection_ChangementDroit_AdminAQ_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Selection_ChangementDroit_AdminAQ.Click
    setDroitUser(Droits.AdminAQ)
End Sub
Private Sub TSMI_Selection_ChangementDroit_AdminDvlp_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Selection_ChangementDroit_AdminDvlp.Click
    setDroitUser(Droits.AdminDvlp)
End Sub
#End Region

#Region "Affichage des signets"
Private Sub TSMI_Selection_Signets_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Selection_Signets.Click
    changerVue(DAOViews.views.AT_Printer_Signet)
End Sub
#End Region

#Region "Gestion des historiques"
Private Sub TSMI_Selection_Historique_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Selection_Historique.Click
    If _Vue_Encours = DAOViews.views.User_ALL Then
        changerVue(DAOViews.views.Histo_DroitUser)
    ElseIf _Vue_Encours = DAOViews.views.AT_Printer_ALL Then
        changerVue(DAOViews.views.Histo_VerifAT)
    ElseIf _Vue_Encours = DAOViews.views.AT_IEA_ALL Then
        changerVue(DAOViews.views.Histo_VerifAT)
    ElseIf _Vue_Encours = DAOViews.views.AT_Printer_Encours Then
        changerVue(DAOViews.views.Histo_VerifAT)
    End If
End Sub

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>127/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------


```

ElseIf _Vue_Encours = DAOViews.views.AT_IEA_Encours Then
    changerVue(DAOViews.views.Histo_VerifAT)
End If
End Sub
#End Region

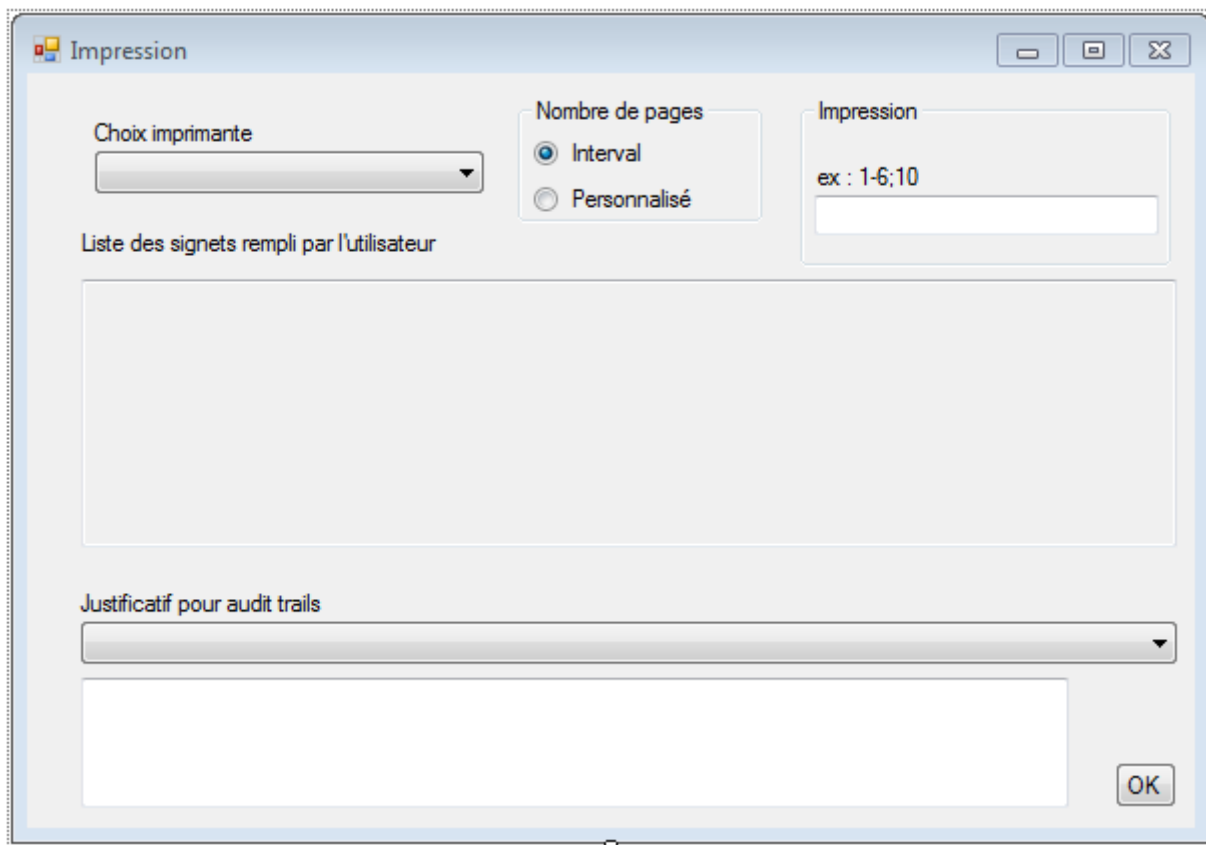
Private Sub TSMI_TOOLS_VOIREC_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_TOOLS_VOIREC.Click
    TSMI_TOOLS_VOIREC.Visible = False
    If _Vue_Encours = DAOViews.views.AT_Printer_Encours Then
        changerVue(DAOViews.views.AT_Printer_ALL)
    ElseIf _Vue_Encours = DAOViews.views.AT_IEA_Encours Then
        changerVue(DAOViews.views.AT_IEA_ALL)
    End If
End Sub

Private Sub ToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ToolStripMenuItem1.Click
    MessageBox.Show("Version SQLite : " & Singleton.GetInstance.ServerVersion, "A
Propos...", MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub
End Class

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>128/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

### 9.36 VueImpression



```
Imports PS = System.Drawing.Printing
```

```
Public Class VueImpression
    Private Const _AT_NBCAR_MIN As Byte = 20
    Private _PrinterProperty As PS.PrinterSettings
    Private _NomPrinter As String = Nothing
    Private _AuditTrails As String = "noAT"
    Private _ZoneImpression As String = Nothing
    Private _PageToPrint As New List(Of Integer)
    Private _FormulaireValid As Boolean = False
    Private _pageMax As Integer

    Private Const _PAGES = "1-{0}"

    #Region "Property"
    Public ReadOnly Property getNomPrinter As String
        Get
            Return _NomPrinter
        End Get
    End Property
    Public ReadOnly Property getAuditTrails As String
        Get
            Return _AuditTrails
        End Get
    End Property
    Public ReadOnly Property getPageAImprimer As List(Of Integer)
        Get
            Return _PageToPrint
        End Get
    End Property
End Class
```



VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>129/156</b>

```

        End Get
    End Property
    Public ReadOnly Property isValidForPrinting As Boolean
        Get
            Return _FormulaireValid
        End Get
    End Property
#End Region

#Region "Constructeur"
    Sub New(ByVal ListeSignet As List(Of Signets), ByVal ListePhraseAT As List(Of String), ByVal
    PageMax As Integer)

        ' Cet appel est requis par le concepteur.
        InitializeComponent()

        _pageMax = PageMax


        ' Ajoutez une initialisation quelconque après l'appel InitializeComponent().
        DefinirPageMax(PageMax)
        ChercheImprimantes()
        ChargeListePhraseAT(ListePhraseAT)
        DescriptionSignet(ListeSignet)

        _ZoneImpression = String.Format(_PAGES, _pageMax)

        TXT_PrintZone.Text = _ZoneImpression
    End Sub
    Private Sub DefinirPageMax(ByVal PageMax As Integer)
        NUD_2.Maximum = PageMax
        NUD_2.Minimum = 1
        NUD_1.Maximum = PageMax
        NUD_1.Minimum = 1
        NUD_2.Value = PageMax
        NUD_1.Value = 1
    End Sub
    Private Sub ChercheImprimantes()
        Dim printers As System.Drawing.Printing.PrinterSettings.StringCollection =
System.Drawing.Printing.PrinterSettings.InstalledPrinters()
        _PrinterProperty = New Printing.PrinterSettings

        Me.CB_ListeImp.Items.Clear()
        For x As Integer = 0 To printers.Count - 1
            _PrinterProperty.PrinterName = printers(x).ToString
            'récupère uniquement l'imprimante par défaut
            If _PrinterProperty.IsValid Then
                If _PrinterProperty.IsDefaultPrinter Then Me.CB_ListeImp.Items.Add(printers(x))
            End If
        Next
    End Sub
    Private Sub ChargeListePhraseAT(ByVal value As List(Of String))
        CB_AT.Items.Clear()
        For Each element As String In value
            CB_AT.Items.Add(element)
        Next
        CB_AT.Items.Add("Autre : ")
        TXT_AT.Visible = False
    End Sub
#End Region

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>130/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

#Region "Méthodes publiques"
Public Sub DescriptionSignet(ByVal ListeSignet As List(Of Signets))
    TXT_Signets.Clear()
    For Each Str As Signets In ListeSignet
        TXT_Signets.Text += Str.getClefSignet & "=" & Str.getValeurSignet & vbNewLine
    Next
End Sub
#End Region

#Region "Méthodes internes"
Private Function isOKToPrint() As Boolean
    'Une imprimante est choisi
    If IsNothing(_NomPrinter) Then
        MessageBox.Show("Veuillez choisir une imprimante", "impression",
        MessageBoxButtons.OK, MessageBoxIcon.Information)
        Return False
    End If

    'Audit trails renseigné
    If _AuditTrails.Length < _AT_NBCAR_MIN Then
        MessageBox.Show(String.Format("Veuillez renseigner l'audit trails avec au moins {0}
caractères", _AT_NBCAR_MIN), "impression", MessageBoxButtons.OK, MessageBoxIcon.Information)
        Return False
    End If


    'Zone d'impression
    Return canPrint()
End Function
Private Function canPrint() As Boolean

    Try
        'suppression des espaces blanc
        _ZoneImpression = _ZoneImpression.Replace(" ", "")
        Dim zone As New List(Of String)

        'sépare en fonction des sections ;
        If _ZoneImpression.Contains(";") Then
            zone = _ZoneImpression.Split(";").ToList
        Else
            zone.Add(_ZoneImpression)
        End If

        _PageToPrint.Clear()
        'pour toutes les sections
        For Each pz As String In zone
            If pz.Contains("-") Then
                Dim debut As Integer = CInt(pz.Split("-")(0))
                Dim fin As Integer = CInt(pz.Split("-")(1))
                For i = debut To fin
                    _PageToPrint.Add(i)
                Next
            Else
                Dim page As Integer = CInt(pz)
                If _PageToPrint.Contains(page) Then
                    MessageBox.Show(String.Format("Il n'est pas autorisé d'imprimer
plusieurs fois la page n°{0}", page), "impression", MessageBoxButtons.OK,
                    MessageBoxIcon.Information)
                    Return False
                End If
            End If
        Next
    Catch
        Return False
    End Try
End Function

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>131/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------


```

        End If
        _PageToPrint.Add(page)
    End If
Next
For Each page As Integer In _PageToPrint
    If page > _pageMax Then
        MessageBox.Show(String.Format("L'impression de la page {0} n'est pas
possible car il n'y a que {1} page(s) max", page, _pageMax), "impression", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        Return False
    End If
    If page < 1 Then
        MessageBox.Show("Impossible d'imprimer une page avec un numéro négatif ou
égal à zéro", "impression", MessageBoxButtons.OK, MessageBoxIcon.Information)
        Return False
    End If
Next
Catch ex As Exception
    Return False
End Try
Return True
End Function
#End Region

#Region "Validation OK Pour Imprimer"
Private Sub PrintZone() Handles TXT_PrintZone.Leave
    _ZoneImpression = TXT_PrintZone.Text
End Sub
Private Sub CB_ListeImp_SelectedIndexChanged() Handles CB_ListeImp.SelectedIndexChanged
    _NomPrinter = Me.CB_ListeImp.SelectedItem
End Sub
Private Sub TXT_AT_TextChanged() Handles TXT_AT.TextChanged
    If TXT_AT.Text.Length >= _AT_NBCAR_MIN Then
        TXT_AT.BackColor = Drawing.Color.White
    Else
        TXT_AT.BackColor = Drawing.Color.Pink
    End If
    _AuditTrails = TXT_AT.Text
End Sub
Private Sub NumericUpDowns_ValueChanged() Handles NUD_1.ValueChanged, NUD_2.ValueChanged
    NUD_2.Minimum = NUD_1.Value
    _ZoneImpression = NUD_1.Value & " - " & NUD_2.Value
End Sub
#End Region

#Region "Evènement"
Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RB_OPT1.CheckedChanged
    'Pour les superpositions de GroupBox
    'Le num1 doit toujours etre visible sinon le GB2 ne le sera pas
    'et si GB 2 visible alors GB1 ne l'est plus
    If RB_OPT1.Checked Then
        GB_OPT2.Visible = False
        Call NumericUpDowns_ValueChanged()
    Else
        GB_OPT2.Visible = True
    End If
End Sub

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>132/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

Private Sub CB_AT_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CB_AT.SelectedIndexChanged
    If CB_AT.SelectedIndex = CB_AT.Items.Count - 1 Then
        TXT_AT.Text = Nothing
        TXT_AT.Visible = True
    Else
        TXT_AT.Visible = False
        _AuditTrails = CB_AT.SelectedItem
    End If
End Sub
''' <summary>
''' vérification du format de page en mode interval
''' </summary>
Private Sub VerifFormat() Handles TXT_PrintZone.TextChanged
    Dim newTxt As String = Nothing

    For Each c As String In TXT_PrintZone.Text.ToCharArray
        Dim Num = Asc(c)
        If Num >= Asc("0") And Num <= Asc("9") Then
            newTxt += Chr(Num)
        ElseIf Num = Asc(";") Or Num <= Asc("-") Then
            newTxt += Chr(Num)
        End If
    Next

    TXT_PrintZone.Text = newTxt

End Sub
#End Region

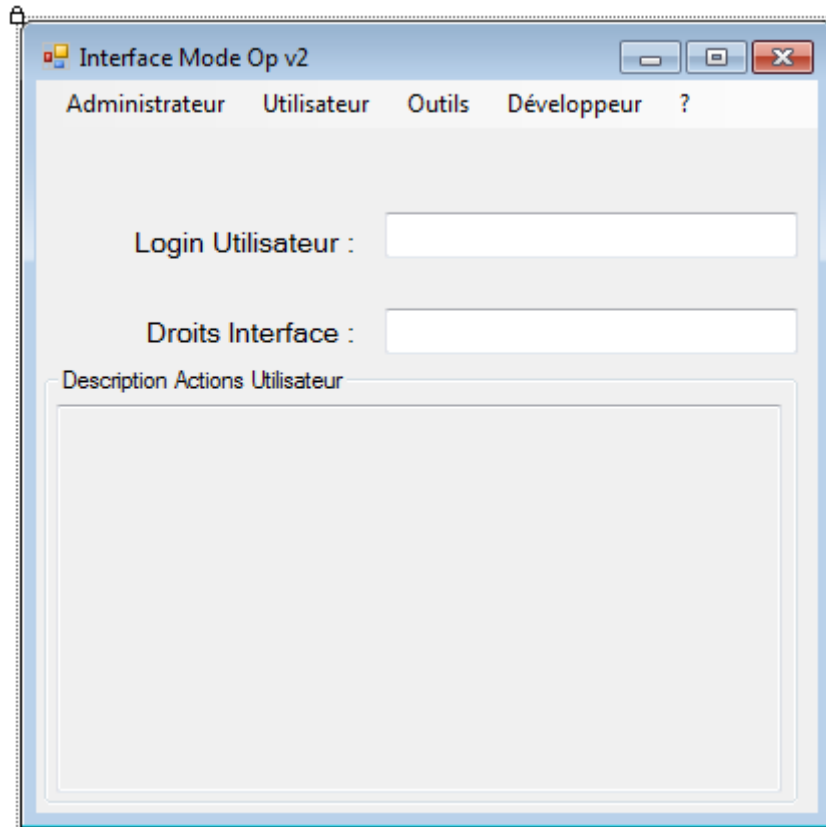
Private Sub BT_Validation_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BT_Validation.Click
    If isOKToPrint() Then
        _FormulaireValid = True
        Me.Close()
    End If
End Sub

End Class

```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>133/156</b>

### 9.37 VuePrincipale



```
Public Class vuePrincipale
```

```
    Private Shared _Instance As vuePrincipale = Nothing
    Private Shared ReadOnly mylock As New Object()
```

```
    Private Const _ESPACELIGNE As String = "_____"
```

```
    Private Const _MAX_CHAR_AFFICHAGE As Integer = 1024
```

```
#Region "Constructeur"
```

```
    Private Sub New()
```

```
        ' Cet appel est requis par le concepteur.
        InitializeComponent()
```

```
        ' Ajoutez une initialisation quelconque après l'appel InitializeComponent().
```

```
        GestionMenuDroitUser()
```

```
        Me.TXT_LoginUtilisateur.Text = Initialisation.__User.getUserName
```

```
        Me.TXT_Droits.Text = Initialisation.__User.getDroitDetermine.ToString
```

```
        Me.TXT_Action.Text = "Initialisation OK"
```

```
#If DEBUG Then
```


```
    Me.Info("--[MODE DEBUG]--", True)
```

```
#End If
```

```
    End Sub
```

```
    Private Sub GestionMenuDroitUser()
```

```
        Select Case Initialisation.__User.getDroitDetermine
```


	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>134/156</b>

```

Case Droits.Guest 'consultation
    GestionMenu(False)
    GestionMenu(True, TSMI_Utilisateur)
    GestionMenu(False, TSMI_Utilisateur_Impression)
Case Droits.User 'GUEST +impression
    GestionMenu(False)
    GestionMenu(True, TSMI_Utilisateur)
Case Droits.KeyUser 'USER +Export
    GestionMenu(False)
    GestionMenu(True, TSMI_Utilisateur, TSMI_Administrateur)
    GestionMenu(False, TSMI_Administrateur_Importation)
Case Droits.UserAQ 'USER +outils
    GestionMenu(False)
    GestionMenu(True, TSMI_Utilisateur, TSMI_Outils)
Case Droits.AdminAQ '+ gestion droit utilisateur
    GestionMenu(False, TSMI_Developpeur)
Case Droits.AdminDvlp
    GestionMenu(True)
Case Else
    GestionMenu(False)
    Info("Droits indéterminés")
End Select
GestionMenu(True, TSMI_Info)
End Sub
''' <summary>
''' Gestion des menus d'en-tete
''' </summary>
Private Overloads Sub GestionMenu(ByVal key As Boolean)
    For Each item As ToolStripMenuItem In MenuStrip1.Items
        item.Visible = key
    Next
End Sub
''' <summary>
''' Gestion des menus en fonction de la liste
''' </summary>
''' <param name="ListeTSMI">liste des menus affectés</param>
''' <remarks></remarks>
Private Overloads Sub GestionMenu(ByVal key As Boolean, ByVal ParamArray ListeTSMI() As
ToolStripMenuItem)
    For Each Menu As ToolStripMenuItem In ListeTSMI
        Menu.Visible = key
    Next
End Sub
''' <summary>
''' Gestion des menus (mère/enfants)
''' </summary>
''' <param name="MenuMere">les menus mère / enfants seront affectés</param>
''' <remarks></remarks>
Private Overloads Sub GestionMenu(ByVal key As Boolean, ByVal MenuMere As ToolStripMenuItem)
    For Each item As ToolStripMenuItem In MenuMere.DropDownItems
        item.Visible = key
    Next
    MenuMere.Visible = key
End Sub

'accesseur de la vue principale
Public Shared Function getVP() As vuePrincipale
    SyncLock (mylock)
        If IsNothing(_Instance) Then

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>135/156</b>


```

        _Instance = New vuePrincipale()
    End If
    Return _Instance
End SyncLock
End Function
#End Region

#Region "Méthodes Externes"
''' <summary>
''' Affichage des informations à destination de l'utilisateur dans le TextBox
''' </summary>
''' <param name="monTexte">Information à ajouter</param>
''' <param name="NouveauBloc">Défini un nouveau bloc d'information et permet d'effacer le
TextBox si surchargé</param>
''' <remarks></remarks>
Public Sub Info(ByRef monTexte As String, Optional ByRef NouveauBloc As Boolean = False)
    Dim Affichage As New System.Text.StringBuilder(monTexte)
#If DEBUG Then
    NouveauBloc = False
#End If
    With Affichage
        .AppendLine()
        If NouveauBloc Then
            .Append(_ESPACELIGNE).AppendLine()
            If Me.TXT_Action.Text.Length >= _MAX_CHAR_AFFICHAGE Then Me.TXT_Action.Clear()
        End If
        .Append(Me.TXT_Action.Text)
        Me.TXT_Action.Text = .ToString
    End With
End Sub
#End Region

#Region "Evenement Menu/interface"
Private Sub TSMI_Utilisateur_Consultation_Officiel_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Utilisateur_Consultation_Officiel.Click
    WAction.doAction(Outils.Action.ConsultationOfficiel)
End Sub
Private Sub TSMI_Utilisateur_Consultation_Archive_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TSMI_Utilisateur_Consultation_Archive.Click
    WAction.doAction(Outils.Action.ConsultationArchive)
End Sub
Private Sub TSMI_Utilisateur_Impression_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Utilisateur_Impression.Click
    WAction.doAction(Outils.Action.Impression)
End Sub
Private Sub TSMI_Administrateur_Importation_DepuisModif_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Administrateur_Importation_DepuisModif.Click
    WAction.doAction(Outils.Action.Importation)
End Sub
Private Sub TSMI_Administrateur_Archivage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Administrateur_Archivage.Click
    WAction.doAction(Outils.Action.Archivage)
End Sub
Private Sub TSMI_Administrateur_Exportation_Officiel_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Administrateur_Exportation_Officiel.Click
    WAction.doAction(Outils.Action.ExportationOfficiel)
End Sub
Private Sub TSMI_Administrateur_Exportation_Archive_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TSMI_Administrateur_Exportation_Archive.Click

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>136/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

        WAction.doAction(Outils.Action.ExportationArchive)
    End Sub
    Private Sub TSMI_Info_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TSMI_Info.Click
        With My.Application.Info
            Dim mesInfo As New System.Text.StringBuilder(.AssemblyName)
            mesInfo.AppendLine.AppendLine()
            mesInfo.Append(.Copyright).AppendLine()
            mesInfo.Append(String.Format("Version {0}.{1:00}", .Version.Major, .Version.Minor))
            mesInfo.Append(" ").Append(.Version).Append("]")
        #If DEBUG Then
            mesInfo.AppendLine.Append("--Mode Debug--")
        #End If
        MessageBox.Show(mesInfo.ToString, .Title, MessageBoxButtons.OK,
        MessageBoxIcon.Information)
        End With
    End Sub
#End Region


#Region "FERMETURE INTERFACE"
    Private Sub FermetureInterface() Handles Me.FormClosed
        WReader.Dispose()
        Initialisation.Close()
    End Sub
#End Region

#Region "Méthode Interne"
    Private Sub vuePrincipale_Resize() Handles MyBase.Resize
        Me.GB_Main.Size = New Size(Me.Width - 30, Me.Height - 185)
        Me.TXT_Action.Size = New Size(Me.Width - 44, Me.Height - 210)
    End Sub
#End Region

#Region "Outils et Protection BDD"
    Private Sub TSMI_Outils_AuditTrails_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Outils_AuditTrails.Click
        Const _MSG_ERR_DAO = "DAO ERROR"
        Const _MSG_ERR_GENERALE = "GENERAL ERROR"
        Try
            Dim vueAT As New VueAudittrails
            vueAT.ShowDialog()
        Catch ex As DAOException
            Info("Source : " & ex.GetErrSource, True)
            Info(ex.Message)
            Info(_MSG_ERR_DAO)
        Catch ex As Exception
            Info(ex.Message, True)
            Info(_MSG_ERR_GENERALE)
        End Try
    End Sub
    Private Sub TSMI_Developpeur_ProtectionBDD_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Developpeur_ProtectionBDD.Click
        Const NOTPROTECT = "La base de données n'est pas protégé par l'interface, voulez vous
activer la protection?"
        Const PROTECT = "La base de données est protégé par l'interface, voulez vous désactiver
la protection?"
        Const ENTETE = "SQLite protection"
        Try
            If Singleton.getModeProtection = 0 Then

```



	Libellé Système  <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système  <b>2300</b>	Type doc  <b>FS</b>	N°Ordre  <b>01</b>	Rév.  <b>F</b>	Page  <b>137/156</b>
---	---	------------------------------	---------------------------	--------------------------	----------------------	----------------------------

```

        If MessageBox.Show(NOTPROTECT, ENTETE, MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) = Windows.Forms.DialogResult.Yes Then
            Singleton.protectionBDD(True)
            MessageBox.Show("Protection Active", ENTETE, MessageBoxButtons.OK,
MessageBoxIcon.Information)
        End If
        ElseIf Singleton.getModeProtection = 1 Then
            If MessageBox.Show(PROTECT, ENTETE, MessageBoxButtons.YesNo,
MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then
                Singleton.protectionBDD(False)
                MessageBox.Show("Protection Désactivée", ENTETE, MessageBoxButtons.OK,
MessageBoxIcon.Warning)
            End If
        End If
    Catch ex As Exception
        Info(ex.Message)
    End Try

End Sub


Private Sub TSMI_Outils_AjoutUtilisateur_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TSMI_Outils_AjoutUtilisateur.Click
    Const ERRUSER = "Erreur lors de l'enregistrement de l'utilisateur {0} : {1}"
    Dim nomUser As String = InputBox("Veuillez choisir un nom de login Fareva, ce log sera
défini en gues. Pour changer les droits, seul l'adminAQ est autorisé à le faire", "Ajout login
utilisateur en GUEST")

    Try
        If Not String.IsNullOrEmpty(nomUser) Then
            If System.Text.RegularExpressions.Regex.IsMatch(Replace(nomUser, " ", ""), "[a-
zA-Z0-9]{4}") And Not nomUser.Contains(" ") Then
                Try
                    DAOFactory.getUtilisateur.dbInsert(New Utilisateur(nomUser.ToUpper))
                    MessageBox.Show("enregistrement effectué", "ajout login utilisateur",
MessageBoxButtons.OK, MessageBoxIcon.Information)
                Catch ex As Exception
                    Info(String.Format(ERRUSER, nomUser, ex.Message))
                End Try
            Else
                MessageBox.Show("Le login : " & nomUser & " n'est pas un login Fareva
valide", "login invalide", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
        End If
    Catch ex As Exception
        Info(ex.Message)
    End Try

End Sub
#End Region

End Class

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>138/156</b>

### 9.38 VBTools : DialogBox

```
Imports System
Imports System.IO
Imports System.Security
Imports System.Security.Cryptography
Imports System.Text
Imports System.Windows.Forms
Imports System.Drawing
Imports System.Drawing.Printing
```

Namespace DialogBox

```
'-----
'  Classe : CHOICEBOX
'-----

Public Class ChoiceBox
    'Liste message d'erreur
    Private Const ERR_CREATE_WINFORM = "Erreur lors de la construction de la fenêtre de
dialogue"
    Private Const ERR_DOSSIER_INCONNU = "Le chemin absolu {0} n'existe pas ou n'est pas
accessible"
    Private Const ERR_CREATIONCONTROLE = "Ereur lors de la création d'un control dans la
fenetre principale"
    Private Const ERR_DESCRIPTION = "Ereur lors du formatage de la zone Texte de
description"
    Private Const ERR_SHOWDIALOG = "Ereur lors de l'affichage de la boîte de dialogue"
    Private Const ERR_CHARGETREEVIEW = "Ereur lors de la récupération des noms de fichier
pour remplir la boîte de dialogue"


    'extention autorisé à l'affichage
    Private Const _EXT_ALL = "*.*)"

    Private _RepertoireBaseAbsolu As String
    Private _FichierSelectionne As String = Nothing
    Private _VoirFichierDansTreeView As Boolean

    Private _TextDescription As String = Nothing
    Private _TextFont As Font = Nothing
    Private _TextColor As System.Drawing.Color

    'Element du windows form
    Private WithEvents _Form_Main As New Form
    Private WithEvents _Form_Main_BT_OK As New Button
    Private WithEvents _Form_Main_BT_Annuler As New Button
    Private WithEvents _Form_Main_TreeView As New TreeView
    Private _Form_Main_TXT_Result As New TextBox
    Private _Form_Main_TXT_Description As New TextBox

#Region "Property"
    Protected WriteOnly Property setRepertoireBaseAbsolu As String
        Set(ByVal value As String)
            _RepertoireBaseAbsolu = value
        End Set
    End Property
    Protected WriteOnly Property setFichierSelectionne As String
        Set(ByVal value As String)
            _FichierSelectionne = value
        End Set
    End Property
End Class
```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>139/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```


End Set
End Property
Public ReadOnly Property getRepertoireBaseAbsolu As String
    Get
        Return _RepertoireBaseAbsolu
    End Get
End Property
Public ReadOnly Property getRepertoireBaseRelatif As String
    Get
        Return Path.GetFileName(_RepertoireBaseAbsolu)
    End Get
End Property
Public ReadOnly Property getRepertoireBeforeBase As String
    Get
        Dim frac() = _RepertoireBaseAbsolu.Split(Path.DirectorySeparatorChar)
        Dim chemin As New System.Text.StringBuilder

        For i = 1 To frac.Count - 1
            chemin.Append(frac(i - 1)).Append(Path.DirectorySeparatorChar)
        Next

        chemin.Remove(chemin.Length - 1, 1)

        Return chemin.ToString
    End Get
End Property
''' <summary>
''' Nom complet absolu du résultat sélectionné
''' </summary>
''' <returns>Le chemin absolu du dossier ou du fichier sélectionné</returns>
Public ReadOnly Property getResultatFull As String
    Get
        Return _FichierSelectionne
    End Get
End Property
''' <summary>
''' Nom complet relatif du résultat sélectionné
''' </summary>
''' <returns>Le chemin relatif du dossier ou du fichier sélectionné</returns>
Public ReadOnly Property getResultatRelatif As String
    Get
        Return Replace(_FichierSelectionne, _RepertoireBaseAbsolu, "..")
    End Get
End Property
''' <summary>
''' Nom du résultat sélectionné
''' </summary>
''' <returns>Le nom du résultat sélectionné</returns>
Public ReadOnly Property getResultatSimple As String
    Get
        Return Path.GetFileName(_FichierSelectionne)
    End Get
End Property
Public Property DialogBoxTexteDescription As String
    Set(ByVal value As String)
        Try
            If IsNothing(_TextDescription) Then
                _TextDescription = value
            End If
        End Try
    End Set
End Property

```


	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>140/156</b>

```

        _Form_Main_TXT_Description.Text = value
    Catch ex As Exception
        Throw New VBToolsException(ERR_DESCRIPTION, ex)
    End Try
End Set
Get
    Return _Form_Main_TXT_Description.Text
End Get
End Property
Public Property DialogBoxPoliceDescription As Font
    Set(ByVal value As Font)
        Try
            If IsNothing(_TextFont) Then
                _TextFont = value
            End If
            _Form_Main_TXT_Description.Font = value
        Catch ex As Exception
            Throw New VBToolsException(ERR_DESCRIPTION, ex)
        End Try
    End Set
Get
    Return _Form_Main_TXT_Description.Font
End Get
End Property
''' <summary>
''' Récupère le 1er texte inscrit dans le cadre description
''' </summary>
''' <returns>Le texte dans le cadre description d'origine</returns>
''' <remarks></remarks>
Protected ReadOnly Property getDescriptionOrigineTexte As String
    Get
        Return _TextDescription
    End Get
End Property
''' <summary>
''' Récupère la 1ere Police inscrite dans le cadre description
''' </summary>
''' <returns>La 1ere police du cadre description d'origine</returns>
''' <remarks></remarks>
Protected ReadOnly Property getDescriptionOrigineFont As Font
    Get
        Return _TextFont
    End Get
End Property
#End Region

#Region "Property Windows Forms"
Protected Property MainForm_BTOK_Enabled As Boolean
    Get
        Return _Form_Main_BT_OK.Enabled
    End Get
    Set(ByVal value As Boolean)
        _Form_Main_BT_OK.Enabled = value
    End Set
End Property
Protected Property MainForm_TXTResultat_Size As Size
    Set(ByVal value As Size)
        _Form_Main_TXT_Result.Size = value
    End Set

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>141/156</b>

```

        Get
            Return _Form_Main_TXT_Result.Size
        End Get
    End Property
    Protected Property MainForm_TXTResultat_Location As Point
        Set(ByVal value As Point)
            _Form_Main_TXT_Result.Location = value
        End Set
        Get
            Return _Form_Main_TXT_Result.Location
        End Get
    End Property
    Protected Property MainForm_TreeView_Size As Size
        Set(ByVal value As Size)
            _Form_Main_TreeView.Size = value
        End Set
        Get
            Return _Form_Main_TreeView.Size
        End Get
    End Property
    Protected Property MainForm_TreeView_Location As Point
        Set(ByVal value As Point)
            _Form_Main_TreeView.Location = value
        End Set
        Get
            Return _Form_Main_TreeView.Location
        End Get
    End Property
    Protected Property MainForm_TXTResultat_Text As String
        Get
            Return Me._Form_Main_TXT_Result.Text
        End Get
        Set(ByVal value As String)
            Me._Form_Main_TXT_Result.Text = value
        End Set
    End Property
#End Region

#Region "Constructeur"
''' <summary>
''' Ouverture d'une boîte de dialogue
''' </summary>
''' <param name="RepertoireBase">Chemin absolu du dossier ou va s'ouvrir la boîte de
dialogue</param>
''' <remarks></remarks>
Public Sub New(ByVal RepertoireBase As String, ByVal VoirFichier As Boolean)

    If Directory.Exists(RepertoireBase) Then
        _RepertoireBaseAbsolu = RepertoireBase
        _VoirFichierDansTreeView = VoirFichier
    Else
        Throw New VBToolsException(String.Format(ERR_DOSSIER_INCONNU, RepertoireBase))
    End If

    Try
        With _Form_Main
            .Size = New Size(686, 508)
            .MinimumSize = .Size
            .MaximumSize = .Size

```

VALDEPHARM	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>142/156</b>

```

'TEXTBOX
Setup(_Form_Main_TXT_Result, 496, 23, 0, 440)
_Form_Main_TXT_Description.Multiline = True
Setup(_Form_Main_TXT_Description, 670, 58, 0, 0)
_Form_Main_TXT_Description.ReadOnly = True
_Form_Main_TXT_Result.ReadOnly = True
_TextColor = _Form_Main_TXT_Description.ForeColor

'TREEVIEW
Setup(_Form_Main_TreeView, 670, 370, 0, 66)
RemplissageTreeView()


'BOUTON
Setup(_Form_Main_BT_Annuler, 75, 23, 502, 440)
_Form_Main_BT_Annuler.Text = "Annuler"
Setup(_Form_Main_BT_OK, 75, 23, 583, 440)
_Form_Main_BT_OK.Text = "OK"
_Form_Main_BT_OK.Enabled = False

'FORM PRINCIPALE
.Text = _RepertoireBaseAbsolu
End With
Catch ex As Exception
Throw New VBToolsException(ERR_CREATE_WINFORM, ex)
End Try
End Sub
#End Region

#Region "Gestion ChoixFichier en Externe"
Public Overridable Sub ShowDialog()
Try
_Form_Main.ShowDialog()
Catch ex As Exception
Throw New VBToolsException(ERR_SHOWDIALOG, ex)
End Try
End Sub
#End Region

#Region "Gestion des controles"
''' <summary>
''' Création d'un control dans la fenetre principale
''' </summary>
''' <param name="Ctrl">nom du control</param>
''' <param name="W">Size Width</param>
''' <param name="H">Size Height</param>
''' <param name="X">Location X</param>
''' <param name="Y">Location Y</param>
''' <remarks></remarks>
Protected Sub Setup(ByVal Ctrl As Control, ByVal W As Integer, ByVal H As Integer, ByVal
X As Integer, ByVal Y As Integer)
Try
_Form_Main.Controls.Add(Ctrl)
With Ctrl
.Size = New Size(W, H)
.Location = New Point(X, Y)
End With
Catch ex As Exception
Throw New VBToolsException(ERR_CREATIONCONTROLE, ex)

```


	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>143/156</b>

```

End Try
End Sub
Protected Sub RemplissageTreeView(Optional ByRef listeExtention As List(Of String) =
Nothing)
Try
    _Form_Main_TreeView.Nodes.Clear()
    With _Form_Main_TreeView
        .TopNode = .Nodes.Add(_RepertoireBaseAbsolu, getRepertoireBaseRelatif)
        'boucle sur tout les répertoires du dossier courant
        For Each Rep As String In Directory.GetDirectories(_RepertoireBaseAbsolu)
            'ajoute le dossier dans le treeview
            .TopNode.Nodes.Add(Rep, Path.GetFileName(Rep))
            'entre dans le dossier
            NextNode(Rep, _Form_Main_TreeView.TopNode, listeExtention)
        Next
        If _VoirFichierDansTreeView Then
            'boucle sur tout les fichiers du dossier courant
            For Each Fichier As String In Directory.GetFiles(_RepertoireBaseAbsolu)
                If Not Path.GetFileName(Fichier).Chars(0) = "~" Then
                    If IsNothing(listeExtention) Then
                        .TopNode.Nodes.Add(Path.GetFileName(Fichier))
                    Else
                        If
listeExtention.Contains(Path.GetExtension(Fichier).ToLower) Then
                            .TopNode.Nodes.Add(Path.GetFileName(Fichier))
                        End If
                    End If
                End If
            Next
        End If
    End With
    Catch ex As Exception
        Throw New VBToolsException(ERR_CHARGETREEVIEW, ex)
    End Try
End Sub
Private Sub NextNode(ByVal Repertoire As String, ByVal NodeActuel As TreeNode, ByRef
listeExtention As List(Of String))
Try
    Dim Node As TreeNode = NodeActuel.Nodes(Repertoire)

    For Each rep As String In Directory.GetDirectories(Repertoire)
        Node.Nodes.Add(rep, Path.GetFileName(rep))
        NextNode(rep, Node, listeExtention)
    Next
    If _VoirFichierDansTreeView Then
        For Each Fichier As String In Directory.GetFiles(Repertoire)
            If Not Path.GetFileName(Fichier).Chars(0) = "~" Then
                If IsNothing(listeExtention) Then
                    Node.Nodes.Add(Path.GetFileName(Fichier))
                Else
                    If listeExtention.Contains(Path.GetExtension(Fichier).ToLower)
Then
                        Node.Nodes.Add(Path.GetFileName(Fichier))
                    End If
                End If
            End If
        Next
    End If
End Sub

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>144/156</b>

```

        Catch ex As Exception
            Throw New VBToolsException(ERR_CHARGETREEVIEW, ex)
        End Try
    End Sub
#End Region

#Region "Evenement"
Private Sub BTCancel_click() Handles _Form_Main_BT_Annuler.Click
    _FichierSelectionne = Nothing
    _Form_Main.Close()
End Sub
Private Sub BTOK_click() Handles _Form_Main_BT_OK.Click
    _Form_Main.Close()
End Sub

Protected Overridable Sub TreeViewSelectionFichier(ByVal sender As Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles _Form_Main_TreeView.AfterSelect
    Dim selection As New System.Text.StringBuilder

    With selection
        .Append(getRepertoireBeforeBase)
        .Append(Path.DirectorySeparatorChar)
        .Append(_Form_Main_TreeView.SelectedNode.FullPath)
    End With

    If _VoirFichierDansTreeView Then
        If File.Exists(selection.ToString) Then
            _FichierSelectionne = selection.ToString
            _Form_Main_TXT_Result.Text = _Form_Main_TreeView.SelectedNode.Text
            _Form_Main_BT_OK.Enabled = True
        Else
            _FichierSelectionne = Nothing
            _Form_Main_TXT_Result.Text = String.Empty
            _Form_Main_BT_OK.Enabled = False
        End If
    Else
        _FichierSelectionne = selection.ToString
        _Form_Main_TXT_Result.Text = _Form_Main_TreeView.SelectedNode.Text
        _Form_Main_BT_OK.Enabled = True
    End If
End Sub
#End Region
End Class

'-----
' Classe : BOXSAVEFILE
'-----
Public Class BoxSaveFile
    Inherits ChoiceBox


    Private Const ERR_NOMETHODEEXT = "La méthode de gestion de l'extention n'est pas connue"
    Private Const ERR_NOTEXT = "{0} n'est pas une extention de fichier valable"
    Private Const ERR_NOMNOTGOOD = "{0} n'est pas un nom de fichier reconnu"

    'si le fichier existe déjà
    Private Const ERR_SAMEFILE = "Le fichier {0} existe déjà!!!"
    Private _TextDescFont As New Font("Arial", 12.0, FontStyle.Italic)

    Private _nomExtention As String

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>145/156</b>

```

Private _nomFichierParDefault As String
Private _selectionTreeView As String

'Element Windows Forms
Private WithEvents TXT_Fichier As New TextBox
Private TXT_Ext As New TextBox

Enum ext As Integer
    RemplaceExtention = 1
    AdditionneExtention = 2
End Enum


#Region "Constructeur"
''' <summary>
''' Sélection d'un dossier pour enregistrement d'un fichier
''' </summary>
''' <param name="RepertoireRacine">Répertoire Racine lors de l'enregistrement</param>
''' <param name="NomFichierParDefault">Fichier avec extension non modifiable</param>
''' <remarks></remarks>
Sub New(ByVal RepertoireRacine As String, ByVal NomFichierParDefault As String)
    MyClass.New(RepertoireRacine, NomFichierParDefault,
Path.GetExtension(NomFichierParDefault), ext.RemplaceExtention)
End Sub
''' <summary>
''' Sélection d'un dossier pour enregistrement d'un fichier
''' </summary>
''' <param name="RepertoireRacine">Répertoire Racine lors de l'enregistrement</param>
''' <param name="NomFichierAvecExtention">Fichier avec extension</param>
''' <param name="Extension">.<ext></param>
''' <param name="action">remplace l'extension existante (.txt->.ext) ou l'ajoute (.txt-
>.txt.<ext></param>
''' <remarks></remarks>
Sub New(ByVal RepertoireRacine As String, ByVal NomFichierAvecExtention As String, ByVal
Extension As String, ByVal action As ext)
    MyBase.New(RepertoireRacine, False)

    '(note : création classe service avec check regex?)
    Dim regex As New System.Text.RegularExpressions.Regex("^\\.[a-zA-Z0-9_]+")
    If Not regex.Match(Extension).Success Then
        Throw New VBToolsException(String.Format(ERR_NOTEXT, Extension))
    End If

    Dim regex2 As New System.Text.RegularExpressions.Regex("[a-zA-Z0-9\\s\\-_éàè\\+]+[\\.a-
zA-Z0-9]+")
    If Not regex2.Match(NomFichierAvecExtention).Success Then
        Throw New VBToolsException(String.Format(ERR_NOMNOTGOOD,
NomFichierAvecExtention))
    End If

    If action = ext.RemplaceExtention Then
        _nomExtention = Extension
        _nomFichierParDefault =
Path.GetFileNameWithoutExtension(NomFichierAvecExtention)
    ElseIf action = ext.AdditionneExtention Then
        _nomExtention = Path.GetExtension(NomFichierAvecExtention) & Extension
        _nomFichierParDefault =
Path.GetFileNameWithoutExtension(NomFichierAvecExtention)
    Else
        Throw New VBToolsException(ERR_NOMETHODEEXT)
    End If
End Sub

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>146/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

End If

CtrlConstruction()
MyBase.MainForm_BTOK_Enabled = False
End Sub
Private Sub CtrlConstruction()
'TREEVIEW
MainForm_TreeView_Size = New Size(MainForm_TreeView_Size.Width,
MainForm_TreeView_Size.Height - 29)

'TEXTBOX
'on copie les coordonnées de TXT_Result avant de la changer de place
Setup(TXT_Fichier, MainForm_TXTResultat_Size.Width - 46,
MainForm_TXTResultat_Size.Height, MainForm_TXTResultat_Location.X,
MainForm_TXTResultat_Location.Y)
MainForm_TXTResultat_Location = New Point(MainForm_TXTResultat_Location.X,
MainForm_TXTResultat_Location.Y - 29)
MainForm_TXTResultat_Size = New Size(MainForm_TreeView_Size.Width,
MainForm_TXTResultat_Size.Height)

With TXT_Fichier
Setup(TXT_Ext, 40, .Size.Height, .Location.X + .Size.Width + 6, .Location.Y)
TXT_Ext.ReadOnly = True
End With

TXT_Ext.Text = _nomExtention
TXT_Fichier.Text = _nomFichierParDefault


End Sub
#End Region

#Region "Region des controls"
Private Sub changeTextDescriptionSameFile(ByVal maDescription As String, ByVal monFont
As Font, ByVal maCouleur As Color)
MyBase.DialogBoxPoliceDescription = monFont
MyBase.DialogBoxTexteDescription = maDescription
Me.TXT_Fichier.BackColor = maCouleur
End Sub
#End Region

#Region "Evènement choix utilisateur"
Private Sub traitementChoixUtilisateur()
Dim cheminAbs As New System.Text.StringBuilder
Dim nomFichier As String = TXT_Fichier.Text.Trim
'détermination du nom absolu du fichier choisi
With cheminAbs
.Append(getRepertoireBeforeBase)
.Append(Path.DirectorySeparatorChar)
.Append(_selectionTreeView)
.Append(Path.DirectorySeparatorChar)
.Append(nomFichier)
.Append(_nomExtention)
End With

'si fichier existe
If File.Exists(cheminAbs.ToString) Then
changeTextDescriptionSameFile(String.Format(ERR_SAMEFILE, cheminAbs.ToString),
_TextDescFont, Color.Red)
MyBase.MainForm_BTOK_Enabled = False

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>147/156</b>

```

Else
    changeTextDescriptionSameFile(MyBase.getDescriptionOrigineTexte,
MyBase.getDescriptionOrigineFont, Color.White)
    MyBase.setFichierSelectionne = cheminAbs.ToString
    MyBase.MainForm_BTOK_Enabled = True
End If
End Sub
Protected Overrides Sub TreeViewSelectionFichier(ByVal sender As Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs)
    _selectionTreeView = e.Node.FullPath
    MainForm_TXTResultat_Text = _selectionTreeView
    traitementChoixUtilisateur()
End Sub
Private Sub TXTFichier_TextChanged() Handles TXT_Fichier.TextChanged
    Me.TXT_Fichier.Text = Me.TXT_Fichier.Text.Trim()
    traitementChoixUtilisateur()
End Sub
#End Region

End Class

'-----
' Classe : BOXOPENFILE
'-----
Public Class BoxOpenFile
    Inherits ChoiceBox
    Private Const _EXT_ALL_DESCRIPTION = "Tous (*.*)"
    Private Const _EXT_SPECIF_DESCRIPTION = "Type ({0})"
    Private Const _ERR_INITIALISATION = "Erreur pendant l'ouverture de l'objet BoxOpenFile"


    'élément Windows form ajoutés
    Private WithEvents CB As New ComboBox
    Private TXT_CBox As New TextBox

    Protected Friend _Close As Boolean = False
    Private _Ext As New List(Of String)
    Private _ListExtAPrendre As List(Of String) = Nothing

#Region "property"
''' <summary>
''' Liste d'extention de fichier à afficher de la forme (.ext)
''' </summary>
''' <value>doit être conforme à la regex : ^\[a-zA-Z0-9_\]+</value>
''' <returns>la liste des extentions</returns>
''' <remarks></remarks>
Public Property listeExtention As String()
    Get
        If IsNothing(_ListExtAPrendre) Then
            Return Nothing
        Else
            Return _ListExtAPrendre.ToArray
        End If
    End Get
    Set(ByVal value As String())
        If Not IsNothing(value) Then
            _ListExtAPrendre = New List(Of String)
            Dim desc As New System.Text.StringBuilder()

            For Each ext As String In value

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>148/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

9_]+")

        Dim regex As New System.Text.RegularExpressions.Regex("^\\.[a-zA-Z0-
        If regex.Match(ext).Success Then
            _ListExtAPrendre.Add(ext.ToLower)
            desc.Append(ext.ToLower).Append("|")
        End If
    Next

    'retire le dernier caractère
    desc.Remove(desc.Length - 1, 1)

    TXT_CBox.Text = String.Format(_EXT_SPECIF_DESCRIPTION, value)
Else
    _ListExtAPrendre = Nothing
    TXT_CBox.Text = _EXT_ALL_DESCRIPTION
End If

MyBase.RemplissageTreeView(_ListExtAPrendre)

End Set
End Property
#End Region

#Region "Constructeur"
Sub New(ByVal RepertoireRacine As String)
    MyBase.New(RepertoireRacine, True)
    Try
        'TEXTBOX
        MainForm.TXTResultat_Size = New Size(MainForm.TXTResultat_Size.Width - 177,
MainForm.TXTResultat_Size.Height)


        Setup(TXT_CBox, 171, 23, 325, 440)
        With TXT_CBox
            .Enabled = False
            .Text = _EXT_ALL_DESCRIPTION
            .BringToFront()
        End With

        Catch ex As Exception
            Throw New VBToolsException(_ERR_INITIALISATION, ex)
        End Try
    End Sub
#End Region

Public Overrides Sub ShowDialog()
    MyBase.RemplissageTreeView(_ListExtAPrendre)
    MyBase.ShowDialog()
End Sub

End Class
End Namespace

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>149/156</b>

### 9.39 VBTools : GestionDataGridView

```
Imports System.Drawing
Imports System.Windows.Forms
Imports System.Drawing.Printing
```

```
Namespace GestionDataGridView
```

```
Public Class PrintDataGridView
```

```
Private Const ERR_IMPRESSSION = "erreur lors du processus d'impression"
```

```
Private Const ERR_NODATA = "Le dataGridView est vides, il ne contient aucune donnée"
```

```
(pour le rendre visible vous devez l'ajouter à un control Form selon la méthode
```

```
[monForm.Controls.Add(monDGV)])"
```

```
'Donnée à imprimer
```

```
Private WithEvents _PrintDoc As New PrintDocument
```

```
Private _DataGV As DataGridView
```

```
'Paramètres d'affichages ENTETE
```

```
Private _Entete_Font As Font = New Font("Arial", 10, FontStyle.Bold)
```

```
Private _Entete_Stylo As Pen = New Pen(Color.Black)
```

```
Private _Entete_Brush As SolidBrush = New SolidBrush(Color.Black)
```

```
Private _Entete_StringFormat As StringFormat = New StringFormat With {.Alignment =
StringAlignment.Center,
                                     .LineAlignment =
StringAlignment.Center}
```

```
'Paramètres d'affichages CORPS DU DOCUMENT
```

```
Private _CorpsTexte_Font As Font = New Font("Arial", 10, FontStyle.Regular)
```

```
Private _CorpsTexte_Stylo As Pen = New Pen(Color.Black)
```

```
Private _CorpsTexte_Brush As SolidBrush = New SolidBrush(Color.Black)
```

```
Private _CorpsTexte_StringFormat As StringFormat = New StringFormat With {.Alignment =
StringAlignment.Center}
```

```
'Paramètres d'affichages Ligne individuelle
```

```
Private _LigneIndiv_Font As Font = New Font("Arial", 10, FontStyle.Italic)
```

```
Private _LigneIndiv_Stylo As Pen = New Pen(Color.White)
```

```
Private _LigneIndiv_Brush As SolidBrush = New SolidBrush(Color.Black)
```

```
Private _LigneIndiv_StringFormat As StringFormat = New StringFormat With {.Alignment =
StringAlignment.Near}
```

```
Private _LigneIndiv_Text As String = Nothing
```

```
'Coordonnées de la zone d'impression
```

```
Private _Xbegin, _Ybegin, _Xend, _Yend As Integer
```

```
Private _LigneEnCours As Integer = 1
```

```
Private _modeAffichage As Affichage
```

```
'Détermine la position d'écriture de la nouvelle ligne
```

```
Private _Ycursor As Integer
```

```
'nombre de page imprimées
```

```
Private _nbPage As Integer = 0
```

```
'Détermine si une nouvelle page est nécessaire pour finir l'impression
```


```
Private _NextPage As Boolean
```

```
#Region "Property"
```

```
Public Property textePremierePage As String
```

```
Get
```


```
Return _LigneIndiv_Text
```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>150/156</b>

```

        End Get
        Set(ByVal value As String)
            _LigneIndiv_Text = value
        End Set
    End Property
    Public Property Font_Entete As Font
        Get
            Return _Entete_Font
        End Get
        Set(ByVal value As Font)
            _Entete_Font = value
        End Set
    End Property
    Public Property Font_CorpsTexte As Font
        Get
            Return _CorpsTexte_Font
        End Get
        Set(ByVal value As Font)
            _CorpsTexte_Font = value
        End Set
    End Property
    Public Property Font_LigneIndiv As Font
        Get
            Return _LigneIndiv_Font
        End Get
        Set(ByVal value As Font)
            _LigneIndiv_Font = value
        End Set
    End Property
    Public Property CouleurEncadrement_Entete As Pen
        Get
            Return _Entete_Stylo
        End Get
        Set(ByVal value As Pen)
            _Entete_Stylo = value
        End Set
    End Property
    Public Property CouleurEncadrement_CorpsTexte As Pen
        Get
            Return _CorpsTexte_Stylo
        End Get
        Set(ByVal value As Pen)
            _CorpsTexte_Stylo = value
        End Set
    End Property
    Public Property CouleurEncadrement_LigneIndiv As Pen
        Get
            Return _LigneIndiv_Stylo
        End Get
        Set(ByVal value As Pen)
            _LigneIndiv_Stylo = value
        End Set
    End Property
    Public Property CouleurTexte_Entete As SolidBrush
        Get
            Return _Entete_Brush
        End Get
        Set(ByVal value As SolidBrush)
            _Entete_Brush = value

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>151/156</b>

```

        End Set
    End Property
    Public Property CouleurTexte_CorpsTexte As SolidBrush
        Get
            Return _CorpsTexte_Brush
        End Get
        Set(ByVal value As SolidBrush)
            _CorpsTexte_Brush = value
        End Set
    End Property
    Public Property CouleurTexte_LigneIndiv As SolidBrush
        Get
            Return _LigneIndiv_Brush
        End Get
        Set(ByVal value As SolidBrush)
            _LigneIndiv_Brush = value
        End Set
    End Property
#End Region

#Region "Enumération"
    Public Enum Affichage As Byte
        All = 1
        Selection = 2
    End Enum
#End Region


#Region "Constructeur"
    Sub New(ByRef MyDataGridView As DataGridView)
        If MyDataGridView.Rows.Count = 0 Then
            Throw New VBToolsException(ERR_NODATA)
        End If
        _DataGV = MyDataGridView
    End Sub
#End Region

#Region "Procédure externe"
    Public Overloads Sub Impression(Optional ByVal mode As Affichage = Affichage.All)
        _modeAffichage = mode

        Try
            Dim P As New PrintPreviewDialog
            _PrintDoc.DefaultPageSettings.Landscape = True
            _PrintDoc.DefaultPageSettings.Margins = New Margins(Conv(10), Conv(10),
Conv(10), Conv(10))
            P.Document = _PrintDoc
            P.ShowDialog()
        Catch ex As Exception
            Throw New VBToolsException(ERR_IMPRESSION, ex)
        End Try
    End Sub
#End Region

#Region "Processus d'impression"
    Private Sub DeterminationParametreImpression(ByRef e As
System.Drawing.Printing.PrintPageEventArgs)
        'Détermination point (0,0)
        _Xbegin = e.MarginBounds.Left
        _Ybegin = e.MarginBounds.Top

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>152/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

```

'Détermination point (Max,Max)
_Xend = e.MarginBounds.Right
_Yend = e.MarginBounds.Bottom
End Sub

''' <summary>
''' Se déclenche pour chaque page
''' </summary>
Private Sub ConstructionImpressionPages(ByVal sender As System.Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles _PrintDoc.PrintPage
'-----
' - initialisation du document -
'-----
If _nbPage = 0 Then
'Détermination de la zone imprimable
DeterminationParametreImpression(e)
End If

'-----
' - Initialisation de la page -
'-----
'on commence en haut du document
_Ycursor = _Ybegin
_NextPage = False

'-----
' - Ecriture ligne indiv -
'-----
If _nbPage = 0 Then
If Not IsNothing(_LigneIndiv_Text) Then
WriteLine(e, _LigneIndiv_Text)
End If
End If

'-----
' - Ecriture de l'entete -
'-----
WriteDataGridViewHeader(e)


'-----
' - Ecriture du Corps de page -
'-----
For i = _LigneEnCours To _DataGV.Rows.Count
'si ligne non sélectionnée en mode sélection
If _modeAffichage = Affichage.All Or (_modeAffichage = Affichage.Selection And
_DataGV.Rows(i - 1).Selected) Then
WriteDataGridViewRow(e, i - 1)
End If
If _NextPage Then
Exit For
End If
Next

'Détermination du nombre de ligne à imprimer pour cette page
e.HasMorePages = _NextPage

'fin de la page et incrémentation
_nbPage += 1

```



	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>153/156</b>

```

End Sub

''' <summary>
''' Ecriture d'une ligne
''' </summary>
Private Sub WriteLine(ByRef e As System.Drawing.Printing.PrintPageEventArgs, ByVal Texte
As String)
    Dim monRectangle As Rectangle
    Dim coeffLigne As Integer = determineLignes(Texte)

    'on prepare le format de la cellule
    monRectangle = New Rectangle(_Xbegin, _Ycursor, _Xend - _Xbegin,
_LigneIndiv_Font.GetHeight() * coeffLigne * 1.05)

    'on complète l'intérieur
    e.Graphics.DrawString(Texte, _LigneIndiv_Font, _LigneIndiv_Brush, monRectangle,
_LigneIndiv_StringFormat)
    'on dessine le contour de la cellule
    e.Graphics.DrawRectangle(_LigneIndiv_Stylo, Rectangle.Round(monRectangle))

    'on déplace le curseur
    _Ycursor += _LigneIndiv_Font.GetHeight() * coeffLigne * 1.05
End Sub

''' <summary>
''' Ecriture d'une ligne du DataGridView
''' </summary>
Private Sub WriteDataGridViewRow(ByRef e As System.Drawing.Printing.PrintPageEventArgs,
ByVal ligneDataGridView As Integer)
    Dim monRectangle As Rectangle
    Dim monTexte As String
    Dim Xcursor As Single = _Xbegin
    Dim Xcol As Single = 0
    Dim coeffLigne As Integer = determineLignes(ligneDataGridView)

    'si ligne à écrire sort de la zone d'impression
    If _Ycursor + (_CorpsTexte_Font.GetHeight() * coeffLigne) > _Yend Then
        _LigneEnCours = ligneDataGridView
        _NextPage = True
        Exit Sub
    End If


    For i = 1 To _DataGV.ColumnCount
        If _DataGV.Columns(i - 1).Visible = True Then

            'calcul de la longueur du rectangle en proportion équivalente a celle du
DataGridView
            Xcol = _DataGV.Columns(i - 1).Width * (_Xend - _Xbegin) /
XTotalDataGridView()

            'on prepare le format de la cellule
            monRectangle = New Rectangle(Xcursor, _Ycursor, Xcol,
_CorpsTexte_Font.GetHeight() * coeffLigne * 1.05)
            'le texte de la cellule
            monTexte = _DataGV.Rows(ligneDataGridView).Cells(i - 1).Value

            'on complète l'intérieur
            e.Graphics.DrawString(monTexte, _CorpsTexte_Font, _CorpsTexte_Brush,
monRectangle, _CorpsTexte_StringFormat)

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>154/156</b>

```

        'on dessine le contour de la cellule
        e.Graphics.DrawRectangle(_CorpsTexte_Stylo, Rectangle.Round(monRectangle))
        'prochaine coordonnée d'écriture
        Xcursor += Xcol

    End If
Next

    'on déplace le curseur
    _Ycursor += _CorpsTexte_Font.GetHeight() * coeffLigne * 1.05
End Sub


''' <summary>
''' Ecriture de l'entete du DataGridView
''' </summary>
Private Sub WriteDataGridViewHeader(ByRef e As
System.Drawing.Printing.PrintPageEventArgs)
    Dim monRectangle As RectangleF
    Dim monTexte As String
    Dim Xcursor As Single = _Xbegin
    Dim Xcol As Single = 0
    Dim coeffLigne As Integer = determineLignes()

    For i = 1 To _DataGV.ColumnCount
        If _DataGV.Columns(i - 1).Visible = True Then
            'calcul de la longueur du rectangle en proportion équivalente a celle du
DataGridView
            Xcol = _DataGV.Columns(i - 1).Width * (_Xend - _Xbegin) /
XTotalDataGridView()
            'on prepare le format de la cellule
            monRectangle = New RectangleF(Xcursor, _Ycursor, Xcol,
_Entete_Font.GetHeight() * coeffLigne * 1.05)
            'le texte de la cellule
            monTexte = _DataGV.Columns(i - 1).HeaderText
            'on dessine le contour de la cellule
            e.Graphics.DrawRectangle(_Entete_Stylo, Rectangle.Round(monRectangle))
            'on complète l'intérieur
            e.Graphics.DrawString(monTexte, _Entete_Font, _Entete_Brush, monRectangle,
_Entete_StringFormat)
            'prochaine coordonnée d'écriture
            Xcursor += Xcol
        End If
    Next

    'on déplace le curseur
    _Ycursor += _Entete_Font.GetHeight() * coeffLigne * 1.05
End Sub
#End Region

#Region "Fonction interne"
Private Function Conv(ByVal Millimetre As Integer) As Integer
    Return CInt(PrinterUnitConvert.Convert(Millimetre * 10.0R,
PrinterUnit.TenthsOfAMillimeter, PrinterUnit.Display))
End Function
Private Function XTotalDataGridView() As Single
    Dim L As Single
    For i = 1 To _DataGV.ColumnCount
        If _DataGV.Columns(i - 1).Visible Then
            L += _DataGV.Columns(i - 1).Width
        End If
    Next
    Return L
End Function

```

	Libellé Système	N°Système	Type doc	N°Ordre	Rév.	Page
	<b>Interface de gestion des modes opératoires de la chimie</b>	<b>2300</b>	<b>FS</b>	<b>01</b>	<b>F</b>	<b>155/156</b>

```

        End If
    Next
    Return L
End Function
''' <summary>
''' Retourne le nombre de caractère max affichable par une ligne
''' </summary>
''' <returns>le nombre de caractère max</returns>
''' <remarks>
''' formule déterminé par excel :
''' nbChar = 549.16 * police^-1.039
''' r² = 0.9996
''' pour une longueur de colonne de 364
''' police 40 --> 12 chars
''' police 20 --> 24 chars
''' police 10 --> 51 chars
''' police 8 --> 63 chars
''' </remarks>
Private Function nbCharMaxParLigne(ByVal police As Single, ByVal tailleColonne As
Single) As Integer
    Dim nbCar As Single = Math.Pow(police, -1.039) * 549.16
    Return tailleColonne * nbCar / 364
End Function
''' <summary>
''' Retourne le nombre de lignes nécessaires à l'affichage
''' </summary>
''' <param name="numeroLigneDGV">numéro de ligne du DGV (-1 pour l'entete)</param>
''' <returns>nombre de ligne affichée</returns>
''' <remarks></remarks>
Private Function determineLignes(Optional ByVal numeroLigneDGV As Integer = -1)
    Dim coeffMultiplicateur As Integer = 1
    Dim Xcol As Single = 0
    For i = 1 To _DataGV.ColumnCount
        Dim coeff As Integer
        Dim texte As String


        Xcol = _DataGV.Columns(i - 1).Width * (_Xend - _Xbegin) / XTotalDataGridView()

        If numeroLigneDGV = -1 Then
            texte = _DataGV.Columns(i - 1).HeaderText
            coeff = (texte.Length / nbCharMaxParLigne(_Entete_Font.Size, Xcol)) + 1
        Else
            texte = _DataGV.Rows(numeroLigneDGV).Cells(i - 1).Value
            coeff = (texte.Length / nbCharMaxParLigne(_CorpsTexte_Font.Size, Xcol)) + 1
        End If

        If coeff > coeffMultiplicateur Then coeffMultiplicateur = coeff
    Next

    Return coeffMultiplicateur
End Function
Private Function determineLignes(ByVal monTexte As String)
    Return (monTexte.Length / nbCharMaxParLigne(_LigneIndiv_Font.Size, _Xend - _Xbegin))
+ 1
End Function
#End Region
End Class
End Namespace

```

	Libellé Système <b>Interface de gestion des modes opératoires de la chimie</b>	N°Système <b>2300</b>	Type doc <b>FS</b>	N°Ordre <b>01</b>	Rév. <b>F</b>	Page <b>156/156</b>
---	---	--------------------------	-----------------------	----------------------	------------------	------------------------

#### 9.40 VBTools : VBToolsException

```

Public Class VBToolsException
    Inherits Exception

    Private Const _ERR_NOSOURCE = "Pas d'erreur source"
    Private _errSource As String

#Region "Property"
    Public ReadOnly Property getErrSource As String
        Get
            Return _errSource
        End Get
    End Property
#End Region

#Region "Constructeur"
    Public Sub New(ByVal errmsg As String, Optional ByVal errExcep As Exception = Nothing)
        MyBase.New(errmsg)
        _errSource = If(IsNothing(errExcep), _ERR_NOSOURCE, errExcep.Message)
    End Sub
#End Region

End Class

```