HELMo Gramme

Bachelier sciences de l'ingénieur

Programmation Orientée Objet : Moniteur d'activité

Projet de type 3

Etudiants

Deterville-François Guillaume

Teixeira Lucas

Professeur :

Bourmanne Patricia

BAC3

29 mai 2022

Table des matières

Chapitre 1

Prélude

1.1 Introduction

Notre programme, **Monitor2PDF** , est une application graphique qui consiste à, dans un premier temps, afficher des informations statiques et dynamiques de notre ordinateur. Par exemple, la carte graphique, le système d'exploitation, mais aussi les données internet ou la température de notre ordinateur. Notre ordinateur est déjà équipé d'un gestionnaire d'activité. Mais notre application va plus loin !

Premièrement, elle est plus intuitive et agréable à visualiser. Dans un second temps, et c'est son gros point fort, elle permet à l'utilisateur de générer un fichier au format PDF reprenant certaines informations directement depuis son ordinateur. Les **frontières** de l'application sont assez directes. En effet on ne peut pas afficher toutes les informations contenues dans un ordinateur. Il a donc fallu les trier et les sélectionner. C'est une application utile, qu'on peut garder ouverte en fond de tâche afin de toujours avoir un oeil sur l'activité de notre ordinateur. Et si l'on souhaite obtenir une info, il nous suffit de générer le **fichier PDF**. Il permet de centraliser toutes les informations en une seule page !

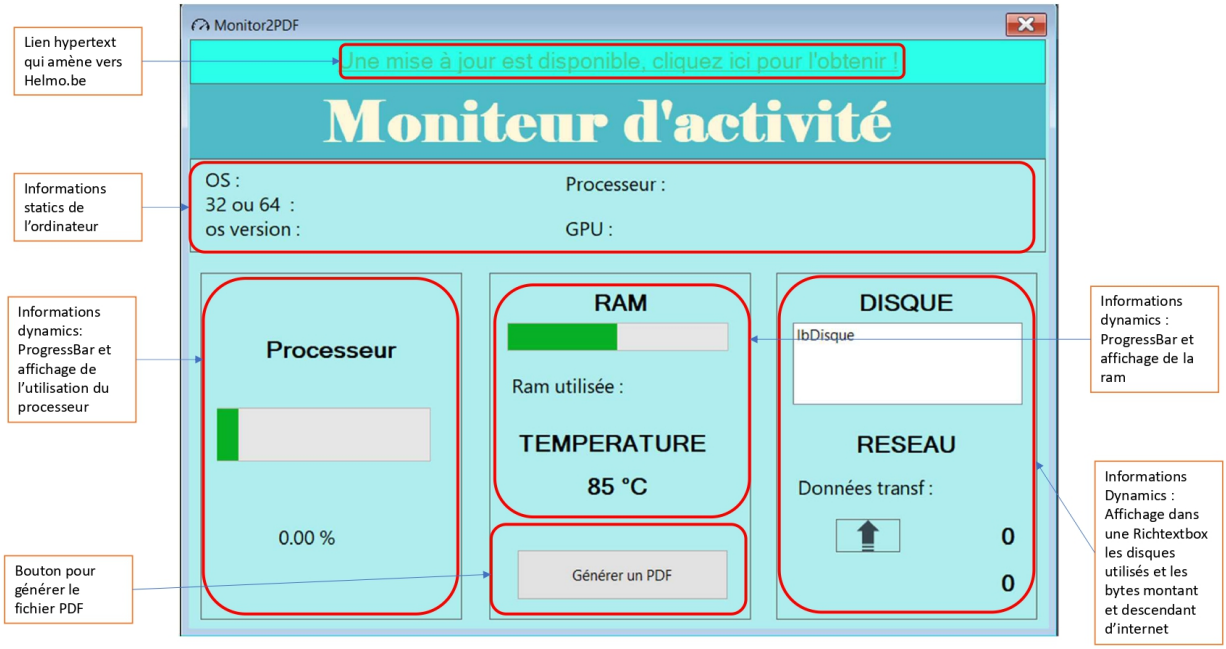
C'est une application que tout le monde peut comprendre et utiliser. Sa force est qu'elle s'adapte à toutes les machines [1] et qu'elle est facile d'utilisation. [2]

Vidéo Youtube réalisée par nos soins pour présenter le logiciel Monitor2PDF : Notre vidéo Youtube

1. Nous n'avons pas testé sur les MAC OS mais cela ne fonctionnerait pas, simplement parce que nous faisonss des requêtes de type WIndows.
2. Il est important que vous lanciez notre projet en mode administrateur. N'ayez crainte pour votre ordinateur, nous récupérons juste des informations qui nécessite parfois le droit de l'administrateur.

1.2 StoryBoard

FIGURE 1.1 – Application Monitor2PDF



Nous nous sommes basé sur plusieurs architectures, design, de moniteur d'activité pour créer le notre. Voici par exemple différentes application que nous avons trouvées et desquelles nous nous sommes inspirées Exemple 1 et Exemple 2.

L'application est scindée en deux. En haut, les informations dites statiques. Ce sont les plus faciles à afficher et récupérer. En bas, il y'a trois blocs pour les informations dites dynamiques. L'affiche de l'utilisation du processeur devait se faire par un compteur type compte-tour. [3]

Le bouton générer un PDF nous renvoie un un nouveau Form afin de donner un titre, un auteur et un sujet au PDF. L'utilisateur pourra ensuite choisir de protéger ou non son document.

3. Voir section : Comparaison avec le XAML

Chapitre 2

Investigation

2.1 L'étendue du C

Etant donné que ce projet est de type 3, il est normal que nous avons du beaucoup investiguer de nouvelles matières. Et en POO, ce n'est pas ce qu'il manque ! Internet a été notre seul moyen pour récupérer de l'informations. Nous nous sommes aidé de Youtube, de blogs, de forums (Stackoverflow est devenu notre meilleur ami). Mais aussi de cours en ligne.

Voici la matière investiguée pour ce projet :

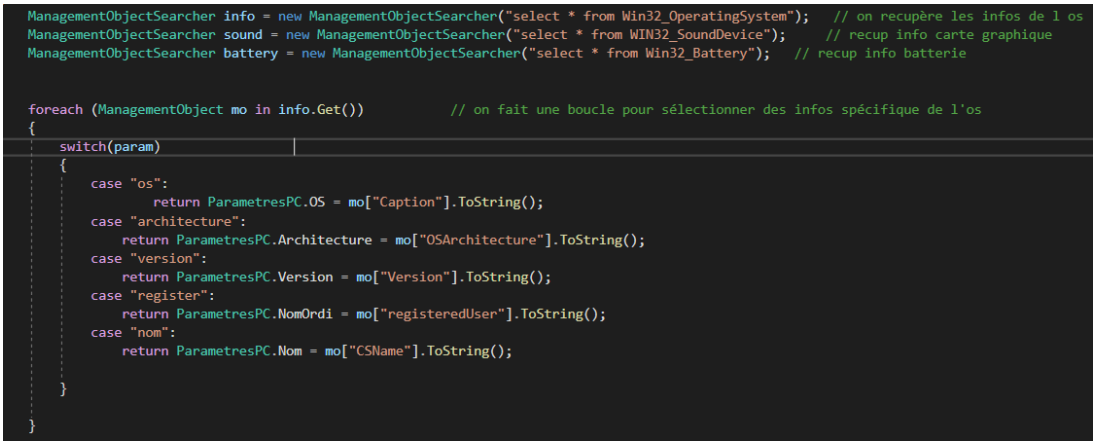
- Class ManagementObjectSearcher avec une requête type SQL
- Compréhension du binôme *foreach* & *switch*
- Class PerformanceCounter pour les valeurs dynamiques
- Class RegistryKey pour accèder à un registre dans l'ordinateur
- Package Nuget iTextSharp pour la création de document depuis c
- Mise en page du fichier PDF sur c grâce à ce site iTextSharp Guide
- Protection du document PDF avec PdfEncryptor Blog
- ...

Chapitre 3

Les points importants

Le principe pour récupérer une information est de manière générale le même, peu importe sa nature.

FIGURE 3.1 – Code de récupération des informations via $WIN32_{O}peratingSystem$



```
ManagementObjectSearcher info = new ManagementObjectSearcher("select * from Win32_OperatingSystem");   // on recupère les infos de l os
ManagementObjectSearcher sound = new ManagementObjectSearcher("select * from WIN32_SoundDevice");     // recup info carte graphique
ManagementObjectSearcher battery = new ManagementObjectSearcher("select * from Win32_Battery");   // recup info batterie


foreach (ManagementObject mo in info.Get())          // on fait une boucle pour sélectionner des infos spécifique de l'os
{
    switch(param)
    {
        case "os":
                return ParametresPC.OS = mo["Caption"].ToString();
        case "architecture":
            return ParametresPC.Architecture = mo["OSArchitecture"].ToString();
        case "version":
            return ParametresPC.Version = mo["Version"].ToString();
        case "register":
            return ParametresPC.NomOrdi = mo["registeredUser"].ToString();
        case "nom":
            return ParametresPC.Nom = mo["CSName"].ToString();

    }
}
```

— On déclare un nouvel Objet d'une classe qui nous permet d'aller récupérer une information.
— On l'instancie avec parfois des paramètres pour le constructeur.
— Ensuite grâce à des méthodes ou des propriétés on récupère les informations que l'on souhaite.
— On renvoie enfin ses informations vers le Forms et vers le fichier PDF.

3.1 Programmation d'un Timer pour les données dynamiques

La Programmation d'un Timer nous servira va de base pour toutes les informations dynamiques sur le Forms. Grâce donc à l'implémentation du Timer on

va pouvoir créer la fonction *GetRAM(),RefreshCPU()* et *GetTemp()*. Elles vont récupérer la valeur de la RAM,du CPU et de la température globale de l'ordinateur. Nous affichons ensuite ça dynamiquement dans le Forms grâce à une progressbar pour la RAM et pour le CPU. La température elle est juste un label car on ne connait pas réellement les limites des processeurs en terme de température.

L'avant dernière information qu'on récupère sur le l'ordinateur est les disques qui sont utilisés sur la machine. On a utilisé une ListBox pour les afficher. La fonction *GetDisqueInfos()* nous permet de récupérer le nom du disque, le format, la taille disponible et la taille totale. Nous avons crée une classe Disque car pour chaque disque utilisé sur la machine on crée un nouveau Disque. On le range ensuite dans une liste avec la boucle foreach et on affiche la liste

3.2 Comparaison avec le XAML

L'idée de base du projet était de faire un programme en XAML. Vous nous avez conseillé de faire un programme sous Forms et de le comparer avec le programme sous XAML. Le code et les Class .cs sont en fait les mêmes. C'est au niveau de l'agencement de la partie graphique qu'il y'a des changements. Le XAML fonctionne comme le HTML ou XML. C'est du code par balise et, nous le trouvons, plus facile à utiliser et intuitif que le Forms. De plus, nous avons réussi sous XAML à faire tourner l'image d'une aiguille, ce que nous n'avons pas su faire avec un Forms. Nous avons juste rajouté ces deux lignes :

FIGURE 3.2 – Lignes de code pour faire tourner l'aiguille sur le compteur

```
RotateTransform rotation = new RotateTransform((val * 2.7f) - 90);
aiguilleimg.RenderTransform = rotation;
```

Soit un angle $\alpha = ouverturedel'aiguille$ $\alpha = -90° \Rightarrow$ CPU $= 0\%$
$\alpha = 180° \Rightarrow$ CPU $= 100\%$
Donc, on a une ouverte de $180 + 90 = 270° \rightarrow 100\%$ L'aiguille bouge donc de 2,7° pour 1%.

On définit la rotation avec un la Class RotateTransform qui prend en paramètre la valeur de la rotation. Soit $val * 2,7f$ et vu qu'on démarre à -90° on l'ajoute. Enfin on applique cette rotation à notre image d'aiguille.

Chapitre 4

Création d'un document PDF

La première partie du projet est faite. On a réussi à récupérer toutes les informations que l'on souhaitait. Maintenant on doit créer un fichier PDF, faire la mise en page puis le remplir avec toutes nos données.

Nous avons donc investigué sur la création de fichier depuis le C#. Comme souvent, on a du rajouter des ressources qui nous permettent de faire cela. Nous avons installé un outil qui permet de générer des documents au format PDF depuis le logiciel Visual Studio. Grâce à l'"extention" du nom de **iTextShap**.
Nous nous sommes renseignés sur la manière d'utiliser iTextSharp sur ce site iTextSharp. Voici donc comment nous avons procédé :

— Déclarer une variable outFile qui contiendra l'emplacement du nouveau fichier PDF. Grâce à **Environnement.CurrentDirectory** nous pouvons obtenir l'emplacement de notre fichier actuel.
— On crée un nouvel objet de la classe Document de type de Document.
— Pour pouvoir travailler avec ce fichier on doit appeler un PDFWriter et utiliser la méthode *GetInstance()*. On lui spécifie le document dans lequel on souhaite travailler, on crée ensuite le document grâce à new *FileStream* à l'emplacement que l'on souhaite
— Ensuite il faut apporter l'esthétique nécessaire à la création d'un document PDF. On instancie des nouveaux objets couleur de la classe BaseColor. On rentre dans les paramètres du constructeur les valeurs que l'on souhaite pour obtenir la couleur bleu par exemple. Ensuite il faut créer des polices avec la classe Font. On crée donc une nouvelle police (grâce à using itextSharp et pas using.Drawing!!) qui prendra en paramètre du constructeur : le type de police souhaité qu'on récupère par iTextSharp, la taille de la police en float, si on veut que le texte soit en gras ou souligné et la couleur de la police.

Nous nous sommes basé sur plusieurs fiches techniques de vendeur d'ordinateur. Tout d'abord pour savoir quelles informations récupérer. Et Ensuite pour connaitre la manière dont sont organiser les informations sur le fichier PDF. Nous nous sommes rendus compte que le plus simple était de faire un tableau et de stocker toutes les informations qui nous sont utiles à l'intérieur. De cette manière le fichier PDF sera lisible facilement et compréhensible par tous. Voilà un exemple de fiche technique sur laquelle nous nous sommes basés :

FIGURE 4.1 – Exemple de fichie technique d'un Ordinateur lambda

Processeur	Intel Centrino M730 1,6 Ghz FSB 533
Chipset graphique	Intel GMA 900 -PCI-Express
Disque dur	Samsung 60 Go 4200tr/min
Mémoire	2x 256Mo DDR
Ecran	12,1" TFT WXGA 16:10 bright résolution 1280x800
Connections	Wifi Intel Pro / wireless 2200 b/g; Ethernet 10/100, modem V90, Firewire, 3 USB2, PCMCIA type II
Carte son	Realtek AC97 ICH6 HD intégrée
Batterie	li-ion 4400mhA
Lecteur Optique	Graveur DVD multi-formats (DVD -R; DVD +RW)
Dimensions	30,2 x 24,8 x 25 mm
poids	1,9 kg (avec batterie)
Prix	899 €
garantie	1 an
Logiciels	home cinéma Suite, Ahead Nero Buning Rom
Système d'exploitation	Microsoft Windows XP Home edition sp2 OEM

4.1 Ranger le tableau de données dans le PDF

Ca y est ! On a les informations, on a créé le PDF. Il faut maintenant ranger toutes ces infos dans un tableau et l'ajouter au fichier. Grâce à la Class `ParametresPC` et `CreationTableau` on range les informations dans un tableau qu'on appelle dans la méthode **GenererPDF()**. On fait ensuite une simple boucle `For` pour remplir le tableau. C'est grâce à ce genre de manipulation que l'on comprend toute la puissance du C#.

FIGURE 4.2 – Création du tableau dans le fichier PDF

```
ParametresPC var = new ParametresPC();

string[] montab = CreationTableau.TableauDeDonnees(var);
for (int i = 0; i < montab.Length; i++)
{
    tableau.AddCell(montab[i]);
}

pdf.Add(tableau);
```

4.2 Protection avec un mot de passe

Nous nous sommes ammusés à protéger notre PDF avec mot de passe choisi par l'utilisateur. Blog

Chapitre 5

Conclusion

Ce projet nous a permis de comprendre un peu mieux toutes la puissance et la porté de l'Orientée Objet. On s'est rendu compte qu'avec le langage C on est vite limité. Le C# est un langage très puissant et permet beaucoup de possibilités.

De plus, nous avons essayé de créer un programme utile que n'importe qui peut utiliser. En un click l'utilisateur a à sa disposition tout un tas d'information à propos de sa machine.

Nous retenons une chose essentielle en programmation : `Il faut toujours chercher et investiguer pour des solutions`. On peut faire pleins de chose et il faut tout le temps rechercher et analyser du code en ligne pour comprendre et l'implémenter dans son propre projet. [1].

Nous retirons que du bénéfice de ce projet. Les pistes d'améliorations sont nombreuses : afficher plus d'informations, lien avec une base de données pour les mises à jour des différents pilots et drivers, proposer à l'utilisateur de nettoyer son ordinateur des fichiers inutiles...

Nom	Temps [H]
Guillaume Deterville	25
Lucas Teixeira	15

1. Une vidéo de 30 secondes qui résume bien la vie de codeur Vidéo Youtube