



Procédure – 07/06/2022

Titre : Mise en place d'un débogueur et affichage avec une application Windows

Table des matières

I. Introduction	2
1. Formulation du projet.....	2
2. Technologies utilisées	2
3. Etapes à réaliser	3
II. Mise en place.....	3
1. Débogueur	3
a) Prise en main de python	3
b) Utilisation de Kernel32	4
c) Prise en main de Pydbg	8
d) Ecrire les détails en base de données.....	9
2. Application Windows	11
a) Réalisation de maquettes	11
b) Prendre en main Visual Studio	11
c) Comprendre Crystal Reports	11
d) Création et importation de la base de données	13
e) Interface de l'application	15
f) Générer un exécutable	22
III. Conclusion	23

I. Introduction

1. Formulation du projet

Dans le cadre de notre stage, nous avons eu la possibilité de pouvoir travailler sur deux lieux différents. Nous avons travaillé le premier mois dans l'établissement de Samoreau, et le deuxième dans l'établissement de Monnetier-Mornex.

Cela fait plusieurs mois que l'établissement de Monnetier-Mornex fait face à des problèmes de « crash » d'application. De plus il était impossible d'avoir un suivi de ces crashes et donc de savoir à quelle heure ils ont eu lieu et sous quels utilisateurs.

Notre mission a donc été de produire un programme capable de pouvoir retracer l'historique de l'application qui s'est arrêté. Pour cela, on utilise un « Débogueur ».

Un Débogueur permet de visualiser l'état du programme en cours d'exécution. On en utilise par exemple lorsque l'on veut déboguer un code et voir si celui-ci est fonctionnel.

Pour nous aider à la mise en place d'un débogueur, nous avons reçu un document assez ancien de M. SILVA démontrant les étapes à suivre pour construire un débogueur.

L'idée est qu'une fois que nous récupérons les informations sur un programme, nous les stockons en base de données afin de par la suite les afficher via une application Windows

Une application Windows est une application qui a été créée avec l'aide de Windows Forms via l'IDE (environnement de développement intégré) Visual Studio. Grâce à cette IDE nous pouvons, lorsque l'application est terminée, générer un .EXE qui sera un exécutable de notre programme.

2. Technologies utilisées

Pour réaliser à bien ce projet, nous utilisons des outils spécifiques. Avant de présenter ces outils, il faut bien comprendre que notre projet s'apparente à de l'Hacking. En effet, on doit pouvoir avoir un visuel sur toutes les valeurs d'un programme à un temps donné. Des outils ont donc été déjà créés afin de répondre à ce besoin, mais ici, on souhaite développer le débogueur de A à Z. Pour réaliser ce « débogueur » nous allons utiliser Python.

Dans un premier temps, nous allons utiliser kernel32.dll qui est un composant logiciel de Microsoft Windows. Grâce à quoi nous aurons la possibilité de retracer en temps réel les processus qui se lancent.

Dans un second temps, il est nécessaire de regarder l'état et les valeurs des zone mémoire (registres) des programmes afin de pouvoir en déduire la cause du crash. Pour cela, il existe une bibliothèque qui s'appelle « Pydbg ».

Cette bibliothèque permet d'observer, lire et manipuler des données du processus cible en temps réel.

Enfin, pour un affichage ergonomique et simple, nous avons opté pour une application Windows qui utilisera plusieurs services externes comme MySQL, et Crystal Reports.

3. Etapes à réaliser

Avant de démarrer directement le projet, on a besoin de savoir par où commencer. C'est pour cela qui est important de réfléchir à la démarche pour réaliser ce projet. Etant donné qu'on connaît le résultat et qu'on a une liberté totale d'organisation, les étapes peuvent être rapidement déduites :

Débogueur :

1. Prise en main de python
2. Utilisation de kernel32
3. Prise en main de Pydbg
4. Ecrire les détails des registres en base de données

Application Windows :

1. Réalisation de maquettes
2. Prendre en main Visual Studio (C#)
3. Comprendre Crystal Reports
4. Création et importation base de données
5. Créer l'interface d'application
6. Générer un .EXE

Voici donc les étapes théoriques qu'on censé faire pour réaliser notre projet.

II. Mise en place

1. Débogueur

a) Prise en main de python

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels.

Python a la chance d'être un logiciel facile à apprendre et à utiliser. Ses caractéristiques sont peu nombreuses, ce qui permet de créer des programmes rapidement et avec peu d'effort.

La plupart des langages utilisent des accolades pour définir la portée d'un bloc de code, mais l'interpréteur de Python le détermine simplement par une indentation. Cela signifie qu'il faut être particulièrement prudent avec les espaces blancs dans le code, ce qui peut interrompre le fonctionnement de l'application.

Les commentaires eux aussi sont quelque peu différent, nous avons l'habitude de commenter nos lignes de code grâce à « // » or, en Python, les commentaires sont effectués avec « # ».

b) Utilisation de Kernel32

Kernel32.dll est le nom de la bibliothèque d'instructions qu'utilise Windows pour gérer l'écriture et la lecture de données dans la mémoire vive de votre micro, ou dans son fichier de mémoire virtuelle. Pour fonctionner, ce composant de Windows se réserve un espace de mémoire vive bien à lui. A chaque fois qu'un autre logiciel tente d'y écrire des données, il se produit une erreur qui plante Windows.

Pour apprendre à utiliser cette bibliothèque nous nous sommes aidé du document fournit par Warren SILVA. Ainsi on a pu rapidement construire notre premier débogueur en Python :

```
from ctypes import *
from pickle import FALSE
from mes_definitions_debugger import *
from mes_definitions_debugger_final import DBG_CONTINUE, DEBUG_EVENT,
INFINITE, PROCESS_ALL_ACCESS

kernel32=windll.kernel32

class debugger():
    def __init__(self):
        self.h_process=None
        self.pid=None
        self.debugger_active=False
        pass

    def load(self,path_to_exe):
        creation_flags=DEBUG_PROCESS
        startupinfo=STARTUPINFO()
        process_information=PROCESS_INFORMATION()

        startupinfo.dwFlags=0x1
        startupinfo.wShowWindows=0x0
        startupinfo.cb=sizeof(startupinfo)

        if kernel32.CreateProcessA(path_to_exe,None,None,None,None,
creation_flags,None,None,byref(startupinfo),
byref(process_information)):

            print ("[*] Nous avons lance le process !" )
            print ("[*] PID: %d"%process_information.dwProcessId )

        else:
            print ("[*] Erreur: 0x%08x." % kernel32.GetLastError())

        print("[*] Nous avons attache le processus avec succes")
        print("[*] PID: %d"%process_information.dwProcessId)
        self.h_process=self.open_process(process_information.dwProcessId0)
```

```

def open_process(self,pid):
    h_process=kernel32.OpenProcess(PROCESS_ALL_ACCESS,pid,False)
    return h_process

def attach(self,pid):
    self.h_process= self.open_process(pid)
    if kernel32.DebugActiveProcess(pid):
        self.debugger_active=True
        self.pid=int(pid)
        self.run()
    else:
        print("[*] impossible d'attacher le processus")

def run(self):
    while self.debugger_active==True:
        self.get_debug_event()

def get_debug_event(self):
    debug_event = DEBUG_EVENT()
    continue_status = DBG_CONTINUE
    if kernel32.WaitForDebugEvent(byref(debug_event),INFINITE):
        input("Appuyer sur une touche pour continuer ....")
        self.debugger_active=False
        kernel32.ContinueDebugEvent(debug_event.dwProcessId,debug_event.
dwThreadId,continue_status)

def detach (self):

    if kernel32.DebugActiveProcessStop(self.pid):
        print ("[*] débogage terminer. sortie ...")
        return True

    else:

        print("Il y a une erreur")
        return False

```

Pour tester ce débogueur, nous avons créé un « Main » simple :

```

import mon_debugger
debugger=mon_debugger.debugger()
debugger.load("C:\\WINDOWS\\system32\\calc.exe")

```

Ce « Main » a pour but de d'attacher le processus à la calculatrice.
Voici la sortie

```

[*] Erreur: 0x00000002.
[*] Nous avons attache le processus avec succes
[*] PID: 0

```

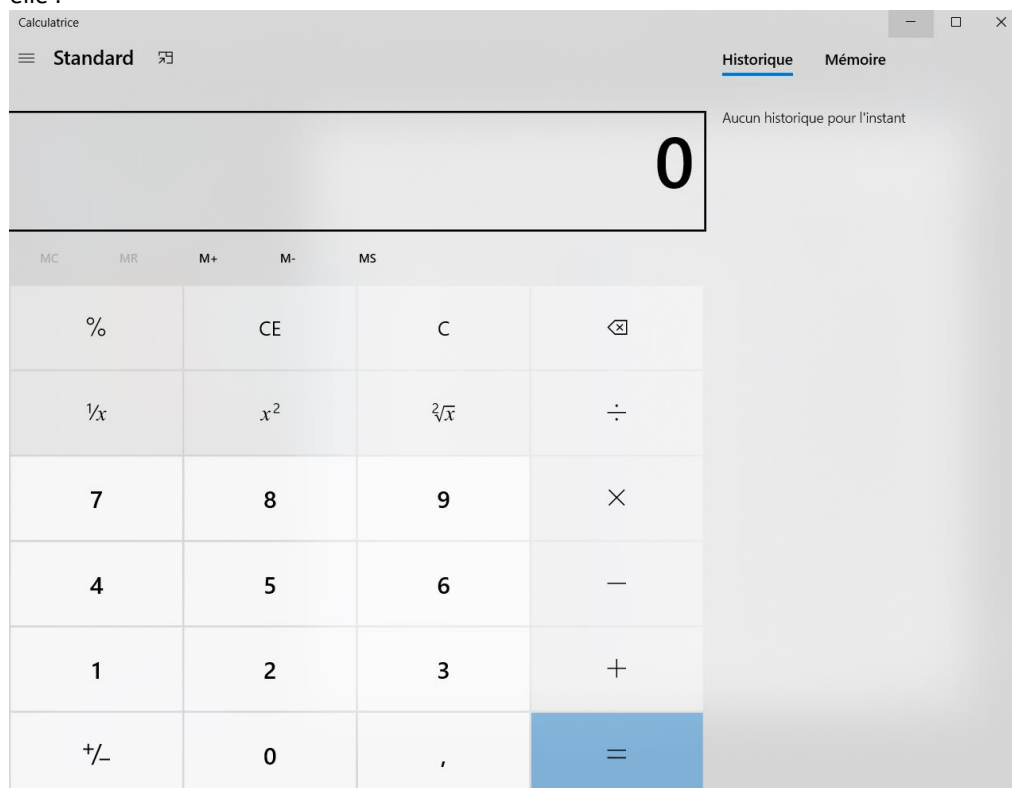
Une fois ce petit test réalisé nous avons faire d'autre test :

```
import mon_debugger2
debugger=mon_debugger2.debugger()
pid=input("entrez le PID du processus a attacher: ")
debugger.attach(int(pid))
debugger.detach()
```

Sortie :

```
entrez le PID du processus a attacher: 596
Appuyer sur une touche pour continuer...
[*] debugage termine. sortie...
```

Maintenant nous avons réalisé un test pour figer la calculatrice, autrement dit on ne peut plus interagir avec elle :



Impossible de d'interagir avec la calculatrice. Si l'on ferme le test alors la calculatrice se ferme aussi.

Vous remarquez qu'à chaque fois, on importe : `mes_definitions_debugger import *`
Voici un petit extrait de ce fichier qui est relativement complexe :

```

from ctypes import *

# Let's map the Microsoft types to ctypes for clarity
BYTE      = c_ubyte
WORD      = c_ushort
DWORD     = c_ulong
LPBYTE    = POINTER(c_ubyte)
LPTSTR    = POINTER(c_char)
HANDLE    = c_void_p
PVOID     = c_void_p
LPVOID    = c_void_p
UINT_PTR  = c_ulong
SIZE_T    = c_ulong

# Constants
DEBUG_PROCESS      = 0x00000001
CREATE_NEW_CONSOLE = 0x00000010
PROCESS_ALL_ACCESS = 0x001F0FFF
INFINITE           = 0xFFFFFFFF
DBG_CONTINUE       = 0x00010002
DBG_EXCEPTION_NOT_HANDLED = 0x80010001

# Debug event constants
EXCEPTION_DEBUG_EVENT      = 0x1
CREATE_THREAD_DEBUG_EVENT  = 0x2
CREATE_PROCESS_DEBUG_EVENT = 0x3
EXIT_THREAD_DEBUG_EVENT    = 0x4
EXIT_PROCESS_DEBUG_EVENT   = 0x5
LOAD_DLL_DEBUG_EVENT       = 0x6
UNLOAD_DLL_DEBUG_EVENT     = 0x7
OUTPUT_DEBUG_STRING_EVENT  = 0x8
RIP_EVENT                  = 0x9

# debug exception codes.
EXCEPTION_ACCESS_VIOLATION = 0xC0000005
EXCEPTION_BREAKPOINT       = 0x80000003
EXCEPTION_GUARD_PAGE       = 0x80000001

```

....

c) Prise en main de Pydbg

La prise en main de Pydbg a été relativement complexe. Bien que cette bibliothèque propose des services simples, il était difficile de trouver des informations fiables sur ce sujet. De plus, cette bibliothèque est assez ancienne et donc elle est actuellement dépréciée c'est-à-dire que python ne prend plus en charge ce service.

Nous avons alors deux possibilités :

- Trouver une version qui serait mis à jour par des utilisateurs et que l'on pourrait utiliser
- Changer notre version de python (descendre en version) pour retrouver un pydbg qui serait compatible

Nous nous sommes tout d'abord orientés vers la première solution. En parcourant beaucoup de Github nous avons trouvé plusieurs Pydbg. Nous avons donc installé cette bibliothèque mais plusieurs problèmes ont été rencontrés :

-Le fichier de Pydbg était en ancien python et donc il y'avait beaucoup d'erreurs de syntaxe.

-Il faut aussi installer le module « Pydasm » qui permet le bon fonctionnement de Pydbg.

Nous avons donc tout désinstaller pour tout recommencer avec cette fois une installation de Pydasm. Après toutes les installation terminée (ce fut long) le programme ne marchait toujours pas.

Il nous restait donc plus qu'une chose à faire : changer notre version de python.

Nous avons désinstallé python pour le réinstaller en 2.7 (nous avions la 3.10). On réitère les mêmes manipulations mais on arrive au même constat : Cela ne marche pas

Cela faisait environ une semaine entière que nous essayions d'installer un module mais sans succès. Par peur de rendu, notre responsable Warren SILVA et moi-même nous sommes laissés une semaine pour régler le problème, sans succès.

Nous nous sommes donc orientés vers la mise en place de l'application Windows pour revenir par la suite sur notre « Débogueur ».

Voici tout de même le code du test de Pydbg :

```
from pydbg import *

import struct
import random

def printf_randomizer(dbg):

    # Lecture de la valeur du compteur en ESP + 0x8 comme un DWORD
    parameter_addr = dbg.context.Esp + 0x8
    counter = dbg.read_process_memory(parameter_addr,4)
    counter = struct.unpack("L",counter)[0]
    print ("Counter: %d" % int(counter))
    random_counter = random.randint(1,100)
    random_counter = struct.pack("L",random_counter)[0]
    dbg.write_process_memory(parameter_addr,random_counter)
```



```

return DBG_CONTINUE

dbg = pydbg()
pid = input("Enter the printf_loop.py PID: ")
dbg.attach(int(pid))
printf_address = dbg.func_resolve("msvcrt", "printf")
dbg.bp_set(printf_address, description="printf_address",
handler=printf_randomizer)
dbg.run()

```

Ici un extrait du fichier pydbg :

```

#####
def __init__(self, ff=True, cs=False):
    """
    Set the default attributes. See the source if you want to modify the default creation values.
    @type ff: Boolean
    @param ff: (Optional, Def=True) Flag controlling whether or not pydbg attaches to forked processes
    @type cs: Boolean
    @param cs: (Optional, Def=False) Flag controlling whether or not pydbg is in client/server (socket) mode
    """

    # private variables, internal use only:
    self._restore_breakpoint = None # breakpoint to restore
    self._guarded_pages = set() # specific pages we set PAGE_GUARD on
    self._guards_active = True # flag specifying whether or not guard pages are active

    self.page_size = 0 # memory page size (dynamically resolved at run-time)
    self.pid = 0 # debuggee's process id
    self.h_process = None # debuggee's process handle
    self.h_thread = None # handle to current debuggee thread
    self.debugger_active = True # flag controlling the main debugger event handling loop
    self.follow_forks = ff # flag controlling whether or not pydbg attaches to forked processes
    self.client_server = cs # flag controlling whether or not pydbg is in client/server mode
    self.callbacks = {} # exception callback handler dictionary
    self.system_dlls = [] # list of loaded system dlls
    self.dirty = False # flag specifying that the memory space of the debuggee was modified
    self.system_break = None # the address at which initial and forced breakpoints occur at
    self.peb = None # process environment block address

```

d) Ecrire les détails en base de données

Nous n'arrivons pas à utiliser Pydbg donc nous ne pouvons pas construire une base de données avec les informations de crash des applications. Cependant, comme nous devons avancer, nous avons créé une base de données test afin de pouvoir plus tard établir des requêtes.

Cette base de données possède le même contenu que la base de données qui sera créée avec la récupération des informations des programmes. Le but est de justement que si Pysbg est fonctionnel, on puisse juste s'appuyer sur cette base de données « Test ».

Cette base de données ne contient qu'une seule table, elle a donc été relativement simple à créer. Nous ne possédons pas de serveur, donc la création de cette base de données est créée en local sur notre pc actuel. À terme, la vraie base de données sera hébergée sur le serveur de l'établissement.

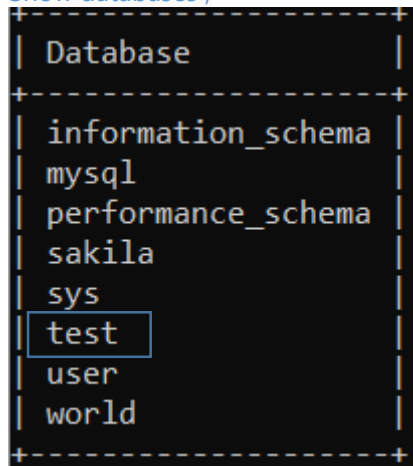
Pour ce faire, on a téléchargé MySQL puis créé notre base de données.

//creation de la BD :

```
CREATE DATABASE test
```

Si l'on affiche nos bases de données on remarque que la base de donnée « test » à bien été créée.

```
Show databases ;
```



Database
information_schema
mysql
performance_schema
sakila
sys
test
user
world

Maintenant créons la table :

```
Use test ; //on dit quelle base de données on souhaite utiliser
```

```
CREATE TABLE test1 ( ; //on créer notre table
```

```
Colonne1 varchar(20),
```

```
Colonne2 varchar(20),
```

```
Colonne3 int primary key ,
```

```
Etat tinyint(1)
```

```
)
```

```
;
```

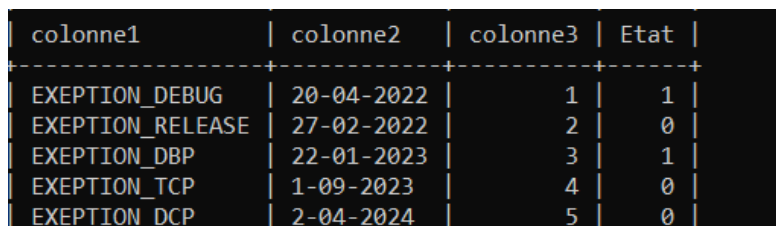
L'état est relativement important car il va nous permettre d'identifier les données qui ont été traitées et celle qui ne le sont pas encore.

La valeur de la colonne « Etat » peu prendre trois valeurs : NULL,0,1

Plus tard dans notre programme la valeur NULL sera remplacer automatiquement par un 0.

Maintenant il nous reste juste à peupler la table avec un « Insert »

```
INSERT INTO test1 values ('EXEPTION_1','20-02-2022',1,0) ;
```



colonne1	colonne2	colonne3	Etat
EXEPTION_DEBUG	20-04-2022	1	1
EXEPTION_RELEASE	27-02-2022	2	0
EXEPTION_DBP	22-01-2023	3	1
EXEPTION_TCP	1-09-2023	4	0
EXEPTION_DCP	2-04-2024	5	0

2. Application Windows

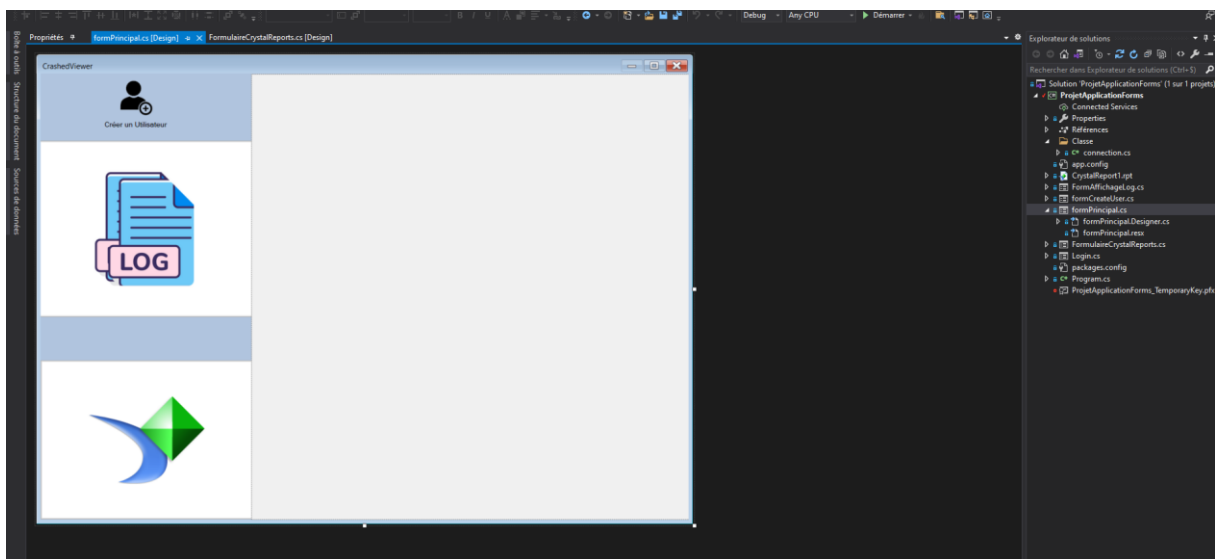
a) Réalisation de maquettes

Pour bien réaliser et appréhender l'interface nous commençons par la réalisation de maquettes pour avoir une idée du graphique des pages de l'application.

Voici le lien vers les maquettes : [ici](#)

b) Prendre en main Visual Studio

Cette étape est la plus rapide car la documentation ainsi que les tutoriels sur ce logiciel sont abondants. Le plus complexe est de rapidement prendre en main le projet « Windows Forms » afin de créer une application Windows. Mais là aussi la documentation est abondante. Voici un exemple de l'environnement de programmation :



c) Comprendre Crystal Reports

Crystal Reports est un progiciel d'informatique décisionnelle qui permet de générer une grande variété de rapports à partir de données informatiques.

Le plus dur avec Crystal Report a été l'importation des modules. De base Visual Studio nous permet d'ajouter directement un élément Crystal Report à n'importe quel formulaire. Cependant, les erreurs d'importation sont abondantes et pour cause nous sommes sur le mauvais Framework. En effet nous utilisons un Framework trop ancien pour utiliser ce progiciel. Nous devons donc tout recommencer, c'est-à-dire créer un nouveau projet mais cette fois en faisant bien attention de choisir le bon Framework

Une fois cela effectuer nous sommes en capacité d'ajouter un élément Crystal Report à notre projet :

The screenshot displays the Visual Studio IDE with a Crystal Report project open. The main window shows the report design, which includes a header section with the 'Fondation Cognacq-Jay' logo and a title 'Rapport des derniers Crash'. Below the header is a section titled 'Récapitulatif des Crashes' containing a table with columns for 'Nombre de Crash', 'Résolu', and 'En Attente'. The table data is as follows:

Nombre de Crash	Résolu	En Attente
{COUNT(*)}	{Count(*)}	{shResolve}

Below the summary is a section titled 'Détail des derniers Crashes'. The bottom of the report design shows a footer section with columns labeled 'colonne1', 'colonne2', 'colonne3', and 'Etat'.

The Solution Explorer on the left shows the project structure for 'Solution 'ProjetApplicationForms' (1 sur 1 projets)'. The project 'ProjetApplicationForms' is expanded, showing a list of files and folders. The file 'CrystalReport1.rpt' is highlighted in yellow, indicating it is the active report.

- Solution 'ProjetApplicationForms' (1 sur 1 projets)
 - ProjetApplicationForms
 - Connected Services
 - Properties
 - Références
 - Classe
 - connection.cs
 - app.config
 - CrystalReport1.rpt
 - FormAffichageLog.cs
 - formCreateUser.cs
 - formPrincipal.cs
 - formPrincipal.Designer.cs
 - formPrincipal.resx
 - FormulaireCrystalReports.cs
 - Login.cs
 - packages.config
 - Program.cs
 - ProjetApplicationForms_TemporaryKey.pfx

d) Création et importation de la base de données

L'application Windows récupère ces données dans deux bases de données distinctes :

- Logs (créer lors de la phase de débogage)
- Utilisateur (encore inexistant)

La base de données « Logs » est pour nous notre BD « test ».

Il faut donc créer une deuxième base de données avec une table qui contient les utilisateurs ayant accès à l'application.

Autrement dit la base de données contient les informations de connexion de chaque utilisateur de l'application. Nous devons donc chiffrer les informations sensibles comme le mot de passe pour une meilleure sécurité. Pour cela on va utiliser un algorithme déjà fourni par « MySQL » qui est le « MD5 ». Le MD5 permet de pouvoir chiffrer facilement une information. Nous rappelons qu'un texte chiffré ne peut pas être déchiffré. Pour chiffrer des données facilement nous avons juste à utiliser la fonction MD5('Password ') lors de l'insertion d'un enregistrement.

Maintenant nous possédons une base de données avec les informations de connexion pour les utilisateurs. Il nous reste juste à importer cette base de données dans notre projet. Pour cela nous sommes obligés de passer par ODBC. De base on ne peut pas récupérer une BD MySQL en localhost (héberger sur notre appareil) nous devons donc utiliser MySQL connecteur pour faire une passerelle de MySQL vers ODBC.

Une fois cela fait, on remarque que nous avons nos bases de données qui sont synchronisées sur ODBC :

Sources de données utilisateur :

Nom	Plate-forme	Pilote	
Excel Files	64 bits	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)	Ajouter...
MS Access Database	64 bits	Microsoft Access Driver (*.mdb, *.accdb)	Supprimer
test	32/64 bits	MySQL ODBC 8.0 Unicode Driver	Configurer...
Users	32/64 bits	MySQL ODBC 8.0 Unicode Driver	

Maintenant nous pouvons établir une connexion vers nos bases de données :

```
void Requete()
{
    MySqlConnection connexion = new MySqlConnection
    {
        ConnectionString = "SERVER=localhost;" + "DATABASE=test;" + "UID=*****;" + "PASSWORD=*****;"
    };
    connexion.Open();
    MySqlCommand command = connexion.CreateCommand();
    command.CommandText = "SELECT * FROM test1";
    donnee.SelectCommand = command;
    DataSet dataset = new DataSet();
    donnee.Fill(dataset);
    Tableau.DataSource = dataset.Tables[0];
    connexion.Close();

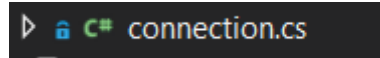
    GererStyleTableau();
}
```

Pour pouvoir utiliser MySqlConnection nous devons importer le module MySqlConnector :

```
using MySqlConnector;
```

Pour la base de données des utilisateurs, nous changeons de stratégie puisqu'il faut qu'on puisse chiffrer directement le mot de passe rentré lors du formulaire de login afin de voir si celui-ci correspond au mot de passe enregistré dans la BD.

Pour cela, nous allons créer une classe « connection.cs »



Cette classe permet de rendre plus facile les requêtes et donc à terme de gagner du temps :

```
class Connection
{
    MySqlConnection conn;

    static string host = "localhost";
    static string database = "User";
    static string userDB = "*****";
    static string password = "*****";
    public static string strProvider = "server=" + host + ";Database=" + database + ";User ID=" + userDB + ";Password=" + password;
    2 références | Xan Maris, il y a 11 jours | 1 auteur, 1 modification
    public bool Open()
    {
        try
        {
            strProvider = "server=" + host + ";Database=" + database + ";User ID=" + userDB + ";Password=" + password;
            conn = new MySqlConnection(strProvider);
            conn.Open();
            return true;
        }
        catch (Exception er)
        {
            MessageBox.Show("Connection Error ! " + er.Message, "Information");
        }
        return false;
    }
    1 référence | Xan Maris, il y a 11 jours | 1 auteur, 1 modification
    public void Close()
    {
        conn.Close();
        conn.Dispose();
    }
    0 références | Xan Maris, il y a 11 jours | 1 auteur, 1 modification

    public DataSet ExecuteDataSet(string sql)
    {
        try
        {
            DataSet ds = new DataSet();
            MySqlDataAdapter da = new MySqlDataAdapter(sql, conn);
            da.Fill(ds, "result");
            return ds;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        return null;
    }
    1 référence | Xan Maris, il y a 11 jours | 1 auteur, 1 modification
    public MySqlDataReader ExecuteReader(string sql)
    {
        try
        {
            MySqlDataReader reader;
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            reader = cmd.ExecuteReader();
            return reader;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        return null;
    }
}
```

```

1 reference | Xan Maris, il y a 11 jours | 1 auteur, 1 modification
public int ExecuteNonQuery(string sql)
{
    try
    {
        int affected;
        MySqlConnection mytransaction = conn.BeginTransaction();
        MySqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = sql;
        affected = cmd.ExecuteNonQuery();
        mytransaction.Commit();
        return affected;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    return -1;
}

```

Maintenant nous pouvons utiliser plus facilement les requêtes au travers de nos différentes pages.

e) Interface de l'application

Grâce aux différentes maquettes qui ont été créées, l'interface de l'application a été relativement rapide. Pour chaque page de l'application, on peut ajouter des éléments graphiques puis gérer leur comportement via du C#.

Lorsque l'on ouvre l'application, une page de login s'affiche :

The screenshot shows a Windows-style application window titled "Authentification". The window has a light blue background. It contains two text input fields: "Nom d'utilisateur" and "Mot de passe". Below these fields is a button labeled "Se connecter". The window has a standard Windows title bar with a close button (X) in the top right corner.

Voici le code de validation d'authentification :

```
public partial class Login : Form
{
    MD5 md5 = MD5.Create();
    connection con = new connection();

    string username, password, firstname, lastname, address;
    string Admin;
    1 référence | Xan Maris, il y a 8 jours | 1 auteur, 1 modification
    public Login()
    {
        InitializeComponent();
    }
}
```

//Attribut de la classe « Login »

```
1 référence | Xan Maris, il y a 8 jours | 1 auteur, 1 modification
private void button1_Click(object sender, EventArgs e)
{
    //UTILISATION DE MD5 POUR CHIFFRER MDP
    byte[] b = Encoding.ASCII.GetBytes(InputPassword.Text);
    byte[] hash = md5.ComputeHash(b);

    StringBuilder sb = new StringBuilder();
    foreach(var a in hash)
    {
        sb.Append(a.ToString("X2"));
    }

    InputPassword.Text = sb.ToString();

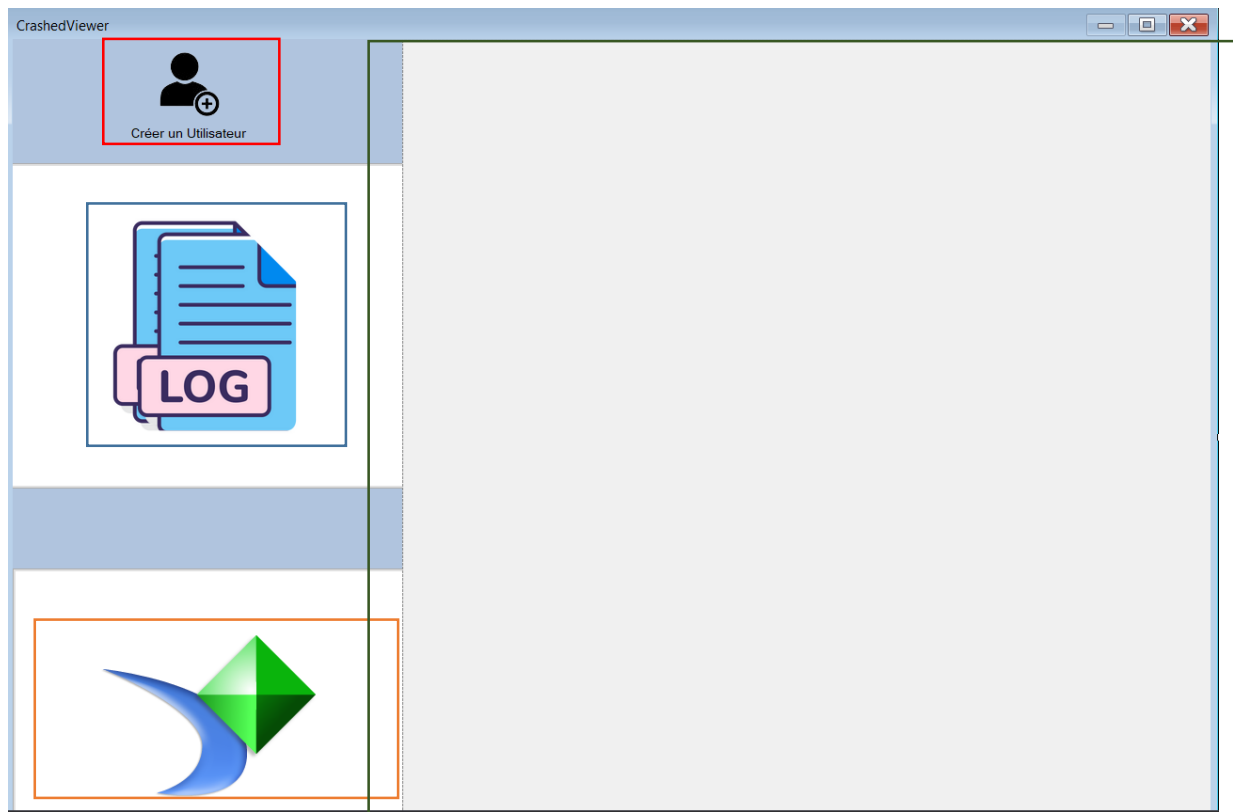
    //FIN CHIFFREMENT


    try
    {
        //EVALUATION DES DONNEES RENTREES
        if (InputUsername.Text != "" && InputPassword.Text != "")
        {
            //CONNECTION BD
            con.Open();
            //ON RECUPERE TOUS LES CHAMPS
            string query = "select username,password,firstname,lastname,address,Admin from Utilisateurs WHERE username =" + InputUsername.Text + " AND password =" + InputPassword.Text + ";";
            MySqlDataReader row;
            row = con.ExecuteReader(query);
            //POUR CHAQUE LIGNE RECUPERER DU SELECT ON VA LES METTRE DANS UNE VARIABLE
            if (row.HasRows)
            {
                while (row.Read())
                {
                    username = row["username"].ToString();
                    password = row["password"].ToString();
                    firstname = row["firstname"].ToString();
                    lastname = row["lastname"].ToString();
                    address = row["address"].ToString();
                    Admin = row["Admin"].ToString();
                }
                //SI UNE LIGNE A ETE TROUVER ON AFFICHE UN MESSAGE DE VALIDATION
                new ToastContentBuilder()
                    .AddArgument("action", "viewConversation")
                    .AddArgument("conversationid", 9813)
                    .AddText("vous etes connecté en tant que " + firstname + " " + lastname)
                    .Show();


                //ON OUVER L APPLICATION
                formPrincipal formPrincipal = new formPrincipal(Admin);
                this.Hide();
                formPrincipal.ShowDialog();
                this.Close();
            }
            else
            {
                //SI AUCUNE LIGNE A ETE TROUVER
                MessageBox.Show("Identifiant ou mot de passe erroné");

                //ON PREPARE LE MOT DE PASSE (POUR UNE NOUVEL SAISIE)
                InputPassword.Text = "";
            }
        }
        else
        {
            //SI LE MDP OU NOM UTILISATEUR EST VIDE
            MessageBox.Show("Username or Password is empty", "Information");
        }
    }
    catch
    {
        //SI LA CONNEXION / REQUETE NA PAS FONCTIONNEE
        MessageBox.Show("Connection Error", "Information");
    }
}
```



La page de login nous emmènera sur la page principale de l'application :



 : Bouton qui permet la saisie d'un nouvel utilisateur (ce bouton s'affiche UNIQUEMENT si on est connecté en temps qu'ADMIN).

 : Bouton qui nous affichera un tableau contenant les données de la BD « Test »

 : Bouton qui nous affichera un rapport des données contenu dans la BD « Test »

 : Panel qui affichera le contenu de d'autres formulaires : Crystal Reports et Logs

Voici le résultat :

CrashedViewer

Créer un Utilisateur

LOG

CrashedViewer

colonne1	colonne2	colonne3	Etat
EXEPTION_DEBUG	20-04-2022	1	<input checked="" type="checkbox"/>
EXEPTION_RELEASE	27-02-2022	2	<input type="checkbox"/>
EXEPTION_DBP	22-01-2023	3	<input checked="" type="checkbox"/>
EXEPTION_TCP	1-09-2023	4	<input type="checkbox"/>
EXEPTION_DCP	2-04-2024	5	<input type="checkbox"/>
test	test	6	<input checked="" type="checkbox"/>
test	test	7	<input checked="" type="checkbox"/>

//PAGE LOGS

CrashedViewer

Créer un Utilisateur

LOG

CrashedViewer

Appliquer Supprimer Rapport principal

SAP CRYSTAL REPORTS*

Imprimé le : 09/06/2022

Créé par :

Fondation
Cognacq-Jay

Rapport des derniers Crash

Récapitulatif des Crashes

Nombre de Crash :	7
Résolus :	4
En Attente :	3

Détail des derniers Crashes

EXEPTION_DEBUG	20-04-2022	1	1
EXEPTION_RELEASE	27-02-2022	2	0

//PAGE CRYSTAL REPORTS

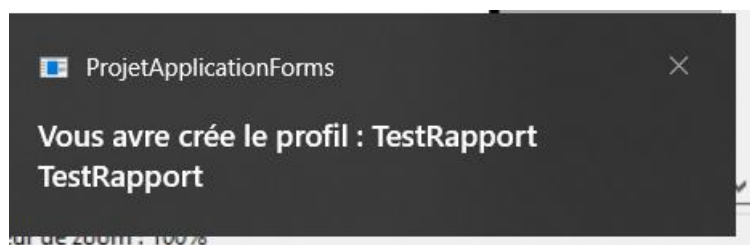
Maintenant intéressons-nous au formulaire de saisie d'un utilisateur qui apparaît après un clique sur le bouton :



Voici le formulaire qui s'affiche :

Une fenêtre de dialogue intitulée "Ajout d'un Utilisateur" avec une croix en haut à droite. À l'intérieur, une section "Création" contient six champs de saisie : "Prénom" (contenant "TestRapport"), "Nom" (contenant "TestRapport"), "Adresse" (contenant "TestRapport"), "Nom d'utilisateur" (contenant "TestRapport"), "Mot de passe" (remplis de points) et "Saisissez à nouveau votre MDP" (remplis de points). À droite de ces champs, il y a une section "Admin" avec une case à cocher cochée. En bas à droite, un bouton "Valider" est visible.

J'ai ici, créé un utilisateur qui s'appelle « TestRapport », regardons dans notre base de données.




Un message Windows apparaît pour nous confirmer que notre utilisateur a bien été créé.

Voici le contenu de notre base de données Utilisateurs :

Username	Password	FirstName	LastName	adress	Admin
Admin	e3afed0047b08059d0fada10f400c1e5	NULL	NULL	NULL	1
Technicien	61C42F9E5647205C90235B3361BE8AD7	Technicien	Technicien	Technicien	1
TestRapport	00ED5E657256BB7CE7DACFB84123D7A3	TestRapport	TestRapport	TestRapport	1
xadmin	b2a5abfeef9e36964281a31e17b57c97	Xan	Maris	19 rue roland	NULL

On remarque que notre utilisateur à bien été enregistré en tant qu'admin, on constate aussi que le mot de passe est bien chiffré.

Connectons-nous avec cet utilisateur :


Authentification
✕

Nom d'utilisateur

Mot de passe

CrashedViewer



Créer un Utilisateur



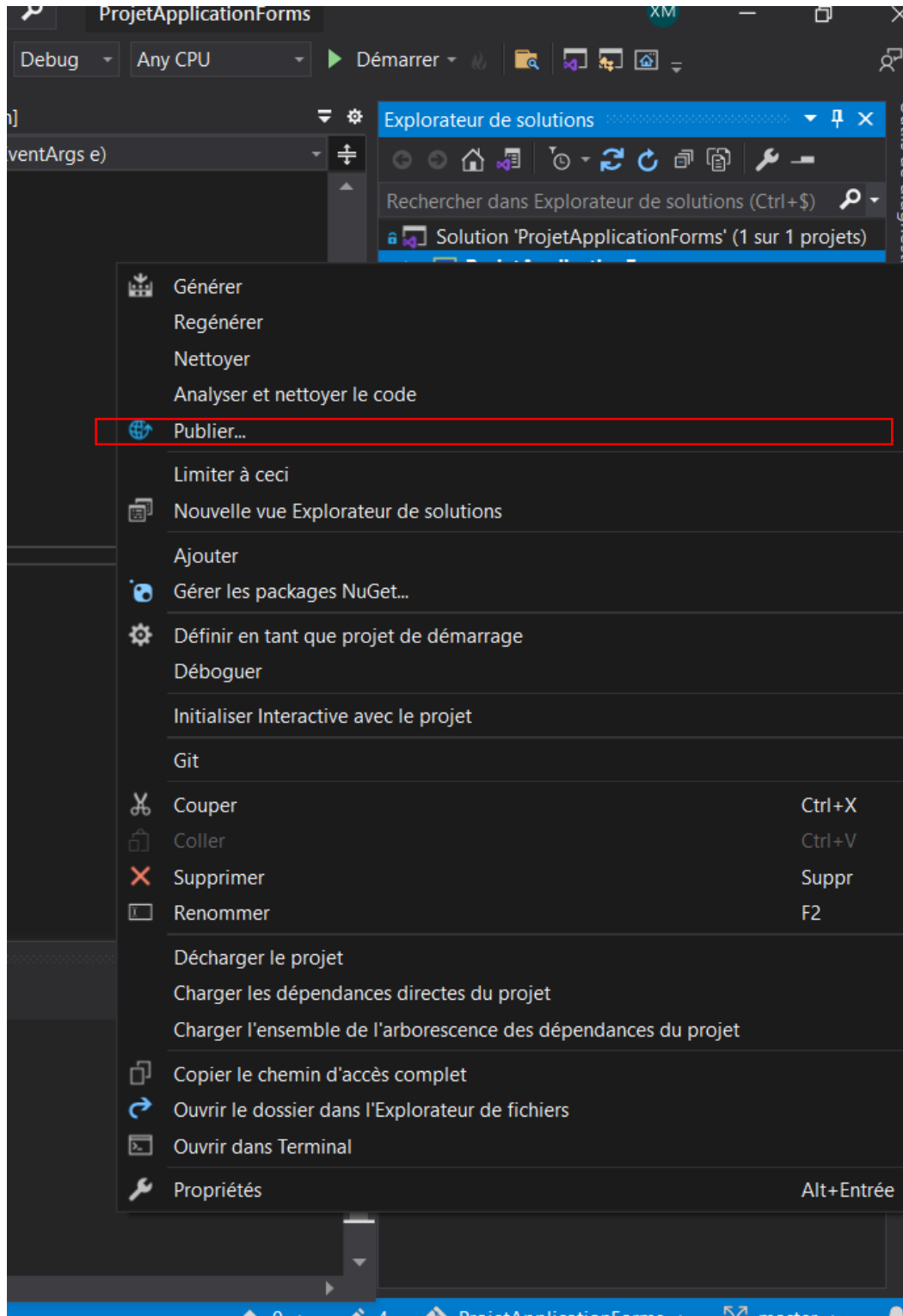
	colonne1	colonne2	colonne3	Etat
▶	EXEPTION_DEBUG	20-04-2022	1	<input checked="" type="checkbox"/>
	EXEPTION_RELEASE	27-02-2022	2	<input type="checkbox"/>
	EXEPTION_DBP	22-01-2023	3	<input checked="" type="checkbox"/>
	EXEPTION_TCP	1-09-2023	4	<input type="checkbox"/>
	EXEPTION_DCP	2-04-2024	5	<input type="checkbox"/>
	test	test	6	<input checked="" type="checkbox"/>
	test	test	7	<input checked="" type="checkbox"/>

ProjetApplicationForms

Vous etes connecté en tant que TestRapport
TestRapport

f) Générer un exécutable

Il est facile de générer un exécutable. Pour ce faire on va sur notre projet puis « Publier »



Puis, nous avons juste à choisir le chemin d'installation.

III. Conclusion

Ce projet m'a demandé beaucoup de technicité. Il fallait que je comprenne rapidement l'utilisation des outils afin de proposer une solution efficace. La partie formulaire a été conçue assez rapidement car j'avais déjà des connaissances en C#, il n'y donc que l'interface pour lequel j'ai dû réfléchir. Pour la partie débogueur c'était un peu plus compliqué car il fallait bien comprendre le langage système et comprendre comment marche les programmes. De plus, j'ai dû faire face à des problèmes d'importation de notre outil principal et donc je n'ai pas pu finir ce projet.

Enfin, ce projet à été complet puisque j'ai pu développer à la fois une interface, mais aussi des actions/événement de cette interface. J'ai pu donc mettre en pratique mes cours d'IHM et développement de ma première année.

Pour conclure, d'un point de vue personnel, j'ai beaucoup aimé ce projet...