

Maxime
Blanc

RAPPORT DE STAGE

02/01/23 - 10/03/23

Sommaire

Remerciements.....	3
I - Introduction.....	4
A - La recherche.....	4
B - Présentation de l'entreprise.....	4
C - Présentation de la mission.....	5
D - Mes objectifs.....	5
II - Les missions.....	6
A - Contexte professionnel des missions.....	6
B - Problèmes / Contraintes / Limites rencontrés.....	7
III - Les démarches suivies.....	8
A - Organisation des missions.....	8
B - Processus et méthodes suivis.....	10
IV - Réalisation des missions.....	12
A - Les outils utilisés.....	12
B - Les étapes essentielles.....	13
V - Evaluation des réalisations et des compétences mobilisées.....	27
A - Adéquation du travail.....	27
B - Compétences mise en œuvre.....	27
VI - Conclusion.....	29
A - Analyse des conditions de travail.....	29
B - Apport des missions pour le stagiaire.....	29
C - Prochain stage / Alternance.....	30
VII - Bibliographie / Annexes.....	31

Remerciements

Je remercie tous ceux qui ont été impliqués de près ou de loin par mon stage.

Je remercie Franck Maisonneuve, qui à la suite de mon premier stage m'a proposé de venir travailler chez MSA et qui m'a donc permis d'entrer dans cette entreprise.

Je remercie Eric Gudin pour m'avoir guidé et accompagné durant toute la durée du stage, d'avoir pris le temps de m'enseigner de nouvelles compétences et un nouveau langage et de m'avoir fait confiance.



I - Introduction

La recherche

Lors de mon stage de première année, j'avais pu réaliser un site web pour le comité des fêtes de Châtillon sur Chalaronne. A la fin de celui-ci, mon maître de stage Franck Maisonneuve, qui travaille chez MSA, m'avait proposé de me prendre chez eux pour mon stage de deuxième année si j'en faisais la demande assez tôt.

Voulant quand même aborder plusieurs domaines, j'ai entrepris des recherches dans différents secteurs de l'informatique, dans plusieurs entreprises mais comme pour ma première année, les résultats n'ont pas été très concluants encore une fois.

Néanmoins, j'avais commencé ma recherche assez tôt et après des prises de contacts avec Franck Maisonneuve, ma demande fût acceptée et mon stage validé pour une durée de 10 semaines (02 janvier au 10 mars) dans l'entreprise MSA.

Présentation de l'entreprise

MSA (Mine Safety Appliances) est une entreprise américaine de fabrication d'équipement de protection individuelle, fondée en 1914 en Pennsylvanie. Dès lors, elle n'a cessé de s'expander et compte désormais plus de 5 200 collaborateurs, répartis dans 5 continents à travers 40 filiales.

Dans sa zone Europe, MSA compte 3 centres d'excellence dont fait partie le centre de Châtillon sur Chalaronne, spécialisé dans la protection de la tête.

En effet, en 2002, l'entreprise américaine MSA (**M**ine **S**afety **A**pliances) rachète l'usine châtillonnaise CGF Gallet, créée en 1860, alors en pleine réussite grâce à ses casques de protection. Depuis, MSA Gallet continue son expansion déjà grandissante en devenant notamment le fournisseur officiel de l'armée de l'air et de la Marine nationale ou encore en étendant sa gamme de produits aux gendarmes et policiers.

Avec un chiffre d'affaires estimé à 50 millions d'euros environ et environ 230 personnes employées dans toute la France, MSA Gallet est réputé comme le leader mondial de la protection de la tête.

La direction de l'entreprise est composée, entre autres, de :

- James Daugherty Jr, le président de MSA Gallet Holding.
- Kimberly Moore, directeur général.
- Richard Roda, directeur général.

Présentation de la mission

Mon secteur d'activité se trouvait sur la partie Injection/Composite. Cette partie de l'usine utilise 9 presses qui étaient reliées à un seul ordinateur comportant un logiciel gérant toutes les presses ensemble.

La mission principale consistait à développer un logiciel en VB .NET qui permettrait aux employés de gérer chacune des presses individuellement. Ce logiciel serait exécuté sur 9 ordinateurs, configurés au préalable, que nous installerions sur chaque presse afin de toutes les gérer séparément et ainsi, plus facilement.

Pour réaliser ce qui m'a été confié, j'ai utilisé des langages de programmation tels que :

- Visual Basic .NET
- SQL
- Visual Basic For Applications (VBA)

Nous avons dû aussi utiliser des commandes Linux lors de la configuration des PC.

Je suis intervenu en tant que développeur junior durant ce stage dans l'entreprise MSA Gallet.

Mes objectifs

Dans le cadre de notre BTS SIO, nous avons la chance de pouvoir effectuer deux stages, au rythme d'un par année scolaire. Le stage, lors de ma première année, était une découverte en tout point aussi bien d'un point de vue professionnel que d'un point de vue technique.

Néanmoins, dans la continuité de ce que j'ai pu faire l'année précédente, je me suis fixé des objectifs afin d'affirmer mon choix d'orientation et de carrière. Mes objectifs pour le stage de deuxième année sont les suivants :

- Découvrir un nouveau langage de programmation.
- Découvrir cette branche plus fonctionnelle et moins créative comme peut l'être le développement web.
- Découvrir le travail et la vie en entreprise.
- Continuer d'affirmer la voie dans laquelle je veux me lancer.

II - Les Missions

Contexte professionnel des missions

Présentation

Mon stage s'est déroulé autour d'une grande mission principale, ce qui fait que nous avons un objectif pour la fin de cette période : rendre un logiciel fonctionnel qui gèrera l'avancée de la production de la presse à laquelle il est associé. Il permettra aussi d'afficher les données concernant tous les éléments de la base de données de l'entreprise.

Ce logiciel est divisé en deux parties. La partie administration où sont listés tous les éléments de l'entreprise tels que les articles, les matières ou encore les contrôles à effectuer selon la période de production avec la possibilité d'en ajouter. La deuxième partie est la partie directement sur le poste lié à la presse. Elle permet de gérer la production et toutes les données liées à cette dernière.

En effet, cela se déroule selon le schéma suivant. Lors du début de la production, l'opérateur ajoute l'ordre de fabrication qui définit l'article en cours de production ainsi que sa quantité et sa date de livraison. Une fois l'ajout terminé, l'opérateur peut suivre en direct l'avancement de la production ainsi que les informations concernant tout ce qui tourne autour de cette dernière tels que la gamme de production ou encore le moule utilisé.

Eric Gudin s'occupe de la partie administration, qu'il avait déjà commencé à développer bien avant mon arrivée, tandis que je travaille sur la partie poste qui sera installée sur les 9 ordinateurs liés aux presses.

Durant la conception de ce logiciel, il est primordial de penser au plus logique et simple possible. En effet, l'application étant un gros morceau, il est important de pouvoir s'y retrouver lors de correctifs de bugs ou d'amélioration du code. De plus, une bonne logique de code entraîne une consommation moindre de data, ce qui est un avantage. De fait, le chargement de l'application est plus rapide et permet d'économiser parfois des petites secondes qui peuvent être précieuses.

Contexte

Aujourd'hui toutes les presses sont liées à un seul et unique ordinateur central. Cette organisation est dangereuse puisqu'en cas de souci technique, toute la chaîne de production peut être affectée et cela peut donc impacter l'entreprise. Le logiciel fonctionne toujours mais cela apporte une sécurité en plus.

En plus des soucis de panne, les paramètres de chaque presse sont donc stockés sur le même ordinateur. Cela rend plus compliqué leur consultation mais aussi, s'il y a le besoin, la modification des valeurs car nous ne sommes pas à l'abri d'une erreur.

Enjeu

L'enjeu est donc de sécuriser le fonctionnement global mais aussi de simplifier l'utilisation du logiciel pour les employés. Cette nouvelle application amènera de nouvelles fonctionnalités garantissant un meilleur contrôle de la qualité et une traçabilité plus efficace.

En effet, des contrôles réguliers seront ajoutés afin de supprimer la moindre petite gêne dès le début et ainsi éviter qu'elle se propage. La traçabilité sera aussi améliorée puisque tout sera sauvegardé et plus facile d'accès.

Problèmes / Contraintes / Limites rencontrés

Le Visual Basic étant un langage assez complet, il m'a été assez compliqué, au début, de comprendre comment s'organisait son code, l'agencement de ses fichiers, etc... Il m'a fallu aussi comprendre tout ce qui est nécessaire au fonctionnement de la solution du projet tels que les références à ajouter pour pouvoir afficher des médias non pris en charge par Visual Basic (vidéos, webcams...) ou encore ce qu'est une solution ou un fichier de configuration.

En outre, en plus de la logique de code du langage en lui-même, j'ai dû aussi m'adapter à la logique de code d'Eric Gudin. En effet, étant donné qu'après mon stage, il n'y aura plus que lui pour s'occuper de l'application. Il est nécessaire qu'il puisse comprendre le code au premier coup d'œil afin d'éviter de perdre du temps à se remémorer ce qui avait été fait jadis.

III - Les démarches suivies

Organisation des missions

Organisation :

Durant mes 10 semaines de stage, je commençais ma journée à 9 heures et je la finissais à 17 heures en comptant une heure pour manger et ce, durant les 5 jours de la semaine. Etant donné que la mission du logiciel de presses était l'objectif principal de mon stage, je travaillais continuellement dessus.

Eric Gudin étant un prestataire externe de l'entreprise, il n'était pas sur site tous les jours de la semaine. Ainsi, il me guidait lorsqu'il était là et je travaillais en autonomie lorsqu'il n'était pas dans l'entreprise.

Lorsqu'il venait voir mon code, il corrigeait mes erreurs mais il en profitait ensuite pour me lancer sur ce qu'il y avait à faire.

Comme nous travaillions sur le secteur Injection/Composite, nous étions souvent en discussion avec Franck Maisonneuve qui gère ce secteur afin qu'il puisse nous faire remonter ce qu'il voyait à l'intérieur de l'usine et ses observations concernant notre logiciel.

Tous les lundis matin à 10h, nous avons une réunion avec Franck Maisonneuve, Eric Gudin et moi-même pour faire un point sur l'avancée de notre travail, faire en sorte que nous soyons tous les trois sur la même longueur d'onde. Eric en profitait aussi pour nous montrer ce que devrait apporter le logiciel une fois mis en place à l'aide d'un projet similaire déjà implémenté dans une autre entreprise.

Formation :

Le VB .NET étant un nouveau langage pour moi, je suis arrivé sans aucune connaissance concernant cette technologie. Afin d'acquérir mes premières bases et de comprendre la logique, j'ai fait des exercices durant les deux premières semaines.

Ainsi, Eric Gudin m'a d'abord fait travailler sur la partie visuelle en me demandant de recréer le logiciel en place dans l'usine par exemple, puis il m'a initié aux fonctions et aux méthodes de code pur. Pour maximiser ma vitesse de

compréhension, il m'a aussi partagé la partie administration du futur logiciel qu'il avait déjà fait pour que je puisse m'inspirer et mieux comprendre.

De plus, dès que je bloquais sur un problème ou que j'avais des doutes sur ce que j'étais en train de réaliser, Eric Gudin prenait le temps de corriger mes erreurs et de me montrer la voie à emprunter afin de résoudre le problème qui m'avait été confié.

Afin de compléter ma formation, je me suis aussi renseigné avec de la documentation sur internet avec des sites tels que Stack Overflow ou Microsoft Learn. Cela m'a permis de mieux comprendre le fonctionnement de certains services mais aussi de compléter efficacement mon code (Microsoft Sql Server, Visual Basic, VBA...).

Processus et méthodes suivis

Méthode

Afin de comprendre comment s'organisait notre méthode, il faut tout d'abord comprendre comment s'établit un projet sur Visual Studio.

Le projet global génère une solution. Notre code se répartissait dans 3 fichiers distincts. Le premier est le formulaire designer. C'est ce qui nous permet de définir ce à quoi va ressembler l'application.

The screenshot shows a Windows application window titled 'Form1'. The form has a header with the MSA logo and 'The Safety Company'. The main title is 'Nom Presse'. Below this, there are several sections: 'Programme' with a dropdown menu, 'transfert vers CleF usb', 'transfert vers Serveur', 'Blivage', and 'Associer Blive'. The 'N° ordre de fabrication' field contains '0'. The 'Référence / Désignation' field is empty. The 'Fin de production le' field shows '15/02/2023 16:00:00'. There are sections for 'Fiche d'instruction' with 'Info. Presse', 'Info. Moule', and 'Gamme démarrage'. A 'Défaillance' section is also present. The 'Cotons' section has a 'SCAN CARTON' button and an 'IMPRESSION ETIQUETTE' button. The status bar at the bottom displays 'lundi 27 février 2023 16:03:22'.

Afin de remplir ces formulaires, nous pouvons utiliser des composants tels que des boutons, des zones de textes ou encore des tableaux mais nous pouvons aussi ajouter des contrôles utilisateurs. Ceux-ci permettent de créer des parties de formulaires afin de les réutiliser plus facilement. Par exemple, la partie basse de notre formulaire d'accueil avec les six boutons est un contrôle utilisateur.

Le deuxième type de fichier est le fichier qui permet de gérer le code derrière le formulaire selon les événements tels que les clics ou le chargement de la page par exemple.

Ces deux types de fichiers portent le même nom. Par exemple, pour la page Accueil, il y a aura un fichier qui s'appellera Accueil.Designer.vb et qui contient le code pour l'aspect visuel de la page et un autre fichier qui s'appellera Accueil.vb qui regroupe donc tout ce qui permet de gérer les événements associés à cette page.

```

Public Class FrmAccueil
    Private Sub FrmAccueil_Load(sender As Object, e As EventArgs) Handles Me.Load
        If LectureConfiguration() = False Then
            End
        End If

        mesParams = New acParametres
        'Connection lecteur code barre
        monlecteurCodeBarre.Connection()

        'Affichage données
        MonEntete.Affichage()

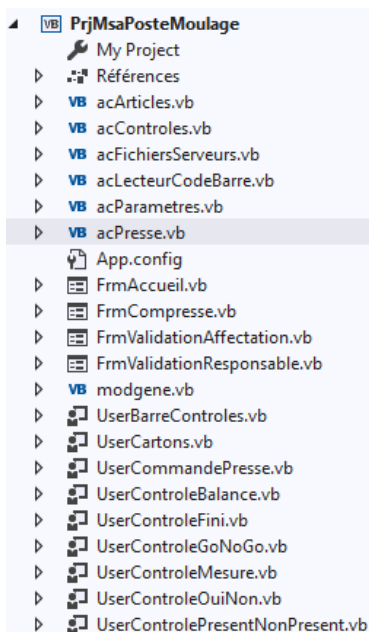
        PilotagePresse.Affichage()

        'If maPresse.EtatPresse = Presse.acEtatPresse.Init Then
        '    CommndePresse.Enabled = False
        'End If
    End Sub
End Class

```

Le troisième et dernier type de fichier est le fichier classe. C'est dans ceux-ci que nous mettons toutes les fonctions et toutes les interactions avec la base de données.

Fort de ses nombreuses années d'expérience, Eric Gudin avait une logique de code bien précise et j'ai dû m'adapter afin qu'il puisse s'y retrouver et corriger plus facilement à l'avenir. Ainsi, afin que tous les types de fichiers soient rangés ensemble, nous inscrivions un sigle devant chaque nom de fichier.



Les formulaires commençaient tous par "Frm", les fichiers classes étaient précédés de l'inscription "ac" suivis de ce qu'ils s'occupaient comme "acPresse" par exemple, les contrôles utilisateurs, quant à eux, étaient nommés "User" suivi de ce qu'ils présentaient. Le projet entier, quant à lui, était intitulé "Prj" puis le nom du projet.

Cette logique s'appliquait aussi aux composants des formulaires, les boutons et les tableaux étaient précédés respectivement de "Cmd" et de "Tab". Les TextBox, qui permettaient à l'utilisateur du logiciel de renseigner du texte, commençaient tous avec la lettre "L".

IV - Réalisation des missions

Les outils utilisés

Dans cette partie, nous allons faire une revue d'effectif des logiciels que nous avons utilisés durant le stage.

Microsoft Visual Studio :

Microsoft Visual Studio est une suite de logiciels de développement. Il permet de générer des applications web ASP.NET , des services web XML, mais aussi des applications bureautiques et applications mobiles.

Ce logiciel est utilisé comme environnement de développement intégré par Visual Basic entre autres car il permet de partager des outils et facilite la création de solution. Il permet aussi de tirer au mieux parti du framework .NET.

Nous avons utilisé ce logiciel comme environnement de développement intégré (IDE) principal durant le stage car il permet d'importer l'ensemble d'un projet et de ses librairies afin de lancer le mode debug et de tester le programme par le biais de points d'arrêts.



Microsoft SQL Server :



Microsoft SQL Server est un système de gestion de base de données en langage SQL. Ce logiciel permet de gérer plusieurs base de données en même temps et de faire des opérations inter-bases. De plus, il permet la connexion à des bases de données se trouvant sur un autre réseau.

Nous avons principalement utilisé ce logiciel pour la gestion de nos bases de données. Lors de la première partie du stage, je gérais une base de données locale, qui se trouvait sur ma machine. Tandis que durant la deuxième partie, les actions que nous faisons se déroulaient sur la base de données du serveur.

Les étapes essentielles

Le logiciel de collecte de données de traçabilité sur des presses

Le logiciel était composé de deux parties. La partie administration et la partie poste. Eric s'occupait de la première partie tandis que je m'occupais de la deuxième.

Le développement s'est déroulé en plusieurs parties :

- L'ajout et vérification de l'ordre de fabrication
- Les contrôles à effectuer
- La récupération des données du serveur
- L'acquisition de données

Chaque partie aborde des méthodes et des domaines différents qui m'ont permis de diversifier mes activités durant le stage et de constamment acquérir de nouvelles connaissances.

Ajout et vérification de l'ordre de fabrication

La première partie que j'ai eu à effectuer a été de gérer l'ajout d'ordre de fabrication afin de gérer la production. Avant tout code, il fallait établir l'apparence de la page qui devait permettre à l'opérateur d'effectuer plusieurs actions : établir les valeurs de la production (quantité, délai), gérer la gamme de production et les moules de pièce, effectuer les contrôles nécessaires à la bonne qualité des pièces produites.

The screenshot shows a software window titled "FmValidationAffectation". The main heading is "Veuillez scanner votre ordre de fabrication". The interface is divided into several sections:

- Validation Ordre de fabrication:** Contains fields for "N° ordre de", "Délai prévu" (mardi 28), "Qty prévue" (0), "Article", "Délai réel" (mardi 28), and "Qty validée" (0). There is a "Validation" button next to the "Qty validée" field.
- Validation Gamme de production:** Contains fields for "Nombre de", "N° de gamme", "N° de Moule", "Cycle Standard", and "Nb Empreintes". There are "Reche Haut" and "Reche Bas" buttons.
- Contrôles:** A large empty area for controls.

At the bottom of the window are buttons for "Enregistrer" and "Annuler". A numeric keypad is visible at the bottom left of the main content area.

Afin de procéder à ces formalités, nous les séparons donc en trois blocs, le bloc ordre de fabrication, le bloc gamme de production et enfin, le bloc contrôles. Ces derniers doivent se faire l'un après l'autre et ne peuvent pas être modifiés en même temps. Pour ce faire, nous avons utilisé des GroupBox afin de les regrouper et de les différencier plus facilement.

Un ordre de fabrication, que nous appellerons désormais OF, est ce qui permet d'organiser les productions. Il contient un article, une quantité et un délai. Ces deux derniers sont les objectifs de la production en cours. Lorsque nous voulons ajouter un OF à la presse afin de lancer la production, il faut d'abord aller le chercher dans la table des ordres de fabrication SAP. Cette table regroupe tous les ordres de fabrications créés.

```
Public Class OrdreFabricationCreation
    Dim mOrdreFabricationMSA As Integer
    Dim mIdArticle As Integer
    Dim mIdLigne As Integer
    2 références
    Public Property Reference As String
    2 références
    Public Property Designation As String
    3 références
    Public Property QtePrevue As Integer
    4 références
    Public Property Delai As Date
    4 références
    Public Property IdGamme As Integer
    2 références
    Public Property QteReelle As Integer
    2 références
    Public Property DelaiReel As Date
    0 références
    Public Property TpsCycle As Integer
    0 références
    Public Property NbEmpreintes As Integer
    2 références
    Public Property TpsCycleReel As Integer
    2 références
    Public Property NbEmpreintesReelles As Integer
    Public msgammes As List(Of Gamme)

    0 références
    Public Property IdValideur As Integer
    0 références
    Public Property ObservationsValidation As String
```

Afin de gérer les données renseignées et de pouvoir les réutiliser, nous créons une classe "OrdreFabricationCreation" qui sert uniquement lors de l'ajout d'un OF à la presse. Elle contient toutes les variables que nous allons utiliser lors de la procédure et va nous permettre d'afficher les données récupérées dans la table SAP mais aussi d'enregistrer celles modifiées par l'opérateur. Mais avant d'afficher les données grâce à cette classe, il nous faut d'abord assigner à chaque variable sa valeur correspondante.

Ainsi, lorsque nous scannons une fiche OF avec un lecteur code barre, nous devons récupérer les informations liées à ce paramètre. Toutefois, si lors du scan, l'identifiant n'existe pas. Alors, nous avons la possibilité de le créer manuellement grâce aux deux boutons que nous voyons en haut de la page. L'ordre de fabrication sera inséré dans la table SAP et nous pourrons le retrouver en scannant sa fiche.

Afin de récupérer les données sur la fiche associée, nous devons utiliser un lecteur code-barre et pour ce faire, nous devons le connecter et le paramétrer. Nous créons une classe "LecteurCodeBarre" qui contient tous les paramètres nécessaires à son bon fonctionnement.

```
Public Class LecteurCodeBarre
```

```
Dim WithEvents monport As New SerialPort
```

```
Public Event DonneesArrivees()
```

Il faut d'abord initialiser plusieurs choses. Un événement qui lorsque les données arriveront déclenche une fonction, une variable qui permet de gérer les paramètres du port série et enfin des propriétés "ReadOnly" qui permettent, comme son nom l'indique, d'utiliser ces propriétés en dehors de cette classe seulement pour lire les variables de cette dernière et non pour les modifier.

```
Public ReadOnly Property Lecture As String
```

```
Get
```

```
Return mDerniereValeur
```

```
End Get
```

```
End Property
```

```
Dim mPort As String
```

```
1 référence
```

```
Public ReadOnly Property EstConnecte As Boolean
```

```
Get
```

```
Try
```

```
Return monport.IsOpen
```

```
Catch ex As Exception
```

```
Return False
```

```
End Try
```

```
End Get
```

```
End Property
```

```
Public Function Connection() As Boolean
```

```
Connection = False
```

```
Try
```

```
If monport.IsOpen Then
```

```
monport.Close()
```

```
End If
```

```
monport.BaudRate = 9600
```

```
monport.StopBits = StopBits.One
```

```
monport.Parity = Parity.None
```

```
monport.PortName = mPort
```

```
monport.DataBits = 8
```

```
monport.Open()
```

```
Connection = True
```

```
Catch ex As Exception
```

```
MsgBox(ex.Message, vbOKOnly, vbCritical)
```

```
End Try
```

```
End Function
```

Une fois les initialisations effectuées, il faut connecter ce lecteur à notre application. Cette fonction "Connection" va nous permettre de vérifier l'état du scan à chaque fois que nous en faisons l'usage.

A l'intérieur, nous paramétrons tout ce qui est nécessaire au bon fonctionnement de l'appareil, la vitesse en bauds série, le nombre de bits d'arrêt ainsi que la longueur des bits de données présents dans un octet. Une fois que tous ces paramètres sont affectés, la fonction demandée prend la valeur "True" et notre code peut continuer de s'exécuter. L'intérêt de cette fonction est d'éviter que de nombreuses erreurs se suivent à cause du simple échec de connexion.

Lorsque les données arrivent, on déclenche l'événement "Données arrivées" et on effectue la fonction associée selon le moment et le besoin grâce à la fonctionnalité "AddHandler" qui permet d'ajouter une détection d'événement et "AddressOf" qui, lui, associe une fonction à ce déclenchement.

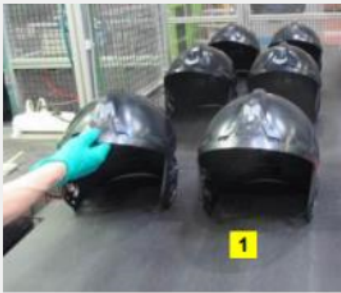

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles TimerDemarrage.Tick
    TimerDemarrage.Enabled = False
    AddHandler monlecteurCodeBarre.DonneesArrivees, AddressOf Traitement
End Sub
```

```
Private Sub Traitement()
    Try
        'RAZ de l'affichage
        Me.Invoke(Sub() RazAffichage())

        If IsNumeric(monlecteurCodeBarre.Lecture) Then
            If OrdreFabricationEncours.Charger(monlecteurCodeBarre.Lecture) Then
                Invoke(Sub() AffichageOF())
                mesFichiers.SynchronisationLotUn(OrdreFabricationEncours.monArticle)
                monImage.Image = Image.FromFile(OrdreFabricationEncours.monArticle.GetCheminFichierArticleLocal())
            Else
                MsgBox("Ordre de fabrication non trouvé dans SAP, veuillez le créer manuellement", vbOKOnly + vbCritical)
            End If
        End If
    Catch ex As Exception
        MsgBox(ex.Message, vbOKOnly + vbCritical)
    End Try
End Sub
```

Dans cette fonction et dans pratiquement toutes les fonctions que nous verrons dans ce rapport, nous avons ajouté une sécurité avec Try Catch. En effet, cette fonctionnalité permet, lorsqu'il y a une erreur, de gérer cette dernière sans que tout le programme ne casse et ainsi de pouvoir continuer à utiliser les parties fonctionnelles du logiciel malgré les erreurs.

Ici, nous procédons tout d'abord à des vérifications, la méthode "RazAffichage" permet de réinitialiser tous les affichages afin qu'il n'y ait pas d'incohérences tandis que la condition if permet de vérifier que la valeur scannée soit bien un OF.



N° ordre de fabrication	Delai prévu	Qte prévue
105687668	vendredi 20 janvier	400

Article	Delai réel	Qte validée
CH504304	vendredi 20 janvier	400

Calotte brute F1 XF pour métallisation, M

Validation

Une fois ces contrôles effectués, nous vérifions que les données sont bien chargées depuis la table SAP puis nous affichons le nom de l'article, la quantité prévue, etc... avec la méthode nommée "AffichageOf" et enfin, l'image que nous récupérons et affichons avec les deux lignes en dessous avec la méthode "SynchronisationLotUn" et "GetCheminFichierLocal" que nous détaillerons prochainement.


```
Private Sub AffichageOF()  
    LOrdreFabrication.Text = OrdreFabricationEncours.OrdreFabricationMSA  
    LNumeroOrdreFabrication.Text = OrdreFabricationEncours.OrdreFabricationMSA  
    LReference.Text = OrdreFabricationEncours.Reference  
    LDesignation.Text = OrdreFabricationEncours.Designation  
    LQtePrevue.Text = OrdreFabricationEncours.QtePrevue  
    QteValidee.Value = LQtePrevue.Text  
    DelaiPrevu.Value = OrdreFabricationEncours.Delai  
    DelaiValide.Value = OrdreFabricationEncours.Delai  
End Sub
```

Lors de l'exécution de la méthode "AffichageOf" nous allons chercher les valeurs correspondantes stockées dans notre variable "OrdreFabricationEncours" qui correspond à la classe "OrdreFabricationCreation" puis nous les assignons à chaque zone de texte.

Lorsque l'opérateur a fini de définir le délai et la quantité, il appuie sur le bouton de validation et les valeurs sont enregistrées. Les gammes associées à l'article sont alors chargées de la même manière et l'opérateur définit alors les données du moule associé à la gamme choisie tels que le cycle standard et le nombre d'empreintes que contient le moule.

Validation Gamme de production

Nombre de gamme :

N° de gamme	N° de Moule	Cycle Standard	Nb Empreintes	
1	270	<input type="text" value="55"/>	<input type="text" value="1"/>	<input type="button" value="Flèche Haut"/> <input type="button" value="Flèche Bas"/>

La troisième et dernière partie à remplir lors de l'ajout d'un OF est la partie des contrôles à effectuer. Ces derniers assurent le bon fonctionnement de la presse mais aussi témoignent de la qualité de la production.

Les contrôles à effectuer

Lors de l'ajout d'un OF, les contrôles peuvent être liés à trois catégories différentes. Ils peuvent être liés au démarrage de la presse, à la gamme de contrôles de démarrage standard ainsi qu'à l'article lui-même.

A la suite de la vérification de l'OF ainsi que celle de la gamme, nous chargeons les contrôles nécessaires. Nous créons une classe nommée "Controle" qui en contient un seul. Celle-ci comprend les variables qui recueillent les valeurs du contrôle mais aussi les résultats et les observations renseignés par l'opérateur.

'Valeurs du controle

Dim mIdCtrl As Integer

15 références

Public Property Libelle As String

15 références

Public Property Sanction As String

4 références

Public Property Bloquant As Boolean

7 références

Public Property ValeurMini As Integer

7 références

Public Property ValeurMaxi As Integer

3 références

Public Property Rang As Integer

Dim mIdParent As Integer

Dim mIdTypeObjet As acTypeObjet

Dim mIdTypeControle As acTypeControl

Dim mIdTypeMedia As acTypeMedia

'Resultat du controle

22 références

Public Property Observations As String

20 références

Public Property Passer As Boolean

16 références

Property Resultat As Decimal

14 références

Property Texte As String

2 références

Property ImageCtrl As Image

3 références

Public Enum acTypeControl As Byte

PresseDemarrage = 9

PresseFin = 10

GammeDemarrage = 11

GammeArret = 12

End Enum

13 références

Public Enum acTypeObjet As Byte

PresentNonPresent = 1

Balance = 2

Scan = 3

Mesure = 4

SaisieTexte = 5

Temperature = 6

GoNoGo = 7

OuiNon = 8

WebCam = 9

Temps = 10

End Enum

6 références

Public Enum acTypeMedia As Byte

Image = 1

Video = 2

Pdf = 3

End Enum

Nous utilisons aussi les énumérations afin de définir des valeurs définitives qui sont liées entre elles et de les utiliser plus facilement. Nous avons dix types de contrôles, par exemple, parmi ces derniers, "OuiNon" pose une question et l'opérateur doit dire oui ou non, même principe pour "GoNoGo" tandis que le contrôle "Mesure" consiste à renseigner une mesure en centimètres.

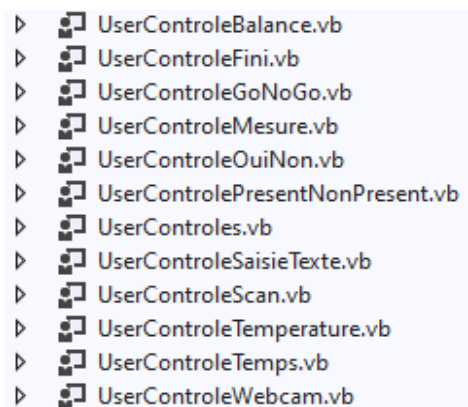
La fonction ci-dessous permet de récupérer les contrôles dans chaque catégorie. Après avoir sélectionné les valeurs dont nous avons besoin, nous initialisons un nouveau contrôle pour chaque ligne récupérée. A l'intérieur de ce dernier, nous assignons les valeurs aux variables correspondantes puis nous ajoutons ce contrôle à notre fonction qui est une liste de contrôles.

```
Public Function GetContrôleDemarrage() As List(Of Controle)  
    Dim cn As New SqlClient.SqlConnection(ChaineConnection)  
    Dim cmd As New SqlClient.SqlCommand  
    Dim rs As SqlDataReader  
    GetContrôleDemarrage = New List(Of Controle)  
    Try  
        cn.Open()  
        cmd.Connection = cn  
        'Contrôles démarrage presse  
        cmd.Parameters.Clear()  
        cmd.CommandText = "select * from [dbo].[TaContrôles] where IdParent=@IdArticle and idTypeControl=@IdTypeControl"  
        cmd.Parameters.AddWithValue("@IdArticle", IdPresse)  
        cmd.Parameters.AddWithValue("@IdTypeControl", modgene.amTypeControl.PresseDemarrage)  
        rs = cmd.ExecuteReader  
        Do While rs.Read  
            Dim moc As New Controle(rs!IdCtrl, rs!IdTypeControl, rs!IdTypeObject, rs!IdParent, rs!IdTypeMedia)  
            moc.Bloquant = rs!Bloquant  
            moc.Libelle = rs!Libelle  
            moc.Sanction = rs!Sanction  
            moc.ValeurMini = rs!ValeurMini  
            moc.ValeurMaxi = rs!ValeurMaxi  
            moc.Rang = rs!Rang  
            GetContrôleDemarrage.Add(moc)  
            moc = Nothing  
        Loop  
        rs.Close()  
        'Contrôles gamme de demarrage  
        cmd.Parameters.Clear()  
    End Try  
End Function
```

A la suite de l'enregistrement de la gamme choisie, nous stockons ensuite cette liste de contrôles dans une variable puis nous chargeons chaque contrôle un par un grâce à un contrôle utilisateur.

Afin d'afficher chaque type de contrôle plus facilement, nous avons créé un contrôle utilisateur pour chacun ainsi qu'un autre dans le but d'afficher tour à tour chaque vérification.

En plus de son visuel, chacun contient le code qui permet l'affichage du contrôle, les sécurités pour vérifier que ce dernier a été rempli ainsi que la gestion des résultats.



Le contrôle utilisateur, qui se nomme "UserControles" et qui regroupe tous les autres "UserControle", contient uniquement la barre de contrôles permettant de naviguer entre chaque vérification.

Afin d'ajouter le contrôle correspondant, nous avons une méthode nommée "Affichage". A l'intérieur de celle-ci, nous procédons d'abord à la réinitialisation en supprimant le contrôle précédent puis c'est ici que l'énumération entre en jeu. Nous utilisons "Select Case" afin de gérer toutes les éventualités du type d'objet. Lors de l'initialisation du contrôle utilisateur, nous avons créé une variable nommée "moncontroleEncours" de type "UserControl". Ce type fournit un contrôle vide qui peut servir d'autres contrôles.


```
Public Sub Affichage()
    'on ferme
    If IsNothing(moncontroleEncours) = False Then
        Me.TableLayoutPanel1.Controls.Remove(moncontroleEncours)
        moncontroleEncours = Nothing
    End If

    'on monte le controle
    Select Case mesControles(mPosition).IdTypeObjet
        Case Controle.acTypeObjet.OuiNon
            moncontroleEncours = New UserControlOuiNon
            Dim gugu As UserControlOuiNon = moncontroleEncours
            gugu.Visible = True
            gugu.Configuration(mesControles(mPosition))
            gugu.Anchor = AnchorStyles.Bottom And AnchorStyles.Left And AnchorStyles.Right And AnchorStyles.Top
            gugu.UserMedia.Charger(mesControles(mPosition))
            UserBarreControles.Rafraichir(mPosition)
            Me.TableLayoutPanel1.Controls.Add(moncontroleEncours)
            AddHandler gugu.Poursuivre, AddressOf Poursuivre
            AddHandler UserBarreControles.Changer, AddressOf ChangerControle
            gugu = Nothing
        Case Controle.acTypeObjet.Scan
            moncontroleEncours = New UserControlScan
            Dim gugu As UserControlScan = moncontroleEncours
            gugu.Visible = True
            gugu.Configuration(mesControles(mPosition))
            gugu.Anchor = AnchorStyles.Bottom And AnchorStyles.Left And AnchorStyles.Right And AnchorStyles.Top
            gugu.UserMedia.Charger(mesControles(mPosition))
            UserBarreControles.Rafraichir(mPosition)
            Me.TableLayoutPanel1.Controls.Add(moncontroleEncours)
            AddHandler gugu.Poursuivre, AddressOf Poursuivre
            AddHandler UserBarreControles.Changer, AddressOf ChangerControle
            gugu = Nothing
        Case Controle.acTypeObjet.Temperature
            moncontroleEncours = New UserControlTemperature
            Dim gugu As UserControlTemperature = moncontroleEncours
            gugu.Visible = True
            gugu.Configuration(mesControles(mPosition))
            gugu.Anchor = AnchorStyles.Bottom And AnchorStyles.Left And AnchorStyles.Right And AnchorStyles.Top
            gugu.UserMedia.Charger(mesControles(mPosition))
            UserBarreControles.Rafraichir(mPosition)
            Me.TableLayoutPanel1.Controls.Add(moncontroleEncours)
```

Ainsi, prenons l'exemple du contrôle "OuiNon". Nous assignons le type de contrôle de ce dernier à notre contrôle vide. La fonction "Configurer" nous permet d'afficher le libellé de la procédure, sa sanction mais aussi d'afficher ce qui a été renseigné par l'opérateur si le contrôle a déjà été effectué mais pas validé. Cette situation est possible grâce à la barre de contrôles qui nous permet de naviguer entre toutes les vérifications présentes. Une fois ceci configuré, nous chargeons

l'image associée au contrôle et pour finir, on l'ajoute au tableau de notre contrôle utilisateur général. Ainsi, nous reprenons la même méthode pour chaque type et les événements "Poursuivre" et "Changer" permettent de se déplacer entre les procédures.

Contrôles

		Pavé dateur												CONTROLE DIMENSIONNEL selon I110 0297											
														<input type="text"/> <input type="text"/> OUI <input type="text"/> NON <input type="text"/> <input type="text"/> POURSUIVRE											

La récupération des données du serveur

Nous avons aussi à récupérer les données nécessaires au bon fonctionnement du logiciel, aussi bien celles se trouvant dans la base de données que les images stockées sur le serveur.

LES ARTICLES

	Id	Reference	Désignation	Ref Fournisseur	Ref finie	Date Creation	Numero Boite	Compétence	Visu
▶	1	CH500700	Calotte brute F1 E12			20/02/2023 11:58:52			
	4	CH500701	Calotte brute F1 E14 BAYBLEND			20/02/2023 12:18:26			
	5	CH502122	Calot.brute ABS/PC +vent. ss trou F2X-TR			20/02/2023 12:21:18			
	6	CH502409	Sous coque F1E avec inserts			20/02/2023 12:26:50			
▶	7	CH502471	Calot.brute ABS/PC+ vent. F2 X-TR			20/02/2023 12:32:45			
	24	CH502499	Calot.brute ABS/PC non ventilé F2 X-TR			20/02/2023 12:38:28			
	25	CH502625	Calotte brute F1 SF			20/02/2023 12:38:28			
	26	CH502877	Calotte brute FUEGO			20/02/2023 12:38:29			
▶	27	CH503220	Calotte brute F1SF			20/02/2023 12:38:29			
	28	CH503577-RD	Calotte brute rouge ventilée F2 X-TREM			20/02/2023 12:38:29			
	29	CH503675	Calotte F1SF transparente			20/02/2023 12:38:29			
	30	CH503686	Calotte brute Noryl F1SF seau champagne			20/02/2023 12:38:29			
▶	31	CH504303	Calotte brute F1 XF pour métallisation, L			20/02/2023 12:38:29			
	32	CH504304	Calotte brute F1 XF pour métallisation, M			20/02/2023 12:38:29			
	33	CH504357	Calotte brute F1 XF L			20/02/2023 12:38:30			
	34	CH504358	Calotte brute F1 XF M			20/02/2023 12:38:30			
▶	35	CH504554	Calotte brute transparente F1 XF L			20/02/2023 12:38:30			
	36	CH504555	Calotte brute transparente F1 XF M			20/02/2023 12:38:30			
	37	CH510050	Calotte brute blanche MO PM			20/02/2023 12:38:30			
	38	CH510051	Calotte brute blanche MO PM			20/02/2023 12:38:30			

Durant tout le stage, je me suis principalement occupé de la partie poste du logiciel. Néanmoins, afin de prendre mes marques et d'approfondir mes connaissances sur la récupération des données de la base, Eric Gudin m'a fait travailler sur la partie administration. J'avais pour tâche de créer des tableaux et d'afficher par type tout ce qui était répertorié dans la base tels que les presses, les articles ou encore les destinations par exemple.

Nous avons utilisé la même méthode pour cette classification que pour les contrôles. Prenons l'exemple des articles, nous avons créé une classe "Article" qui définit donc un seul article. Celui-ci contient toutes les informations liées à lui. Nous avons ensuite créé une classe "Articles" qui regroupe tous les articles sélectionnés. Ainsi, lorsqu'on initialise la classe grâce à la méthode "New", on récupère tous les articles dans la base et nous les ajoutons à la classe "Articles" avec la ligne "Me.Add".

```
Public Sub New()
    Dim cn As New SqlConnection(ModGene.CHaineConnection)
    Dim cmd As New SqlCommand()
    Try
        cn.Open()
        cmd.Connection = cn
        cmd.CommandText = "select idarticle,Reference, Designation, DateCreation,visible,ReferenceFournisseur,"
        cmd.CommandType = CommandType.Text
        Dim rs As SqlDataReader = cmd.ExecuteReader
        Do While rs.Read
            Dim ma As New ArticleVisu
            ma.IdArticle = rs!idarticle
            ma.Reference = rs!Reference.ToString
            ma.Designation = rs!Designation.ToString
            ma.DateCreation = rs!DateCreation
            ma.ReferenceFournisseur = rs!ReferenceFournisseur.ToString
            ma.ReferencePiecefinie = rs!ReferencePiecefinie.ToString
            ma.Visible = rs!visible
            ma.EstSoumisCompetence = rs!EstSoumisCompetence
            ma.NumeroBoite = rs!NumeroBoite.ToString
            Me.Add(ma)
            ma = Nothing
        Loop
        rs.Close()
    Catch ex As Exception
        MsgBox(ex.Message, vbOKOnly + vbCritical)
    End Try
    cmd.Dispose()
    cn.Close()
End Sub
```

Dès lors que les articles étaient tous récupérés, il nous fallait les afficher dans des tableaux. Pour ce faire nous avons utilisé un composant nommé "DataGridView" où nous ajoutons manuellement les colonnes que nous voulons afficher telles que la désignation, l'identifiant, etc... Ce composant retourne une grille qui permet d'afficher des données très facilement. Il ne nous reste plus qu'à afficher ces dernières.

```
Private Sub Affichage()
    Dim mesarticles As New Articles
    Datas.Rows.Clear()
    Dim mindex As Byte
    Dim mindexCompetence As Byte
    For Each mc In mesarticles
        If mc.Visible Then mindex = 0 Else mindex = 1
        If mc.EstSoumisCompetence Then mindexCompetence = 0 Else mindexCompetence = 1
        Datas.Rows.Add(mc.IdArticle, mc.Reference, mc.Designation, mc.ReferencePiecefinie, mc.ReferenceFournisseur,
    Next

    Try
        Datas.FirstDisplayedScrollingRowIndex = mMemoireLigne
    Catch ex As Exception
    End Try
End Sub
```


Pour cela, nous utilisons la fonction “Affichage”. A l’intérieur de celle-ci, nous initialisons nos articles avec la méthode “New” puis nous utilisons un “For Each”. Ainsi, pour chaque article dans notre liste d’articles, on ajoute une ligne dans la grille. Par exemple, dans le code ci-dessous, nous affichons en première colonne, l’identifiant de l’article puis en deuxième, sa référence et ainsi de suite... Nous reprenons cette méthode de récupération puis d’affichage pour tous les éléments répertoriés.

Concernant les images associées, nous avons utilisé une autre méthode puisqu’elles ne sont pas stockées dans une base de données. En effet, celles-ci sont rangées dans un dossier sur le serveur et réparties dans des répertoires différents selon ce qu’elles représentent. De plus, dans le but de retrouver les images plus facilement lorsque nous les aurons importées, elles possèdent un libellé différent pour chaque élément. Les images représentant un article seront nommées avec la préposition “Art_” suivi de son identifiant, un contrôle aura “Chk_” et une presse aura “Pre_” par exemple.

ActionsKaizen	08/08/2022 11:57	Dossier de fichiers
Articles	03/03/2023 09:42	Dossier de fichiers
Conditionnements	03/03/2023 09:42	Dossier de fichiers
Controles	06/03/2023 14:01	Dossier de fichiers
Couts	14/09/2022 12:02	Dossier de fichiers
DefaultTech	14/02/2023 16:40	Dossier de fichiers
Delais	06/03/2023 13:51	Dossier de fichiers
FilEau	17/03/2022 23:25	Dossier de fichiers
Gammes	06/03/2023 14:36	Dossier de fichiers
Integrations	13/09/2022 16:42	Dossier de fichiers
Kaizens	12/09/2022 17:17	Dossier de fichiers
Moules	06/03/2023 12:34	Dossier de fichiers
Personnels	08/08/2022 14:32	Dossier de fichiers
Presentations	12/01/2023 11:59	Dossier de fichiers
Presses	06/03/2023 14:38	Dossier de fichiers
ProgMoulage	01/03/2023 14:22	Dossier de fichiers
Qualites	14/09/2022 12:48	Dossier de fichiers
Securites	27/09/2022 16:26	Dossier de fichiers

Pour les récupérer, nous avons créé une nouvelle classe nommée “acFichiersServeurs”. A l’intérieur de cette dernière, nous avons les fonctions de connexion et déconnexion au serveur mais aussi les fonctions permettant de définir le répertoire et le type d’extension (jpg, pdf...) grâce aux énumérations que nous avons vu plus haut puis le “Select Case”. Nous utilisons ensuite une fonction “CopieFichier”. Celle-ci nous permet, grâce à la méthode “File.Copy” de copier une image d’un point de départ à un point de destination. Ces derniers sont dans les paramètres de la fonction.

```

16 références
Public Function CopieFichier(FichierOrigine As String, FichierDestination As String, AfficherErreur As Boolean) As Boolean
    CopieFichier = False
    If Connection() Then
        Try
            System.IO.File.Copy(FichierOrigine, FichierDestination, True)
            CopieFichier = True
        Catch se As System.Exception
            Dim ret As Integer = Marshal.GetLastWin32Error()
            If AfficherErreur Then
                MsgBox("code erreur: " + ret.ToString & vbCrLf & se.Message, vbOKOnly + vbCritical + vbSystemModal)
            End If
        End Try
    End If
    Deconnection()
End Function

```

Nous avons ensuite créé des fonctions qui synchronisent élément par élément. Par exemple, une fonction qui copie toutes les images liées aux articles, une autre récupérant toutes celles liées aux contrôles, etc... Enfin, nous classons ces fonctions par lots. Chaque lot représente une étape de la production. Le lot 0 va s'exécuter dès le chargement du logiciel, le lot 1 lors de l'affectation de l'OF ou encore le lot 4 lors de l'arrêt de la production.

Toutefois, il nous manque toujours une partie pour que les copies s'effectuent correctement : le point d'origine et le point de destination. Il nous fallait définir le nom complet du fichier de départ mais aussi celui d'arrivée, dans le bon dossier. Pour ce faire, nous avons préalablement créé, dans le dossier source du projet, un dossier nommé "Temporaire" qui accueille tous les fichiers importés depuis le serveur.

10 références

```
Public Function GetCheminFichierControleDemarrage() As String
    Dim mRepertoire As String = ""
    Dim mf As New acFichiersServeurs
    mRepertoire = mf.CheminServeur(acFichiersServeurs.RepertoireMedia.RepertoireControleDemarrage)
    GetCheminFichierControleDemarrage = mRepertoire & GetNomFichierControle() & ".jpg"
End Function
```

11 références

```
Public Function GetCheminFichierControleDemarrageLocal() As String
    Dim mRepertoire As String = Application.StartupPath & "\Temporaire\"
    GetCheminFichierControleDemarrageLocal = mRepertoire & GetNomFichierControle() & ".jpg"
End Function
```

3 références

```
Public Function GetNomFichierControle() As String
    GetNomFichierControle = "Chk_" & (mIdCtrl).ToString
End Function
```

Prenons l'exemple des contrôles, nous avons tout d'abord créé une fonction qui définit uniquement le nom de l'image avec son extension, ici "GetNomFichierControle". Puis, celle-ci est utilisée dans deux autres fonctions. Une qui va permettre de définir le chemin du fichier du serveur, une autre qui retournera le chemin local du fichier. Nous utilisons ensuite ces deux fonctions dans les paramètres de notre "CopieFichier" et les fichiers arrivent bien dans notre dossier "Temporaire".

Acquisition de données

En plus des données de la base et du serveur, nous devons être capable de récupérer aussi des données physiques qui peuvent être sur une feuille par exemple. Nous avons déjà vu précédemment comment nous avons utilisé le lecteur code barre. En plus de ce dernier, nous nous sommes servis d'un scanner ainsi que d'une webcam.


```

Private Sub CmdScan_Click(sender As Object, e As EventArgs) Handles CmdScan.Click
    If TxtNomFiche.Text = "" Then
        MsgBox("Veuillez entrer le nom de la fiche d'instruction", vbOKOnly + vbCritical)
        Exit Sub
    End If
    Dim dlg As Object = New WIA.CommonDialog()
    Dim monScan As ImageFile = dlg.ShowAcquireImage(WiaDeviceType.ScannerDeviceType, WiaImageIntent.ColorIntent, W
    'ScannerDeviceType: Type de device souhaité (scanner, appareil photo, webcam)
    'Intent: Profile du scanner (couleur, noir et blanc ou texte)
    'Bias: Qualité
    'Format: Format de l'image souhaité
    'AlwaysSelectDevice: Affiche ou non la fenêtre de sélection du périphérique
    'UseCommonUI: affiche ou non l'UI
    'CancelError: lève ou non une exception lorsque l'utilisateur annule le scan

```

```

Imports System.Drawing.Imaging
Imports System.IO
Imports WIA

```

Afin de récupérer le scan d'un scanner, nous devons en premier lieu importer l'API Windows Image Acquisition. Cette dernière permet aux applications de numériser des images provenant

de périphériques d'acquisition tels que des scanners, des caméras, etc... Une fois cette API importée, nous pouvons commencer à nous connecter à notre scanner. Pour cela, nous ouvrons la boîte de dialogue "Wia.CommonDialog". Elle permet de sélectionner le périphérique que l'on souhaite utiliser ainsi que ses paramètres. Dès que ce dernier est sélectionné et stocké dans une variable objet, nous pouvons procéder à l'acquisition de l'image. Nous nous servons de la fonction intégrée "ShowAcquireImage" où nous mettons en paramètres le type de périphérique utilisé, son profil (couleur, noir et blanc, etc...), sa qualité et son format.

```

'on sauvegarde en local
mNomFichierComprime = Application.StartupPath & "\FichesInstruction\" & TxtNomFiche.Text & ".jpg"
Dim mNomFichierTemporaire As String = Application.StartupPath & "\Temp.jpg"
If File.Exists(mNomFichierTemporaire) Then
    File.Delete(mNomFichierTemporaire)
End If
monScan.SaveFile(mNomFichierTemporaire)
Application.DoEvents()
Scan1.Load(mNomFichierTemporaire)
Scan2.Image = Image.FromStream(Comprime(Scan1.Image))
Scan2.Image.Save(mNomFichierComprime)

```

Une fois que le scan a fini son travail, il ne nous reste plus qu'à enregistrer l'image. Nous affichons d'abord l'image dans une PictureBox afin que l'opérateur puisse évaluer si le scan est de bonne qualité. Ici, nous compressons l'image avant de l'enregistrer, c'est pourquoi nous passons par un fichier extérieur. En effet, il fallait d'abord afficher l'image non compressée avant de pouvoir la compresser. Nous enregistrons donc en mémoire l'image non compressée sous le nom de "Temp.jpg" puis nous la compressons.

```

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Try
        webcams = New FilterInfoCollection(FilterCategory.VideoInputDevice)
        flux = New VideoCaptureDevice(webcams(0).MonikerString)
        TailleWebCam = VideoSourcePlayer1.Size
        Start()
    Catch ex As Exception
        MsgBox(ex.Message, vbOKOnly + vbCritical)
    End Try
End Sub

```

Concernant la webcam, nous avons dû importer un module qui se nomme AForge.Video.DirectShow provenant de la bibliothèque AForge.NET et qui est développée spécialement pour le Framework .NET. Grâce à cette bibliothèque, nous pouvons ajouter sur notre designer un composant VideoSourcePlayer qui, comme son nom l'indique, permet d'afficher des vidéos. Lors du chargement de la page, nous récupérons le type de périphérique vidéo connecté ainsi que le flux de ce dernier.

Une fois ces deux données collectées, nous avons deux fonctions permettant de gérer l'état du périphérique. De plus, nous avons une fonction "SavePicture" qui permet de capturer l'image à l'instant T et de l'afficher dans une PictureBox. Cette fonction permet de prendre en photo des nouveaux articles par exemple.

```

Public Sub Start()
    VideoSourcePlayer1.VideoSource = flux
    VideoSourcePlayer1.Start()
    Threading.Thread.Sleep(100)
End Sub
1 référence
Public Sub Arret()
    Try
        VideoSourcePlayer1.Stop()
    Catch ex As Exception
    End Try
End Sub
1 référence
Public Sub SavePicture()
    Dim macapture As Bitmap = VideoSourcePlayer1.GetCurrentVideoFrame()
    maphoto.Image = macapture
End Sub

```

Enfin, pour ces deux types d'acquisition, l'opérateur rentre le nom du fichier qu'il vient de prendre en photo/scanner et lorsqu'il valide, le fichier s'enregistre dans le dossier correspondant selon le type de document avec le nom donné par l'opérateur.

V - Évaluation des réalisations et des compétences mobilisées

Adéquation du travail

Durant ce stage, j'ai pu me pencher sur toutes les facettes auxquelles un développeur peut être amené à travailler durant la conception d'un logiciel tel que celui-ci. Cela m'a permis de découvrir ce que sont les missions d'un développeur de logiciel en VB.NET.

Néanmoins, ma mission étant de pouvoir donner vie à la future application et mon rôle étant seulement développeur. Je travaillais donc avec Eric Gudin la plupart du temps mais je ne participais pas à toutes les tâches dont il s'occupait puisque celles-ci nécessitaient des connaissances en réseau. Ainsi, je ne n'étais pas concerné par les opérations de maintenance que pouvait effectuer Eric Gudin concernant les appareils et machines de l'entreprise.

Étant donné les travaux que j'ai pu réaliser durant ce stage, je pense avoir eu une vision d'ensemble et de base de ce à quoi consiste le rôle qui m'a été confié. Ces missions m'ont aussi permis d'acquérir des compétences qui me permettent désormais d'assumer en partie le développement d'un logiciel en VB.NET.

Compétences mises en oeuvre

Afin d'accomplir les objectifs qui m'ont été donné, j'ai pu mobiliser plusieurs compétences :

- Création de projet VB.NET
- Modélisation de page aussi bien avec des composants visuels qu'avec du code
- Elaboration de solution algorithmique par rapport à un problème donné
- Gestion et modification d'une base de données avec SQL Server

Lors de ce stage, j'ai pu apprendre énormément de nouvelles choses. Plus le temps passait, plus les connaissances affluaient et plus je prenais du plaisir à faire ce qui m'était demandé.

Ce stage m'a permis aussi d'améliorer mes compétences dans des domaines que j'avais déjà vu à l'école. Par exemple, j'ai pu gérer une base de données de grande envergure en SQL, là où je ne m'étais entraîné que sur des petites bases.

Cet apprentissage est précieux puisque bien souvent, la base de données est le socle d'un site, d'une application puisque celle-ci regroupe souvent le contenu du projet.

Afin de travailler en tant que Concepteur-Intégrateur DevOps, il me reste plusieurs axes d'améliorations :

- La partie réseau. Durant le stage, nous avons parfois eu besoin de nous servir du linux pour initialiser et configurer les neufs ordinateurs par exemple. Même si je ne souhaite pas faire ma carrière dans ce domaine, j'ai pu voir qu'il est tout de même important d'avoir quelques bases solides afin d'être capable de résoudre les problèmes présents.
- Je trouve qu'il est aussi important de me perfectionner dans les technologies que je connais déjà mais aussi d'en découvrir de nouvelles grâce aux cours prodigués à l'école mais aussi en réalisant des petits projets lors de mon temps libre. Étant donné que je souhaite me spécialiser dans l'intelligence artificielle, je compte passer cette année 2023 à m'initier à cette dernière technologie mais aussi d'approfondir mes compétences en développement mobile grâce à un projet en collaboration avec des camarades de classe.
- La relation client. Lors de ce stage, ce que nous avons à faire ne nous amenait pas à voir des clients néanmoins il était quand même possible d'échanger avec les employés travaillant directement sur les presses afin de recueillir leurs avis et leurs expériences sur le logiciel en cours. Malheureusement, je n'ai pas fait ceci et c'est pourquoi ce point est l'un des plus gros axes d'amélioration.

VI - Conclusion

Analyse des conditions de travail

Ce stage s'est déroulé dans les meilleures conditions possibles. En effet, celui-ci m'a permis d'acquérir ma première expérience dans une entreprise ainsi que de prendre part à un projet en situation réelle avec des vraies attentes à la fin de cette mission.

Concernant les conditions de travail, je ne pouvais pas demander mieux. Lorsque Eric Gudin était présent sur site, nous étions dans le même bureau ce qui me permettait de le solliciter dès que je bloquais quelque part ou que j'avais terminé le bout de code qui m'avait été confié.

Les jours où il n'était pas là m'ont permis de travailler de façon autonome, de réfléchir par moi-même aux problèmes proposés et solutions qui pouvaient être avancées.

Les horaires, quant à elles, étaient optimales. Habitant dans la même ville que l'entreprise, cette situation me permettait de gérer mes déplacements convenablement.

La confiance qui m'a été accordée est un point que j'ai apprécié durant ce stage. En effet, j'ai pu réaliser cette mission avec un objectif précis, des conditions à respecter comme si j'étais un réel employé de l'entreprise et pas comme un simple assistant.

De plus, dès le début de mon stage, j'ai pu constater la convivialité de tous les membres de l'entreprise qui échangeaient avec moi, qui m'intégraient lors des pauses par exemple. Je trouve cela toujours important afin que chacun se sente concerné et puisse donner le meilleur de soi-même pour l'entreprise.

Apport des missions pour le stagiaire

Ce stage m'a énormément apporté. En effet, selon les objectifs que je m'étais fixé en début de stage et que j'ai cité plus haut, ces missions m'ont permis de confirmer mes premières impressions et me lancer définitivement dans la voie du développement plutôt que dans celle du réseau.

J'ai pu aussi découvrir ce nouveau langage, plus fonctionnel et moins créatif visuellement. Néanmoins malgré la découverte de cette nouvelle technologie, cette

branche du développement m'intéresse moins que le développement web et mobile pour l'instant.

De plus, le stage m'a apporté des compétences bien précises telles que mentionnées plus haut dans la partie "Compétences mises en œuvre" ce qui confirme la réussite de mes objectifs initiaux.

Parmi ces derniers se trouvait l'objectif "Découvrir le travail et la vie en entreprise". J'ai pu découvrir que la vie en entreprise était bien différente de l'image qu'on pouvait se donner. En effet, lors d'un projet personnel par exemple, le plus important est que le code fonctionne correctement. Dans un cadre d'entreprise, tout est beaucoup plus réfléchi afin que le rendement soit optimal.

A titre personnel, j'ai appris à rendre des parties de code fonctionnel dans un délai défini afin d'être dans les temps concernant le développement global du logiciel. J'ai aussi pu apprendre à avoir des responsabilités dans un cadre d'entreprise lors des essais du logiciel effectués lors de la fin de mon stage.

Prochain stage / Alternance

Etant donné que mon choix est fait et que je souhaite me confirmer dans le développement, j'aimerais grandement trouver une entreprise spécialisée dans le développement informatique. De plus, ce stage dans une entreprise de l'envergure de MSA a été une réelle opportunité et je trouve que, pour ma future alternance, il est bien plus intéressant de travailler dans une grande entreprise plutôt que dans une entreprise de type TPE par exemple.

Néanmoins, je ne souhaite pas effectuer mon alternance chez MSA puisque le développement en VB.NET ne m'intéresse pas pour ma carrière future. Dans la mesure où je souhaite me spécialiser dans l'intelligence artificielle, il me serait donc préférable de trouver une entreprise avec des missions en lien avec cette technologie.

Concernant cette dernière, un élève en 5ème année à l'EPSI va être embauché dans son entreprise d'alternance et va diriger un projet basé sur l'intelligence artificielle. Sachant mon ambition de spécialisation dans cette technologie, il m'a proposé de candidater afin de prendre sa place en tant qu'alternant. Ceci est donc ma piste la plus importante pour le moment.

VII - Bibliographie / Annexes

Site de MSA :

<https://fr.msasafety.com/?locale=fr>

Site de Eric Gudin :

[E2G Informatique \(e2g-info.fr\)](http://E2G Informatique (e2g-info.fr))

Evaluation tuteur de stage