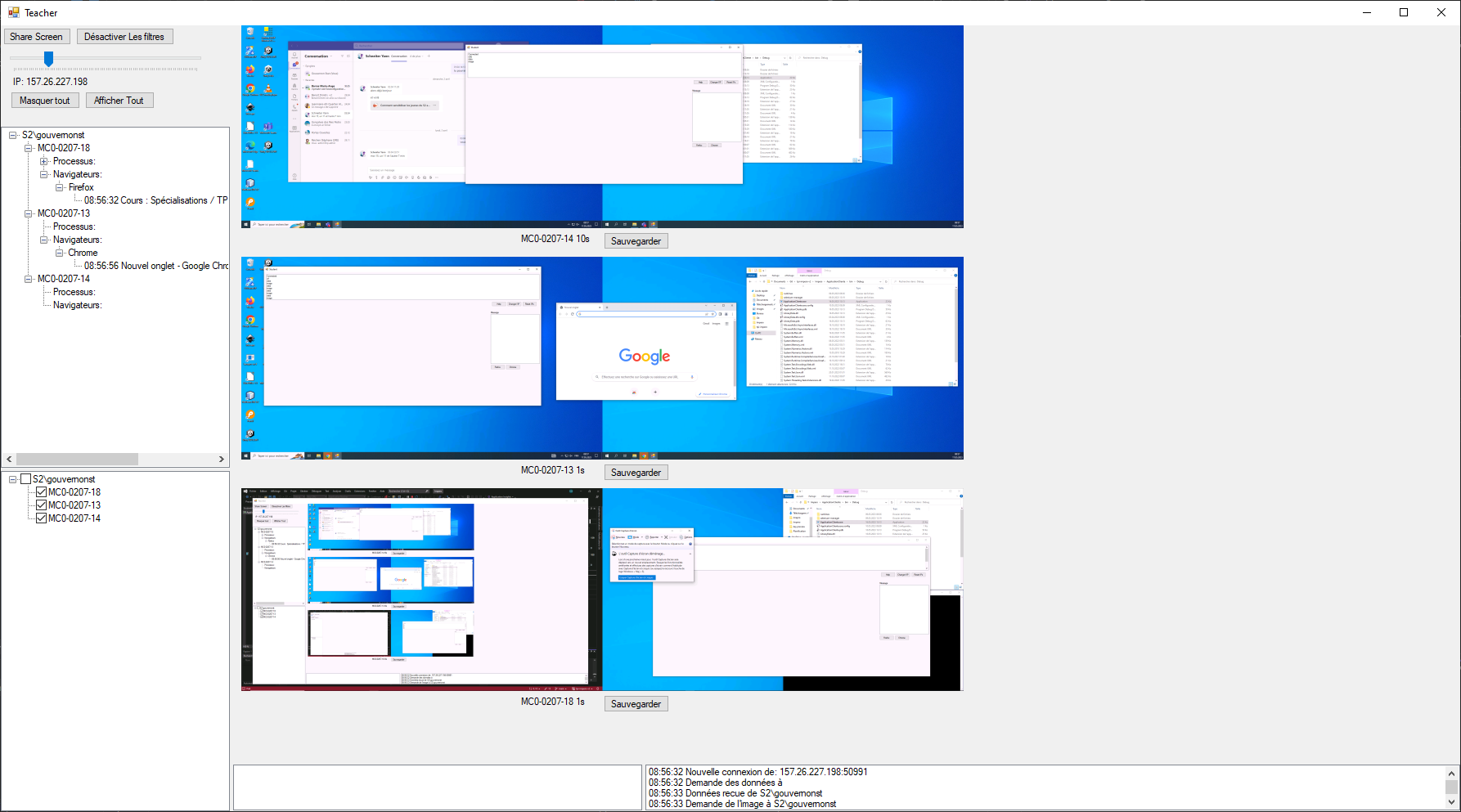
Imepro



Application de monitoring et d’aide des élèves

École : CPNE-TI, École technique

Atelier : Salle 207

Langage : C#

Auteur : Stan Gouvernon

Supérieur : Alain Jeanmaire

Experts : Daniel Rebetez

Morgan Del Torchio

Date : 24.04.2023-17.05.2023

Table des matières

[2 Présentation du projet 5](#_Toc135226007)

[2.1 Prérequis 5](#_Toc135226008)

[2.1.1 Matériel et logiciels 5](#_Toc135226009)

[2.1.2 Connaissances requises 5](#_Toc135226010)

[2.2 Cahier des charges 5](#_Toc135226011)

[2.2.1 Connexion 5](#_Toc135226012)

[2.2.2 Interface graphique (IHM) principale du professeur 5](#_Toc135226013)

[2.2.3 Filtrage de l’affichage des processus 6](#_Toc135226014)

[2.2.4 Capture d’écran 7](#_Toc135226015)

[2.2.5 Diffusion multicast depuis le poste professeur 7](#_Toc135226016)

[2.2.6 Streaming monoposte depuis l’élève, contrôle à distance 7](#_Toc135226017)

[2.2.7 Portabilité de l’élève 7](#_Toc135226018)

[2.2.8 Envoi de fichiers 8](#_Toc135226019)

[2.3 Situation de départ 8](#_Toc135226020)

[2.4 Situation voulue 8](#_Toc135226021)

[2.5 Maquettes des applications 9](#_Toc135226022)

[2.5.1 Élève 9](#_Toc135226023)

[2.5.2 Professeur 10](#_Toc135226024)

[2.6 Fonctionnalités prévues 11](#_Toc135226025)

[2.6.1 Élève 11](#_Toc135226026)

[2.6.2 Professeur 12](#_Toc135226027)

[2.6.3 Communication 12](#_Toc135226028)

[2.7 Schéma de fonctionnement 13](#_Toc135226029)

[2.7.1 Normal 13](#_Toc135226030)

[2.7.2 Stream 14](#_Toc135226031)

[2.7.3 Contrôle 14](#_Toc135226032)

[2.7.4 Sélénium 15](#_Toc135226033)

[2.8 Schéma de classe 15](#_Toc135226034)

[3 Gestion du projet 20](#_Toc135226035)

[4 Mise en place 21](#_Toc135226036)

[4.1 Matériel 21](#_Toc135226037)

[4.2 Logiciel 21](#_Toc135226038)

[5 Application Élève 21](#_Toc135226039)

[5.1 Interface 21](#_Toc135226040)

[5.2 Démarrage 22](#_Toc135226041)

[5.3 Fonctionnement 22](#_Toc135226042)

[*5.3.1* *Spécialisation :* 22](#_Toc135226043)

[*5.3.2* *Trouver les processus lancés par l’utilisateur* 22](#_Toc135226044)

[*5.3.3* *Arrêter les processus lancés par l’utilisateur* 22](#_Toc135226045)

[*5.3.4* *Capture d’écran des deux moniteurs* 22](#_Toc135226046)

[*5.3.5* *Récupération des urls ouverts* 22](#_Toc135226047)

[5.3.6 Sauvegarde de l’IP de professeur 23](#_Toc135226048)

[5.3.7 Navigateur Intégré 23](#_Toc135226049)

[5.3.8 Réception de messages 23](#_Toc135226050)

[5.3.9 Réception du Stream 23](#_Toc135226051)

[5.3.10 Blocage de la souris 24](#_Toc135226052)

[5.3.11 Blocage du clavier 25](#_Toc135226053)

[5.4 Blocage des applications 25](#_Toc135226054)

[5.5 Intégration de NavLogeur 26](#_Toc135226055)

[5.6 Fermeture 26](#_Toc135226056)

[6 Application Professeur 27](#_Toc135226057)

[6.1 Interface 27](#_Toc135226058)

[6.1.1 Principale 27](#_Toc135226059)

[6.1.2 Individuel 28](#_Toc135226060)

[6.2 Démarrage 28](#_Toc135226061)

[6.3 Fonctionnement 29](#_Toc135226062)

[*6.3.1* *Spécialisation :* 29](#_Toc135226063)

[*6.3.2* *Envoi de requêtes* 29](#_Toc135226064)

[*6.3.3* *Affichage des données client* 29](#_Toc135226065)

[6.3.4 TreeView Détails 29](#_Toc135226066)

[6.3.5 Listes de configurations 29](#_Toc135226067)

[6.3.6 TreeView Miniatures 30](#_Toc135226068)

[6.3.7 Affichage des Miniatures 30](#_Toc135226069)

[6.3.8 Affichage individuel 31](#_Toc135226070)

[6.3.9 Options de Streaming 31](#_Toc135226071)

[6.3.10 Filtrage 31](#_Toc135226072)

[6.3.11 Configuration 31](#_Toc135226073)

[6.3.12 Prise de control à distance 32](#_Toc135226074)

[6.4 Fermeture 32](#_Toc135226075)

[7 Liaisons entre les deux applications 32](#_Toc135226076)

[7.1 Spécialisation : 32](#_Toc135226077)

[*7.2* *Connexion* 32](#_Toc135226078)

[*7.3* *Socket* 32](#_Toc135226079)

[*7.4* *Données* 33](#_Toc135226080)

[*7.5* *Images* 33](#_Toc135226081)

[*7.6* *Flux d’image pour les élèves* 33](#_Toc135226082)

[7.7 Réception des demandes 33](#_Toc135226083)

[7.8 Flux d’image pour le professeur 34](#_Toc135226084)

[8 Tests 34](#_Toc135226085)

[8.1 Méthode de test 34](#_Toc135226086)

[8.2 Protocole de Test 35](#_Toc135226087)

[9 Problèmes et solutions 36](#_Toc135226088)

[10 Références 37](#_Toc135226089)

[10.1 Webographie 37](#_Toc135226090)

[10.2 Glossaire 38](#_Toc135226091)

[11 Bilan 38](#_Toc135226092)

[11.1 Améliorations possibles 38](#_Toc135226093)

[11.2 Objectifs atteints 39](#_Toc135226094)

[11.2.1 Élève 39](#_Toc135226095)

[11.2.2 Professeur 39](#_Toc135226096)

[11.3 Conclusion 39](#_Toc135226097)

[12 Annexe WebSummary 40](#_Toc135226098)

[13 Annexe Manuel utilisateur 41](#_Toc135226099)

[13.1 Professeur 41](#_Toc135226100)

[13.2 Élève 41](#_Toc135226101)

[14 Annexe Journal de travail 42](#_Toc135226102)

[15 Annexe Planification 53](#_Toc135226103)

# Présentation du projet

## Prérequis

### Matériel et logiciels

3 ordinateurs avec Windows 10.

Les trois ordinateurs sont connectés sur le même réseau local.

Visual Studio 2022 sur l’un des ordinateurs.

### Connaissances requises

Connaissances en développement orienté objet en C#

Connaissances sur l’envoi de données sur le réseau avec les Socket.

Connaissances sur le fonctionnement des applications.

Connaissances en multithreading.

## Cahier des charges

### Connexion

Il y a une application professeur et une application élève.

Professeur : affiche son adresse IP

Elève : permet la saisie de l’IP du professeur et la stocke de manière pérenne. Reset du stockage possible

Elève : se souvient des IP précédentes et les propose en fonction de la demi-journée courante

Professeur : affiche la liste des élèves connectés et met en évidence ceux possiblement déconnectés

Elève : répond aux sollicitations du professeur (screenshot, liste ou kill de processus, Stream, contrôle...)

Elève/ professeur : annonce sa déconnexion en cas de fermeture, stoppant les processus de dialogue

Professeur : peut mettre fin à une connexion (ex. si l’élève a laissé tourner et n’est pas dans la leçon)

### Interface graphique (IHM) principale du professeur

L’IHM doit être organisée pour montrer le maximum d’information utile sans changer de fenêtre

Par ex, un TreeView avec les postes connectés à gauche et une case à cocher pour les observer

A droite, une mosaïque de captures des écrans, redimensionnée pour voir tous ceux cochés

Les miniatures sont accompagnées du nom du poste et de l’âge de la capture

Un slider de zoom permet d’agrandir les miniatures devenant visible par à une barre de défilement

On peut déployer ou replier les branches du TreeView, une par une ou toutes d’un coup

Le TreeView déployé montre les applications lancées et leur titre (après filtrage, voir filtre plus loin)

Pour les navigateurs, il est possible d’afficher l’historique du jour : heure, titre et si possible url

Il est possible de tuer le processus d’une application ou de l’ajouter au filtre des apps à ignorer

Il est possible d’ouvrir une fenêtre d’observation d’un poste en particulier. Elle est refermable.

Le dialogue entre le professeur et les élève s peut être consulté (historique et temps réel)

Le professeur et les élèves peuvent s’envoyer des courts messages texte qui surgissent chez l’élève

Un bouton démarre la diffusion multicast sur les postes sélectionnés, un bouton la stoppe

Le multicast fonctionne sur n’importe quel PC. Pour le cast de 1 écran, on peut choisir lequel caster

On peut faire un multicast des deux écrans de l’enseignant, il s’affiche sur les deux écrans élève s

Seul l’écran où se trouve la souris est réellement casté. L’autre affiche celui au départ de la souris

Si un élève n’envoie plus d’informations, Une alerte est très visible dans le TreeView (rouge, durée)

### Filtrage de l’affichage des processus

Par défaut, le programme examine les nouveaux processus et ignore ceux présents à son lancement

Le professeur diffuse trois listes d’applications : surveillées (voir), ignorées (cacher) et interdites (alerte)

Afin de compléter la liste, il est possible de désactiver le filtre.

Le professeur diffuse une liste des expressions régulières dans le titre qui déclenchent une alerte

La liste des expressions régulières de titre est un simple fichier texte éditable sur le poste du professeur

Le professeur doit pouvoir envoyer une liste de sites autorisés (avec jokers) sur le même principe

L’élève fait appliquer le filtre des sites (quitte à n’autoriser qu’un seul navigateur compatible)

### Capture d’écran

A intervalle régulier, la capture de tous les écrans des élève s est envoyée et possiblement affichée

Le professeur peut sauver l’image. Filename et texte dans l’image = yymmdd-hhmm-identité-Poste.png

### Diffusion multicast depuis le poste professeur

Le professeur peut diffuser de manière fluide sur les élèves s sélectionnés, avec le curseur souris.

C’est un streaming multicast (il n’y a qu’un flux pour tous les élève s)

Le professeur peut choisir la priorité du Stream : fenêtre, FullScreen, TopMost, TopMost et blocage de l’élève.

Il peut choisir le focus autorisé pendant le Stream : tout, rien, OneNote, DevEnv, VSCode, Word, etc...

### Streaming monoposte depuis l’élève, contrôle à distance

Depuis la fenêtre d’observation, il serait commode d’afficher un flux fluide de capture de l’élève.

Idéalement, le professeur aurait le contrôle à distance sur l’élève pour apporter de l’aide (souris/clavier)

Pendant la prise de contrôle, l’enseignant choisit si élève peut (ou pas) utiliser sa souris et son clavier

L’affichage serait réactif (quand l’enseignant bouge la souris : pas de saccade, le curseur est visible)

Pendant le contrôle à distance, l’enseignant peut basculer entre l’écran 1, 2 ou les deux écrans

### Portabilité de l’élève

L’élève doit être livré sous forme de 3 projets. (1) une bibliothèque de classes contenant la logique (2) un démonstrateur utilisable : l’application élève (3) un projet de tests unitaire (simule l’envoi de cmd professeur)

L’élève doit détecter la configuration des adaptateurs Ethernet et proposer un script à copier-coller en PowerShell administrateur capable d’activer la bonne carte (celle sur le sous-réseau multicast)

La documentation GitLab doit indiquer comment intégrer la bibliothèque de classes dans un projet.L’intégration dans le projet Navlogeur sera faite par sa développeuse uniquement grâce à la doc.

Les mises à jour seront poussées sur GitLab et la développeuse suivra la doc pour les appliquer

L’élève est prévu pour relayer des messages professeur↔ élève gérée par une méthode dédiée

Le navigateur de filtrage pourrait être Navlogueur (fourni avec documentation par la développeuse)

### Envoi de fichiers

Idéalement, il devrait être possible au professeur d’envoyer des fichiers aux élèves sélectionnés

Idéalement, il devrait être possible aux élève s d’envoyer des fichiers au professeur

## Situation de départ

Pendant sa spécialisation, l’étudiant a appris à utiliser des sockets pour connecter des élèves à un serveur et dialoguer avec lui. Il a réussi à envoyer la liste des processus, les titres des fenêtres Web et une capture des 2 écrans de l’utilisateur. Ces informations, gérées dans un thread séparé s’affichaient dans l’interface graphique de l’application du professeur qui pouvait alors les consulter et, par exemple tuer un processus à distance. Un démonstrateur a été réalisé pour effectuer un multicast de l’écran de l’enseignant sur les postes élèves. L’image est découpée en paquets non compressés, envoyés en UPD en multicast sur le sous-réseau. Les élèves reconstituent l’image quand les paquets ont été reçus. Le multicast fonctionne moyennant une configuration correcte des cartes réseau.

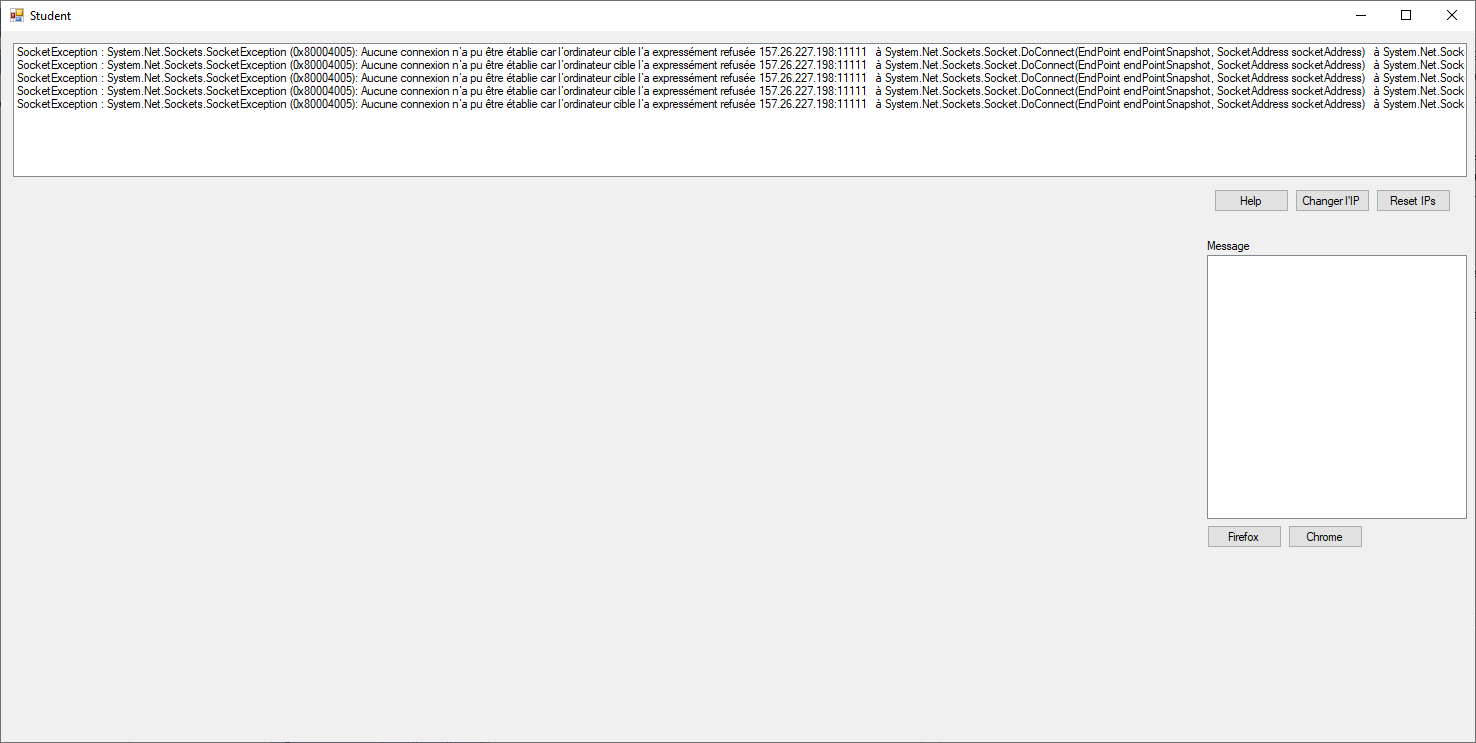
## Situation voulue

Le projet consiste à poursuivre ce développement de façon à disposer d’une alternative à Impero pour la surveillance des postes des élèves, la diffusion de démonstrations, la prise de contrôle pour effectuer une assistance à distance, le tout avec un logiciel léger et offrant des options non disponibles dans Impero, par exemple le Stream double écran et le filtrage pendant le Stream

## Maquettes des applications

### Élève

Principal



IP du professeur

Une image contenant texte, capture d’écran, Police, nombre

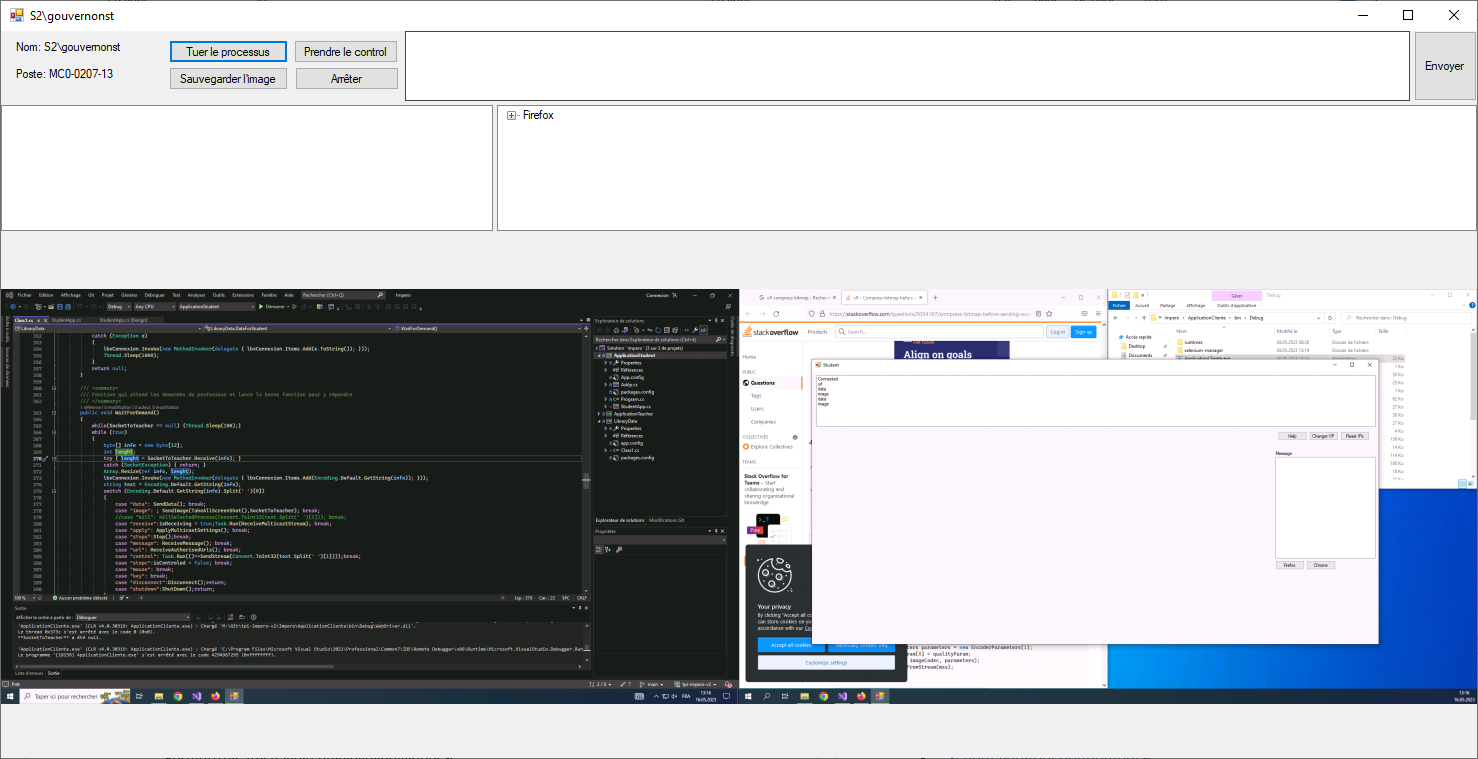
Description générée automatiquement

### Professeur

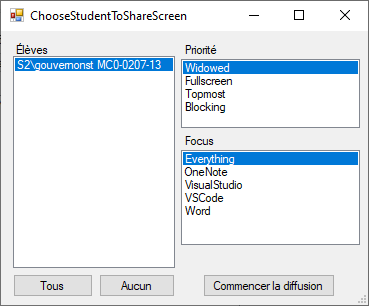
Principale



Affichage individuel



Choix pour le Stream



Choix de l’IP

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

## Fonctionnalités prévues

### Élève

Prendre une capture d’écran de tous les écrans

Récupération des processus lancé par l’utilisateur.

Récupération de toutes les url ouvertes dans tous les navigateurs

Minimiser en TrayIcon pour cacher l’application.

Afficher l’écran de l’application professeur.

Afficher le Stream reçu.

Bloquer le clavier et la souris pendant un Stream.

Autoriser que certaines applications pendant un Stream.

Limiter les sites visitables à la liste reçue.

### Professeur

Arrête des processus indésirables des élèves

Minimiser en TrayIcon pour cacher l’application.

Afficher les urls reçus dans un historique pour chaque navigateur.

Afficher les processus reçus.

Permettre la sélection des images que l’on veut afficher.

Afficher les captures d’écrans reçues de manière dynamique.

Afficher toutes les données dans un TreeView.

Permettre l’affichage d’un élève individuel.

Permet d’activer ou non un filtre couleur pour les processus et les urls.

Permet la capture d’un ou plusieurs écrans et leur Streaming en multicast.

Envoyer la liste des sites visitables.

Affichage modulable pour permettre la visibilité des informations.

### Communication

Envoi des données dans des threads séparé.

Transférer les données à haut débit.

Envoi des données élèves au professeur.

Envoi des captures d’écran au professeur.

Envoi de la liste des sites autorisés à tous les élèves.

Partage d’écran du professeur en multicast.

## Schéma de fonctionnement

### Normal

Voici un fonctionnement normal de l’application sans actions spéciale du professeur après le démarrage. La boucle de demande des données ne s’arrête qu’a l’arête de l’application.



### Stream

Voici ce qui se passe quand le professeur démarre un Stream.



### Contrôle

Voici ce qu’il ce passe quand le professeur prend le contrôle d’un élève



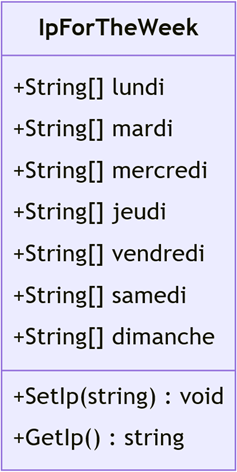
### Sélénium

Voici ce qui se passe quand un élève utilise Sélénium.

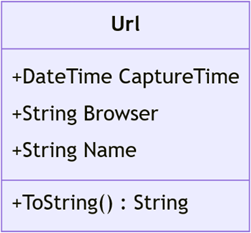


## Schéma de classe

IPForTheWeek : Classe qui stocke toutes les adresse IP des professeurs sur une semaine.



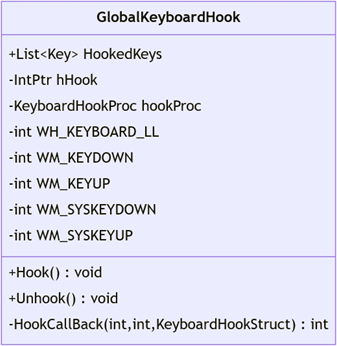
Url : Classe qui représente une url avec l’heure où il a été visité et le navigateur utilisé.



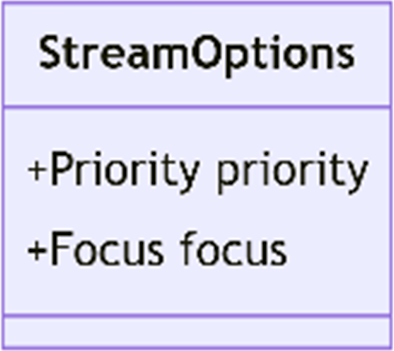
HistoriqueUrl : Classe qui représente un historique d’url pour chaque navigateur.



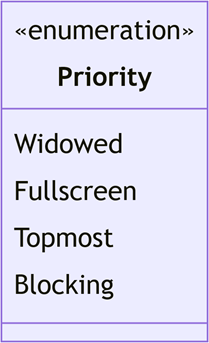
GlobalKeyboardHook : Classe permettant de récupérer l’événement d’une touche de clavier avant que les applications ne les recoivent.



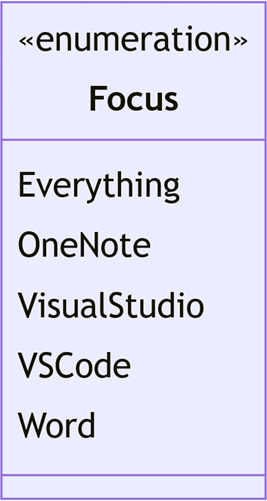
StreamOptions : Classe qui représente les paramètres d’un Stream.



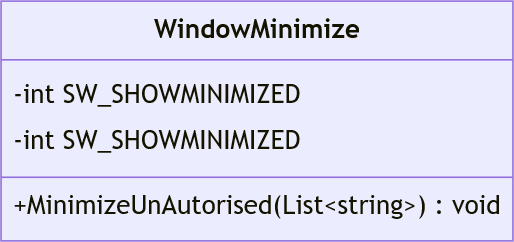
Priority : Enumération des différentes priorités possibles pendant un Stream.



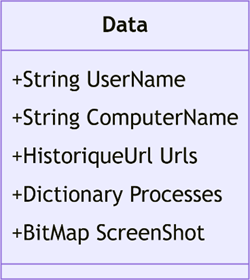
Focus : Enumération des différents focus possibles pendant un Stream.



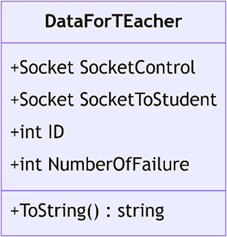
WindowMinimize : Classe permettant de minimiser une autre fenêtre.



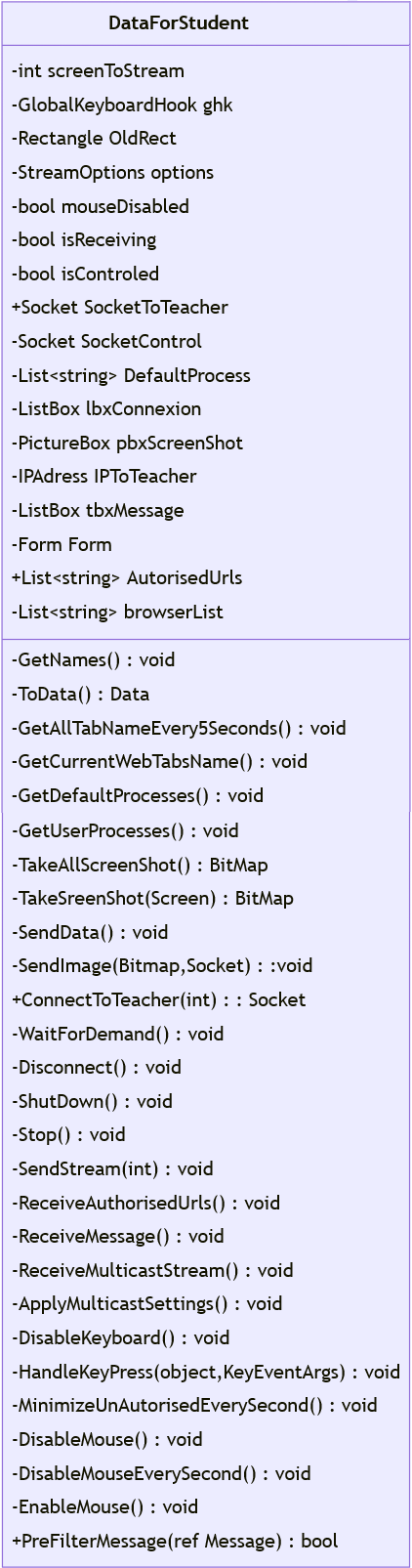
Data : Classe Utilisée pour les données transmises entre les élèves et les professeurs.



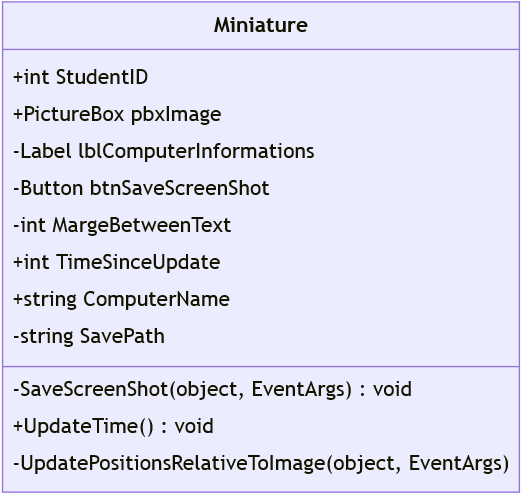
DataForTeacher : Classe qui représente un élève dans l’application professeur.



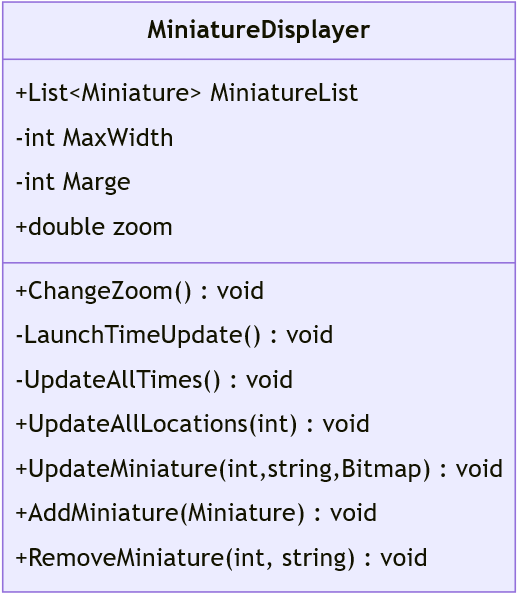
DataForStudent : Classe qui gère toute la logique de l’application élève.



Miniature : Classe qui représente une miniature avec un texte dessous.



MiniatureDisplayer : Classe qui positionne ses miniatures dans un panel.



# Gestion du projet

Pour la gestion de projet j’ai utilisé un développement agile, il s’agit d’une méthode de travail faite pour privilégier des rendus réguliers au client et des retours rapides sur ce que l’on a fait. J’ai donc fait plusieurs rendus pour les nouvelles fonctionnalités que j’avais développé. Chaque rendu est composé de plusieurs tâches qui ont un thème commun. Cela m’a permis de discuter des problèmes et des améliorations avec mon supérieur très rapidement pour ne pas perdre de temps.

# Mise en place

## Matériel

L’installation matériel est relativement simple il suffit que deux ordinateurs Windows soient sur le même réseau local. Je ne décrirais pas l’installation de Windows ici. Mais pour la liaison au même réseau local il suffit de connecter les deux ordinateurs au même switch.

## Logiciel

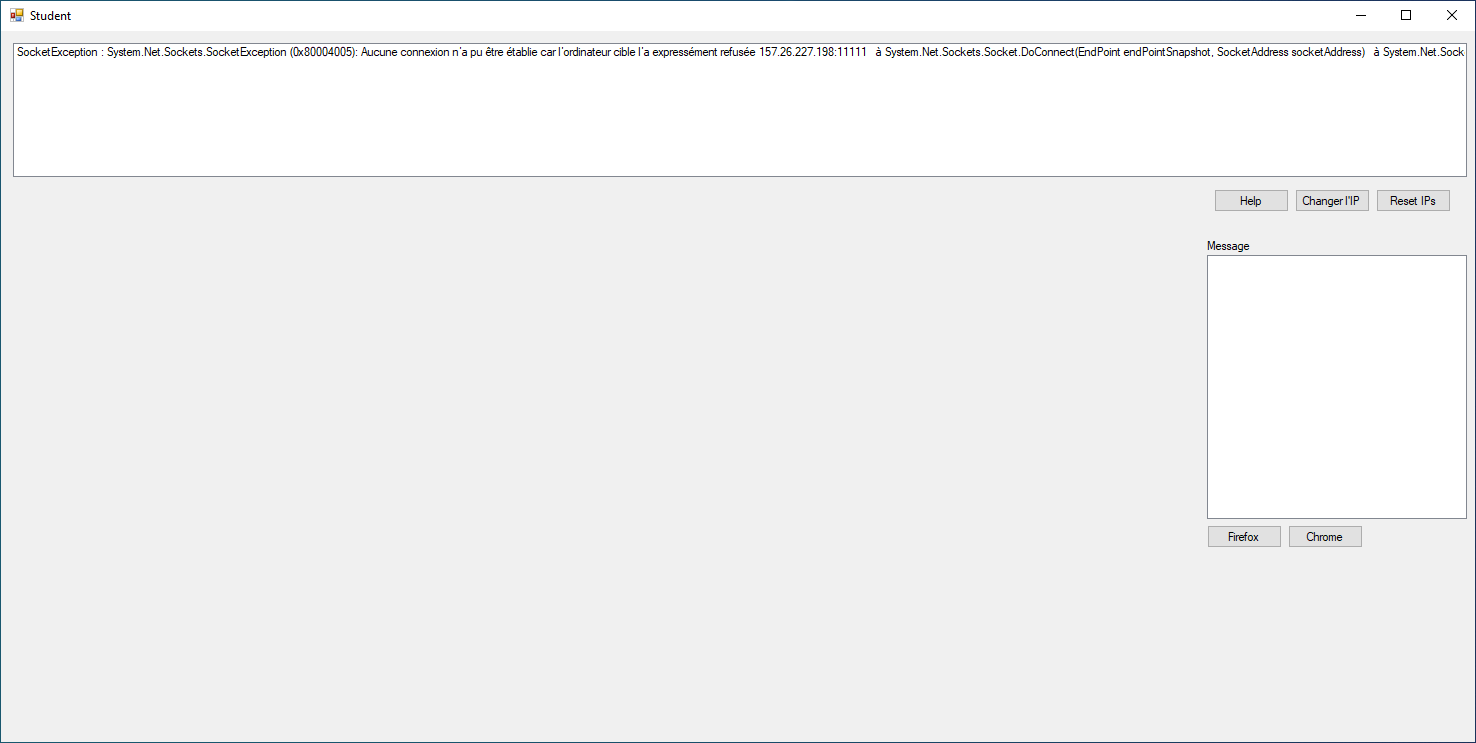
Installation de Visual Studio 2022. Mais elle nécessite tout de même des droits administrateurs. Mais si c’est le cas elle est simple, il vous suffit d’aller sur leur site <https://visualstudio.microsoft.com/fr/> puis de télécharger la version communautaire pour Windows. Á l’installation le seul module qui soit nécessaire est « Développement .NET Desktop ».

Il vous suffit ensuite de cloner le projet <https://git.s2.rpn.ch/stany24/tpi-impero-v2> ce qui nécessite de vous connecter avec vous identifiants de l’école.

Dans le cas ou des erreurs apparaissent dès l’ouverture dans le code il faut que vous désinstalliez puis réinstalliez le paquet « System.Text.Json » il ne semble pas toujours fonctionner correctement après un clonage sur git.

# Application Élève

## Interface



lbxConnection : Cette ListBox permet à l’élève de voir les communications entre lui et le professeur en directe.

PbxScreenShot : Permet d’afficher le Stream reçu par le professeur. La PictureBox est cachée tant qu’il n’y a pas de Stream. Ce qui permet de ne pas surcharger l’interface.

btnHelp : Permet de recevoir de l’aide si le streaming ne fonctionne pas.

btnChangeIP : Permet de changer l’adresse actuel du professeur.

btnResetAllIP : Permet de réinitialiser toutes les adresses IP sauvegardée.

btnFirefox : Permet de lancer une instance de Firefox contrôlée par Sélénium.

btnChrome : Permet de lancer une instance de Chrome contrôlée par Sélénium.

lbxMessage : Cette ListBox permet d’afficher les messages reçus de professeur.

## Démarrage

L’application essaye de charger l’IP du professeur depuis le fichier de configuration. Dans le cas où l’application n’arrive pas à charger l’IP une interface permet à l’élève de rentrer l’IP de son professeur.

Puis un élève est créé avec l’IP obtenue en dessus.

Puis deux tâches sont lancées en parallèles :

La connexion de l’élève au professeur.

La vérification des navigateur lancés avec sélénium.

Après la connexion au professeur une nouvelle tâche est lancée, elle attend les demandes du professeur.

## Fonctionnement

### *Spécialisation :*

### *Trouver les processus lancés par l’utilisateur*

*Pour savoir quels processus ont été lancés par l’utilisateur, il suffit de comparer les processus actuels avec les processus par défaut trouvé au point précédent.*

### *Arrêter les processus lancés par l’utilisateur*

*Après avoir récupéré la liste des processus lancé par l’utilisateur il faut pouvoir arrêter le processus indésirable. Pour cela nous envoyons un dictionnaire avec le nom et l’id du processus à l’application maître. Elle peut ensuite quand elle le désire nous renvoyer l’id pour dire à l’application cliente de stopper ce processus.*

### *Capture d’écran des deux moniteurs*

*Prendre une capture d’écran de plusieurs moniteurs n’est pas facile car on ne peut pas capturer plusieurs écrans en même temps. Il faut donc faire plusieurs captures d’écran que l’on recompose ensuite en une seule image. Pour l’instant les captures d’écran ne sont pas comparées, en cas d’écran dupliqué ils seront les deux présents.*

### *Récupération des urls ouverts*

*Après de multiples essais la récupération de tous les onglets ouverts semble peut-être possible avec AutomationElement, mais cela serait long à apprendre et à faire pour tous les navigateurs. J’avais encore deux options : récupérer le nom de l’onglet actuel ou son url. Mais le code trouvé pour récupérer l’url était différent pour chaque navigateur et semblait ne fonctionner que sur certaines versions des navigateurs. N’importe quelle mise à jour pouvait donc demander une adaptation du code. Alors que La récupération du nom des onglets était indépendante de la version du navigateur et était identique pour tous les navigateurs. C’est pour cela que j’ai finalement choisi de ne récupérer que le nom de l’onglet ouvert actuellement.*

### Sauvegarde de l’IP de professeur

Le but est simple ce souvenir de l’IP du professeur par demi-journée ce qui permet aux élèves de n’entrez l’adresse IP de leur professeur qu’au début du module.

### Navigateur Intégré

L’application élève permet de lancer des navigateurs intégrés avec Sélénium. Ils ont des avantages car nous pouvons les contrôler avec l’application élève. Elle peut donc récupérer l’url actuel ainsi que d’interdire les url non autorisées en revenant en arrière.

|  |  |
| --- | --- |
| 1 | public void VerifyUrlOfWebDriver**(**WebDriver navigateur**,**string navigateurName**)** |
| 2 | **{** |
| 3 | **try** |
| 4 | **{** |
| 5 | **if** **(**navigateur **==** **null){return;}** |
| 6 | Client**.**Urls**.**AddUrl**(new** Url**(**DateTime**.**Now**,** navigateurName**,** navigateur**.**Url**));** |
| 7 | bool navigateback **=** **true;** |
| 8 | **foreach** **(**string url **in** Client**.**AutorisedUrls**)** |
| 9 | **{** |
| 10 | **if** **(**navigateur**.**Url**.**Contains**(**url**))** **{** navigateback **=** **false;** **}** |
| 11 | **}** |
| 12 | **if** **(**navigateback**)** |
| 13 | **{** |
| 14 | string url **=** navigateur**.**Url**;** |
| 15 | navigateur**.**Navigate**().**Back**();** |
| 16 | **}** |
| 17 | **}** |
| 18 | **catch** **{** navigateur**.**Dispose**();** **}** |
| 19 | **}** |

### Réception de messages

L’application élève est capable de recevoir des messages venant du professeur. Ils sont affichés dans une listbox pour avoir un semblant d’historique des messages.

### Réception du Stream

L’application élève reçoit juste avant le flux d’image du Stream sa priorité et son focus autorisé.

Pour la priorité il y a 4 possibilités :

Windowed : l’application élève reste fenêtrée et n’a pas de caractéristique spéciale elle peut être minimisée ou cachée.

FullScreen : l’application élève est forcément en pleine écran mais elle ne peut pas être minimisée. D’autres applications peuvent tout de même se mettre devant.

TopMost : l’application élève est forcément en pleine écran ainsi que devant les autres applications elle ne peut pas non plus être minimisée.

Blocking : l’application est dans le même état qu’en TopMost mais l’élève n’a plus aucun contrôle sur le pc, ni avec la souris ni avec le clavier.

|  |  |
| --- | --- |
| 1 | **private** void ApplyMulticastSettings**()** |
| 2 | **{** |
| 3 | byte**[]** message **=** **new** byte**[**128**];** |
| 4 | int size **=** SocketToTeacher**.**Receive**(**message**);** |
| 5 | Array**.**Resize**(ref** message**,** size**);** |
| 6 | options **=** JsonSerializer**.**Deserialize**<**StreamOptions**> (**Encoding**.**Default**.**GetString**(**message**));** |
| 7 | pbxScreenShot**.**Invoke**(new** MethodInvoker**(delegate** **{** pbxScreenShot**.**Visible **=** **true;** **}));** |
| 8 | pbxScreenShot**.**Invoke**(new** MethodInvoker**(delegate** **{** pbxScreenShot**.**Dock **=** DockStyle**.**Fill**;** **}));** |
| 9 | form**.**Invoke**(new** MethodInvoker**(delegate** |
| 10 | **{** |
| 11 | form**.**Show**();** |
| 12 | form**.**Controls**.**SetChildIndex**(**pbxScreenShot**,** 0**);** |
| 13 | **switch** **(**options**.**priority**)** |
| 14 | **{** |
| 15 | **case** Priority**.**Fullscreen**:** |
| 16 | form**.**FormBorderStyle **=** FormBorderStyle**.**None**;** |
| 17 | form**.**WindowState **=** FormWindowState**.**Maximized**;** |
| 18 | **break;** |
| 19 | **case** Priority**.**Blocking**:** |
| 20 | form**.**FormBorderStyle **=** FormBorderStyle**.**None**;** |
| 21 | form**.**WindowState **=** FormWindowState**.**Maximized**;** |
| 22 | form**.**TopMost **=** **true;** |
| 23 | mouseDisabled **=** **true;** |
| 24 | Task**.**Run**(**DisableMouseEverySecond**);** |
| 25 | DisableKeyboard**();** |
| 26 | **break;** |
| 27 | **case** Priority**.**Topmost**:** |
| 28 | form**.**TopMost **=** **true;** |
| 29 | form**.**FormBorderStyle **=** FormBorderStyle**.**None**;** |
| 30 | form**.**WindowState **=** FormWindowState**.**Maximized**;** |
| 31 | **break;** |
| 32 | **case** Priority**.**Widowed**:** |
| 33 | form**.**Controls**.**SetChildIndex**(**pbxScreenShot**,** 0**);** |
| 34 | **break;** |
| 35 | **}** |
| 36 | **}));** |
| 37 | **}** |

### Blocage de la souris

Le blocage de la souris empêche l’élève d’utiliser sa souris lorsque le professeur fait une diffusion ou qu’il à pris le contrôle du pc de l’élève.

La première fonction désactive la souris

|  |  |
| --- | --- |
| 1 | private void DisableMouse**()** |
| 2 | **{** |
| 3 | Cursor**.**Clip **=** **new** Rectangle**(**0**,** 60**,** 1**,** 1**);** |
| 4 | Cursor**.**Hide**();** |
| 5 | Application**.**AddMessageFilter**(this);** |
| 6 | **foreach** **(**var process **in** Process**.**GetProcessesByName**(**"Taskmgr"**))** |
| 7 | **{** |
| 8 | process**.**Kill**();** |
| 9 | **}** |
| 10 | **}** |

Le deuxième appel la première toutes les secondes car les raccourcis clavier WINDOWS+L et CTRL+ALT+DEL redonne le contrôle de la souris

|  |  |
| --- | --- |
| 1 | private void DisableMouseEverySecond**()** |
| 2 | **{** |
| 3 | OldRect **=** Cursor**.**Clip**;** |
| 4 | **while** **(**mouseDisabled**)** |
| 5 | **{** |
| 6 | DisableMouse**();** |
| 7 | Thread**.**Sleep**(**1000**);** |
| 8 | **}** |
| 9 | EnableMouse**();** |
| 10 | **}** |

La dernière fonction est appelée pour redonner le contrôle de la souris à l’élève.

|  |  |
| --- | --- |
| 1 | private void EnableMouse**()** |
| 2 | **{** |
| 3 | Cursor**.**Clip **=** OldRect**;** |
| 4 | Cursor**.**Show**();** |
| 5 | Application**.**RemoveMessageFilter**(this);** |
| 6 | **}** |

### Blocage du clavier

Le blocage du clavier à une fonction similaire au blocage de la souris : on ne veut pas que l’élève puisse les utiliser pendant un Stream ou une prise de control. Pour cela j’utilise la classe GlobalKeyboardHook qui me permet de choisir quelle touche je veux surveiller ainsi que les fonctions à appeler lorsqu’elles sont pressées. Pour que l’élève regagne le contrôle il suffit de désactiver la surveillance de la classe GlobalKeyboardHook avec la fonction unHook().

|  |  |
| --- | --- |
| 1 | private void DisableKeyboard**()** |
| 2 | **{** |
| 3 | **foreach** **(**Keys key **in** Enum**.**GetValues**(typeof(**Keys**)))** |
| 4 | **{** |
| 5 | gkh**.**HookedKeys**.**Add**(**key**);** |
| 6 | **}** |
| 7 | gkh**.**KeyDown **+=** **new** KeyEventHandler**(**HandleKeyPress**);** |
| 8 | gkh**.**KeyUp **+=** **new** KeyEventHandler**(**HandleKeyPress**);** |
| 9 | **}** |

## Blocage des applications

Durant un Stream le professeur peut choisir quelles applications les élèves peuvent utiliser ce qui permet de minimiser actions que les élèves peuvent faire pendant le Stream.

|  |  |
| --- | --- |
| 1 | **public** static void MinimizeUnAuthorised**(**List**<**string**>** autorisedProcesses**)** |
| 2 | **{** |
| 3 | Process thisProcess **=** Process**.**GetCurrentProcess**();** |
| 4 | List**<**Process**>** processes **=** Process**.**GetProcesses**().**ToList**();** |
| 5 | **foreach** **(**Process process **in** processes**)** |
| 6 | **{** |
| 7 | **if(**process**.**ProcessName **!=** autorisedProcesses**[**0**]** **&&** process**.**ProcessName **!=** thisProcess**.**ProcessName**)** |
| 8 | **{** |
| 9 | **try** **{** ShowWindow**(**process**.**MainWindowHandle**,** SW\_SHOWMINIMIZED**);** **}** **catch** **{** **}** |
| 10 | **}** |
| 11 | **}** |
| 12 | **}** |

## Intégration de NavLogeur

Pour proposer un navigateur intégré à l’application élève je devais initialement intégrer le projet de Marie-Ange : NavLogeur. Ce qui aurait permis d’avoir un navigateur intégré et qu’il connecte automatiquement les élèves sur plusieurs sites que l’école utilise.

Malgré tous mes efforts pour faire fonctionner nos projets ensemble un problème majeur c’est imposé : J’utilise WindowForm avec .NET 4.7.2 alors que Marie-Ange utilisait WPF avec .NET 5.0. Intégrer sons projet dans le mien était devenu impossible.

J’ai donc gardé son projet et essayé d’ajouter ma classe cliente, mais elle est elle aussi faite pour du WindowForm avec .NET 4.7.2.

J’ai donc fait ce qui semblait être la dernière solution utiliser WebView2 dans mon projet WindowForm. Cela fonctionnait mais posait des complications majeures pour l’utilisateur : pas d’onglet, pas d’extension, pas de thème, edge, etc…

J’ai fini par avoir une idée, j’avais au paravent utilisé un navigateur programmable : Sélénium, il allait régler les problèmes de WebView2 car Sélénium peut être démarré avec tous les navigateurs modernes. Les onglets fonctionnent par défaut et les thèmes et les extensions peuvent être activées.

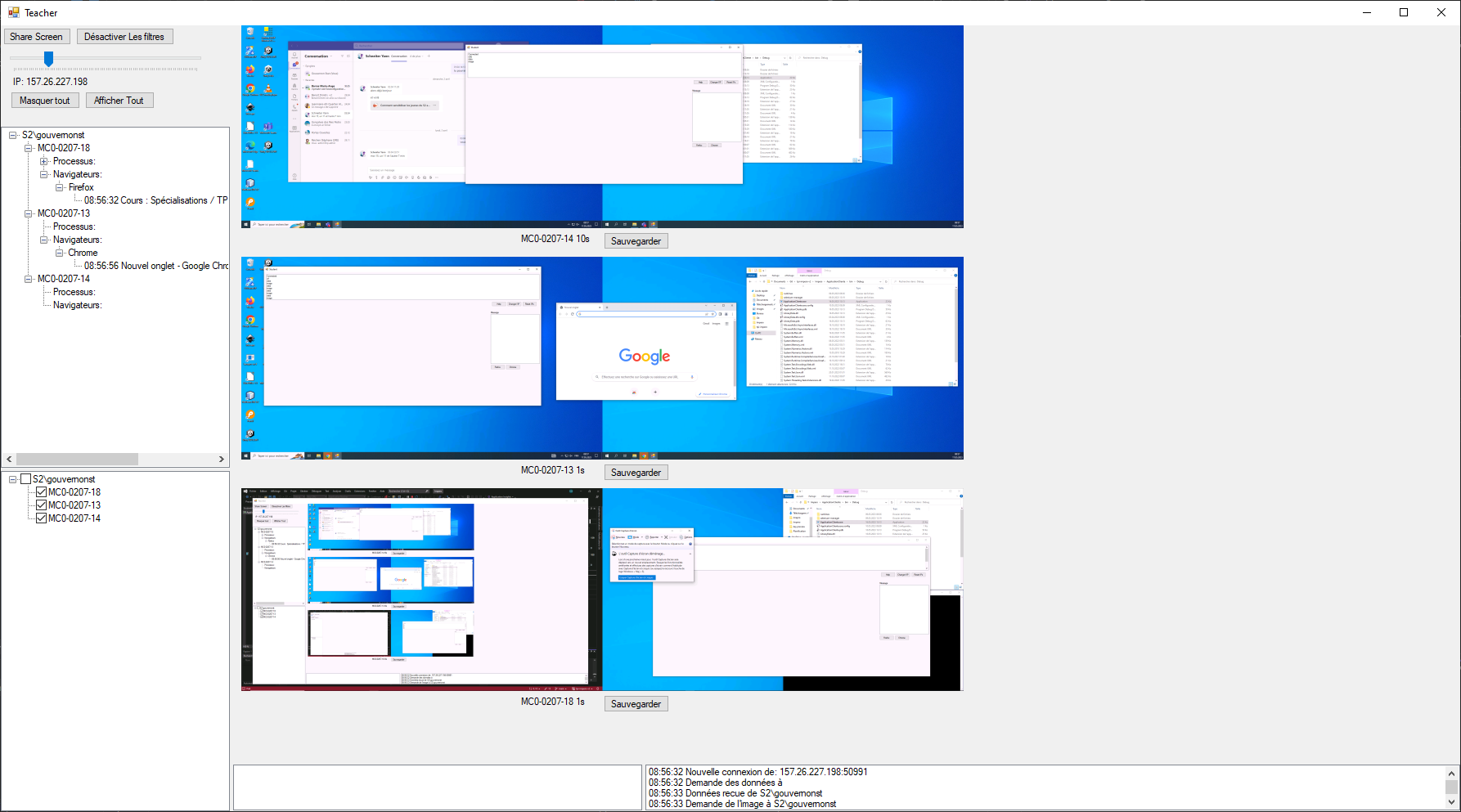
## Fermeture

L’application élève préviens l’application professeur qu’il s’arrête ce qui permet à l’application professeur de ne plus essayer de le contacter inutilement. Puis l’application arrête tous les navigateurs lancés avec sélénium.

# Application Professeur

## Interface

### Principale



L’interface principale est complexe, elle est désignée pour afficher le plus d’informations possible en même temps. Ce qui permet au professeur de voir un maximum d’élève en même temps.

btnShare : Permet de partager l’écran.

btnFilter : Permet d’activer ou non les filtres.

Slider : Permet de changer le zoom des miniatures dans panelMiniatures.

btnHideTreeView : Permet de minimiser toutes les nodes des TreeViews.

btnShowTreeView : Permet d’ouvrir toutes les nodes des TreeViews.

TreeViewDetails : Affiche tous les détails des élèves.

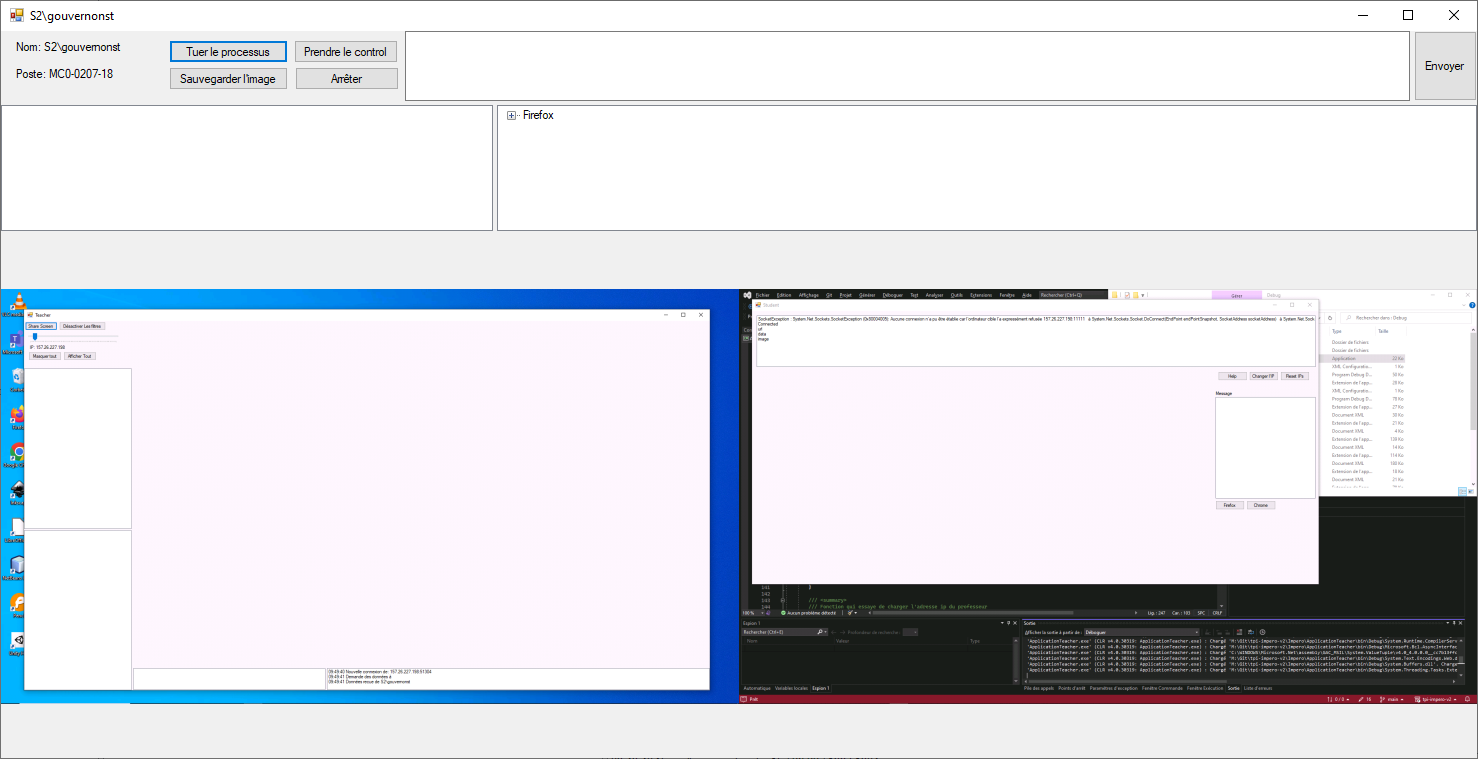
TreeViewSelection : Affiche tous les élèves et permet leur sélection.

panelMiniatures : Affiche toutes les miniatures sélectionnée dans TreeViewSelection

lbcConnexion : Affiche toutes les connexion des élèves.

lbxRequetes : Affiche toutes les requêtes faites entre les élèves et le professeur.

### Individuel



L’interface individuel permet un focus sur un seul élève ce qui permet au professeur de se concentrer sur lui. Cela lui permet aussi de nouvelles actions tel que l’envoi de message ou l’arête d’un processus.

lblUserName : Affiche le nom d’utilisateur de l’élève.

lblPoste : Affiche le nom de la machine de l’élève.

btnKillProcess : permet de tuer le processus selectionnée dans lbxProcessus.

btnSaveImage : Permet de sauvegarder l’image dans pbxScreenShot.

btnTakeControl : Permet de prendre le contrôle de l’élève.

btnStop : Permet d’arrêter l’application de l’élève.

tbxMessage : Permet d’écrire un message pour l’élève.

btnSend : Permet d’envoyer le message dans tbxMessage à l’élève.

TreeViewProcess : Affiche tous les processus de l’élève.

TreeViewUrls : Affiche tous les urls de l’élève par navigateur.

pbxScreenShot : Affiche la dernière capture d’écran de l’élève.

## Démarrage

Pour commencer l’application crée une instance de la classe MiniatureDisplayer d’après la taille du panel où elle va afficher les miniatures.

Puis l’application va chercher l’IP actuel de la machine, dans le cas ou la machine est sur plusieurs réseau une interface permet au professeur de choisir l’IP qu’il désire.

La prochaine tâche charge toutes les listes de configuration :

Processus ignorés au moment de l’affichage dans l’interface.

Processus alertés dans l’interface, signalé en rouge.

Urls alertés dans l’interface, signalé en rouge.

Urls autorisées dans les navigateurs Sélénium.

Puis il va lancer deux tâches qui vont tourner continuellement en arrière-plan :

La première va permettre aux élèves de se connecter au professeur.

La deuxième va être une demande périodique des informations aux élèves.

## Fonctionnement

### *Spécialisation :*

### *Envoi de requêtes*

*L’application maître fait des demandes à toutes les applications clientes connectées, elle demande les données puis après les avoir reçues elle demande la capture d’écran quand elle la reçoit elle passe au prochain client. Elle boucle indéfiniment tant qu’elle a des applications clientes connectée.*

### *Affichage des données client*

*L’application maître doit permettre deux vues. Une avec tous les écrans des ordinateurs clients qui permet d’avoir une vue d’ensemble sur totalité des ordinateurs clients. Et une deuxième plus précise qui affiche toutes les informations sur un seul client, et permet de d’arrêter un processus chez ce client.*

### TreeView Détails

Le TreeView en haut à gauche permet de voir les informations des élèves. Les branches commencent par le nom de l’utilisateur puis le nom du pc. Ici la première branche se sépare en deux : les navigateurs et les processus. Dans la branche processus nous pouvons voir tous les processus lancés par cet utilisateur. Dans la branche navigateur nous pouvons voir les navigateurs lancés ainsi que les urls visités avec l’heure de visite.

### Listes de configurations

L’application professeur peut être configurée avec 4 listes de configurations :

La première permet de choisir quels processus sont alerté dans les TreeViews.

La seconde permet de choisir quels processus sont ignoré dans les TreeViews.

La troisième permet de choisir quelles urls sont alerté dans les TreeViews.

La dernière permet de choisir quels sites sont autorisé dans les navigateurs sélénium des élèves.

Ces listes sont chargées au démarrage de l’application dans le dossier C:\ProgramData\Imepro si les fichiers de configuration n’existe pas l’application les créera.

|  |  |
| --- | --- |
| 1 | public void LoadFileToList**(**string pathToFile**,**List**<**string**>** list**,** ListBox lbxOutput**)** |
| 2 | **{** |
| 3 | **try** |
| 4 | **{** |
| 5 | **if** **(!**File**.**Exists**(**pathToFile**))** **{** File**.**Create**(**pathToFile**);** **}** |
| 6 | **using** StreamReader fichier **=** **new(**pathToFile**);** |
| 7 | list **=** **new(**JsonSerializer**.**Deserialize**<**List**<**string**>>(**fichier**.**ReadToEnd**()));** |
| 8 | fichier**.**Close**();** |
| 9 | **}** |
| 10 | **catch** **(**Exception e**)** |
| 11 | **{** |
| 12 | lbxOutput**.**Invoke**(new** MethodInvoker**(delegate** **{** lbxOutput**.**Items**.**Add**(**"Problème au chargement de la list : " **+** e**.**ToString**());** **}));** |
| 13 | **}** |
| 14 | **}** |

A la fermeture de l’application les listes de configuration sont sauvegardées.

|  |  |
| --- | --- |
| 1 | public void SaveListToFile**(**string pathToFile**,** List**<**string**>** list**,** ListBox lbxOutput**)** |
| 2 | **{** |
| 3 | **try** |
| 4 | **{** |
| 5 | **if** **(!**File**.**Exists**(**pathToFile**)){**File**.**Create**(**pathToFile **);}** |
| 6 | **using** StreamWriter writeFichierProcesusIgnore **=** **new(**pathToFile**);** |
| 7 | writeFichierProcesusIgnore**.**WriteLine**(**JsonSerializer**.**Serialize**(**list**));** |
| 8 | writeFichierProcesusIgnore**.**Close**();** |
| 9 | **}** |
| 10 | **catch** **(**Exception e**)** |
| 11 | **{** |
| 12 | lbxOutput**.**Invoke**(new** MethodInvoker**(delegate** **{** lbxOutput**.**Items**.**Add**(**"Problème au chargement de la list : " **+** e**.**ToString**());** **}));** |
| 13 | **}** |
| 14 | **}** |

### TreeView Miniatures

Le TreeView en bas à gauche permet de sélectionner quelles miniatures seront affichée dans le panel en haut à droite de l’écran.

Double cliquer sur un ordinateur permet d’ouvrir un affichage individuel.

### Affichage des Miniatures

Toutes les miniatures cochées dans le TreeView en bas à gauche seront affichée dans le panel. Leur taille peut être changée grâce au slider. Les miniatures sont mises à jour dès qu’une nouvelle image est reçue de l’élève. En dessous des miniatures on peut voir le nom du poste ainsi que le temps depuis la dernière mise à jour de l’image.

|  |  |
| --- | --- |
| 1 | **public** void UpdateAllLocations**(**int maxwidth**)** |
| 2 | **{** |
| 3 | MaxWidth **=** maxwidth**;** |
| 4 | int OffsetTop **=** 0**;** |
| 5 | int OffsetRight **=** 0**;** |
| 6 | int MaxHeightInRow **=** 0**;** |
| 7 | **for** **(**int i **=** 0**;** i **<** MiniatureList**.**Count**;** i**++)** |
| 8 | **{** |
| 9 | **if** **(**OffsetRight **+** MiniatureList**[**i**].**Width **>** MaxWidth**)** |
| 10 | **{** |
| 11 | OffsetTop **+=** MaxHeightInRow**;** |
| 12 | MaxHeightInRow **=** 0**;** |
| 13 | OffsetRight **=** 0**;** |
| 14 | **}** |
| 15 | MiniatureList**[**i**].**Top **=** OffsetTop**;** |
| 16 | MiniatureList**[**i**].**Left **=** OffsetRight **+** Marge**;** |
| 17 | OffsetRight **+=** MiniatureList**[**i**].**Width **+** Marge**;** |
| 18 | **if** **(**MiniatureList**[**i**].**Height **>** MaxHeightInRow**)** **{** MaxHeightInRow **=** MiniatureList**[**i**].**Height**;** **}** |
| 19 | **}** |
| 20 | **}** |

### Affichage individuel

Un seul affichage individuel ne peut être ouvert par élève. Il permet d’avoir un fenêtre entière dédiée à un élève ce qui permet de mieux voir ce que fait cet élève ou de lui envoyer des messages.

### Options de Streaming

Au démarrage d’un Stream le professeur peut choisir quelles élèves participeront ainsi que la Priorité et le Focus.

La Priorité du Stream définit :

Si l’application doit être en pleine écran.

Si l’application doit être devant toutes les autres.

Si l’utilisateur à le contrôle.

Le Focus de Stream définit dans le cas où l’utilisateur à le contrôle quelles applications peuvent être ouvertes et utilisée pendant le Stream. Par exemple : tout, OneNote, Visual Studio, VSCode, Word, etc…

### Filtrage

Le filtrage dès le TreeView Détails peut être activé ou non grâce au bouton « Activer/Désactiver » en haut à gauche de l’interface. Cela permet de ne pas afficher les processus ignorés ainsi que de faire apparaitre en rouge les processus et les urls alertés.

### Configuration

Processus Ignorés : Ces processus ne seront pas affichés dans le TreeView Détails

Processus Alertés : Ces processus seront affichés en rouge dans le TreeView Détails

Urls Alertés : Ces urls seront affichés en rouge dans le TreeView Détails.

Urls Autorisés : Ces urls seront bloqué dès les navigateurs contrôlés par sélénium sur les ordinateurs des élèves.

Toutes les configurations sont stockées dans le des fichiers sur le disque.

### Prise de control à distance

Le professeur aurait du pouvoir prendre le contrôle d’un élève à distance mais je n’ai hélas pas eu le temps de faire cette fonctionnalité.

Pour que cela fonctionne il aurait fallu que j’établisse un Stream de l’élève affiché chez le professeur, ce que j’ai eu le temps de faire mais pas de stabiliser.

Bloquer l’élève durant la prise de control, ce qui ne devrait pas être très difficile car je l’ai déjà fait.

Récupérer les actions sur la souris et le clavier du professeur

Les transmettre à l’application de l’élève.

Les simuler sur l’ordinateur de l’élève.

## Fermeture

Á la fermeture l’application professeur préviens tous les élèves qu’il s’arrête. Puis il sauve les listes de configuration dans des fichiers :

Processus ignorés au moment de l’affichage dans l’interface.

Processus alertés dans l’interface, signalé en rouge.

Urls alertés dans l’interface, signalé en rouge.

Urls autorisées dans les navigateurs Sélénium.

# Liaisons entre les deux applications

## Spécialisation :

## *Connexion*

*La connexion entre les deux applications se fait par une requête de l’application cliente à l’application maître. Pour cela l’application cliente doit connaitre l’adresse IP de l’application maître auquel elle doit se connecter. Cette connexion est utile pour créer un socket qui permet la connexion entre les deux applications à tout moment. C’est pour cela que l’application maître écoute pour des requêtes de connexion client en continu.*

Après des améliorations l’application cliente peut maintenant mémoriser les adresses IP des professeurs d’une semaine à l’autre. Ce qui permet à l’élève de ne rentre l’adresse IP de son professeur qu’au début des cours. L’élève peut bien sur réinitialiser les adresses IP ou les changer.

## *Socket*

*Un socket permet d’établir une connexion permanente entre les applications, ils nous serviront à transmettre les données sous forme de tableau de byte entre l’application cliente et l’application maître. Au moment où l’une des applications s’arrête, elle doit prévenir l’autre pour qu’elle arrête son socket.*

## *Données*

*Les données ne peuvent pas être transférées tel quel, L’application cliente doit donc sérialiser les données cliente contenue dans ClientData avant de les transformer en un tableau de byte qui est la seule forme de donné qui peut être envoyé dans le socket. Á la réception, l’application maître devra retransformer le tableau de byte reçu en une chaine de caractères qu’il pourra ensuite désérialiser pour recréer la classe ClientData.*

L’application élève utilise désormais la classe DataForStudent qui hérite de data. Le Professeur utilise la classe DataForTeacher qui hérite elle aussi de la classe data. Et la classe data contient toutes les données qu’il faut transmettre. Je convertis donc la classe DataForStudent en Data pour l’envoyer sur le réseau puis c’est au professeur de créer une instance de DataForTeacher grâce à la class Data obtenue.

## *Images*

*L’utilisation de SendFile() pour envoyer une image semblait être une bonne idée mais finalement il était plus simple de transformer le fichier en un tableau de byte et de l’envoyer ainsi. Il suffit simplement à l’application maître de recréer une image à partir du tableau de byte.*

## *Flux d’image pour les élèves*

*Le flux d’image permet un partage vidéo de l’écran de l’application maître à toutes les applications clientes. Pour cela elle doit prendre une capture d’écran puis la séparer un plusieurs paquets pour pouvoir l’envoyer.*

*L’application cliente doit recevoir tous ces paquets puis recomposer une image avec puis l’afficher à l’utilisateur jusqu’à ce que la prochaine arrive.*

## Réception des demandes

La réception des demandes du professeur se fait dans une fonction dédiée à cela. Elle est constamment en train d’écouter les demandes du professeur. Lorsqu’elle reçoit une demande elle exécute la bonne fonction pour répondre à la demande.

|  |  |
| --- | --- |
| 1 | private void WaitForDemand**()** |
| 2 | **{** |
| 3 | **while(**SocketToTeacher **==** **null)** **{**Thread**.**Sleep**(**100**);}** |
| 4 | **while** **(true)** |
| 5 | **{** |
| 6 | byte**[]** info **=** **new** byte**[**12**];** |
| 7 | int lenght**;** |
| 8 | **try** **{** lenght **=** SocketToTeacher**.**Receive**(**info**);** **}** |
| 9 | **catch** **(**SocketException**)** **{** **return;** **}** |
| 10 | Array**.**Resize**(ref** info**,** lenght**);** |
| 11 | lbxConnexion**.**Invoke**(new** MethodInvoker**(delegate** **{** lbxConnexion**.**Items**.**Add**(**Encoding**.**Default**.**GetString**(**info**));** **}));** |
| 12 | string text **=** Encoding**.**Default**.**GetString**(**info**);** |
| 13 | **switch** **(**Encoding**.**Default**.**GetString**(**info**).**Split**(**' '**)[**0**])** |
| 14 | **{** |
| 15 | **case** "data"**:** SendData**();** **break;** |
| 16 | **case** "image"**:** **;** SendImage**(**TakeAllScreenShot**(),**SocketToTeacher**);** **break;** |
| 17 | **case** "kill"**:** KillSelectedProcess**(**Convert**.**ToInt32**(**text**.**Split**(**' '**)[**1**]));** **break;** |
| 18 | **case** "receive"**:**isReceiving **=** **true;**Task**.**Run**(**ReceiveMulticastStream**);** **break;** |
| 19 | **case** "apply"**:** ApplyMulticastSettings**();** **break;** |
| 20 | **case** "stops"**:**Stop**();break;** |
| 21 | **case** "message"**:** ReceiveMessage**();** **break;** |
| 22 | **case** "url"**:** ReceiveAuthorisedUrls**();** **break;** |
| 23 | **case** "control"**:**screenToStream **=** Convert**.**ToInt32**(**text**.**Split**(**' '**)[**1**]);** Task**.**Run**(()=>**SendStream**());break;** |
| 24 | **case** "stopc"**:**isControled **=** **false;** **break;** |
| 25 | **case** "mouse"**:** **break;** |
| 26 | **case** "key"**:** **break;** |
| 27 | **case** "disconnect"**:**Disconnect**();return;** |
| 28 | **case** "shutdown"**:**ShutDown**();return;** |
| 29 | **}** |
| 30 | **}** |
| 31 | **}** |

## Flux d’image pour le professeur

Lorsque le professeur veut prendre le contrôle d’un ordinateur c’est à l’application de l’élève de streamer son écran pour le professeur mais cette fois ce n’est pas en multicast ce qui nous permet d’utiliser le protocole TCP, je peux donc envoyer les images en un seul paquet de donnée. Mais je ne peux pas utiliser le même port sur l’application professeur cela risquerait de faire des collisions avec les communications.

# Tests

## Méthode de test

Je voulais faire des tests de deux manières : avec des tests visuels et des tests unitaires. Les tests unitaires sont très efficaces et permettent de faire beaucoup de tests en peu de temps, mais il faut les créer et certaines vérifications sont très difficiles à faire. Leur création prend du temps que je n’ai pas eu. C’est pourquoi je n’ai finalement fais que des tests visuels.

## Protocole de Test

|  |  |  |
| --- | --- | --- |
| **Actions** | **Attente** | **Résultat** |
| Démarrage | | |
| Brancher 4 ordinateurs sur le même réseau. |  | OK |
| Démarrer les 4 ordinateurs. | 4 ordinateurs allumés | OK |
| Démarrer l’application professeur sur l’un des ordinateurs. | 1 application professeur allumée | OK |
| Démarrer l’application élève sur les 3 autres ordinateurs. | 3 applications élèves allumée | OK |
| Regarder l’adresse IP du professeur sur l’application professeur. | Voir par exemple 157.26.227.198 | OK |
| Rentrer l’adresse sur les 3 applications élèves. | Elle est demandée ou cliquez sur le bouton Changer l'IP | OK |
| Vérifier la connexion au professeur sur les applications élèves « Connected ». | Sur les applications élèves dans la listbox il y a Connected | OK |
| Communications | | |
| Lancer des navigateurs web sur les ordinateurs élèves. | Des navigateurs ouverts | OK |
| Vérifier que les informations sont correctement affichées pour tous les élèves. | Dans l'application professeur vérifiez que les urls apparaissent | OK |
| Lancez des applications sur les élèves. | Des applications ouvertes | OK |
| Vérifiez que les processus sont correctement affichés chez le professeur. | Dans l'application professeur vérifiez que les processus apparaissent | OK |
| Naviguer sur internet sur un pc élève. | Allez sur plusieurs sites différents | OK |
| Vérifiez que les urls sont correctement affiché chez le professeur. | Tous les sites visités sont affichés avec leur heure de visite | OK |
| Images | | |
| Ajouter/retirer des captures d’écran avec le TreeView en bas à gauche | Les captures sont ajoutées et retirées | OK |
| Redimensionnez la zone. (Sélectionnez les bords et déplacer les) | Les captures sont affichées pour rester dans la zone | OK |
| Modifier le zoom. (Avec le slider en haut à gauche) | Les captures sont affichées pour rester dans la zone | OK |
| Streaming | | |
| Démarrer un Stream, sélectionnez tous les élèves |  | OK |
| Vérifiez que tous les élèves reçoivent bien les images. |  | OK |
| Priorité du Stream | | |
| Démarrer un Streaming en multicast en bloquant le clavier et la souris. |  | OK |
| Vérifier que les élèves ne peuvent pas regagner le contrôle : |  | OK |
| Essayer de déplacer la souris, de cliquer. | Aucun résultat | OK |
| Essayer d’utiliser le clavier. | Aucun résultat | OK |
| Essayer WINDOWS+L | Bloque la souris mais regagne le clavier pour se reconnecter | OK |
| Essayer CTRL+ALT+DEL | Bloque la souris mais regagne le clavier pour se reconnecter | OK |
| Arrêter le Stream : |  | OK |
| Vérifier que les élèves ont bien regagné le contrôle du clavier et la souris. | Tous fonctionnent normalement | OK |
| Lancer le Stream en TopMost | Les autres fenêtres restent derrière le Stream | OK |
| Arrêter le Stream : | Tous fonctionnent normalement | OK |
| Focus de Stream | | |
| Démarrez des applications au Hazard sur les pc des élèves ainsi que Word. | Beaucoup d'application ouvertes | OK |
| Démarrer un Streaming en multicast avec en focus Word. | Toutes les application ce ferme sauf le Stream et Word | OK |
| Essayer d'ouvrir d'autres applications | Elle se referment en moins d'une seconde | OK |
| Arrêter le Stream : | Tous fonctionnent normalement | OK |

# Problèmes et solutions

**TreeView customisé**

Afficher des cases à cocher dans un seul niveau du TreeView ainsi que des boutons dans le TreeView. Cela ne semble pas possible sans écrire ma propre fonction de génération d’un TreeView. Ce qui me prendrait beaucoup trop de temps. J’ai donc opté pour un deuxième TreeView qui permet juste la sélection.

**IP au démarrage**

Au démarrage l’application professeur prenait la première adresse ipv4 quelle trouvait, j’ai donc ajouté une interface au démarrage pour choisir l’IP que l’on veut utiliser. Dans le cas où il n’y a qu’un seul IP elle est choisie automatiquement.

**Affichage des miniatures**

La miniature et le label n’étaient visibles que dans une petite zone, car la miniature ainsi que le label sont une classe qui hérite de UserControl. De ce fait la classe à elle aussi une taille ce qui est à l’extérieur de cette taille n’est pas affiché.

**Retirer les ListBox de la classe Élève**

On voudrait éviter d’avoir des éléments d’affichage dans la classe mais tous mes essais (BindingList, DataSource, Mise à jour manuelle) ont été futiles, car changer la source de donnée d’une ListBox (ou autres éléments d’affichage) depuis un thread qui ne s’occupe pas de l’interface est visiblement presque impossible car la ListBox n’accepte que les changements qui viennent du thread qui s’occupe de l’interface.

**Intégration du projet Navlogeur.**

Le premier essai a été de transformer mon application Windows Forms en projet WPF. Cela s’est avéré très difficile car les versions utilisées de .NET ne sont pas identiques (Windows Forms <=4.7.2, WPF >=5.0). Le projet WPF ne pouvait pas utiliser les classes que j’avais créé lorsque je les lui donnais en référence. La solution a été d’ajouter la classe dans le projet WPF en tant que ressource liée.

**La classe ne peux pas fonctionner.**

La classe ne peux pas fonctionner avec les deux using (System.Windows.Forms et System.Windows.Controls) qui sont tout les deux nécessaires pour le fonctionnement de Windows Forms et WPF. Cela m’a fait choisir d’arrêter d’essayer d’implémenter mon application en WPF

**WebView2 n’est pas complet**

Pour avoir un navigateur intégré à mon application j’ai décidé d’utiliser une fenêtre WebView2. C’était fonctionnel mais il manquant des commodités de navigations : onglets, barre de recherche, bouton retour, etc... Finalement j’ai décidé d’intégrer Sélénium au projet car il offrirait un navigateur complet à tous les élèves et permet aussi de récupérer les urls.

**Empêcher l’utilisation du clavier avec des Hook.**

Après avoir effectué des recherches je suis arrivé sur ce projet : <https://www.codeproject.com/Articles/19004/A-Simple-C-Global-Low-Level-Keyboard-Hook> qui avait déjà implémenté le verrouillage du clavier. Mais après l’avoir testé dans mon projet une erreur revenait : Le delegate était perdu dans le Garbage Collector. Il a fallu que je modifie le code pour créer une instance du delegate dans une variable et ne plus l’instancier à chaque utilisation ce qui permettait de ne plus perdre le delegate.

**Empêcher l’utilisation de certaines applications pendant le Stream**

La première solution aurait été de fermer toutes les applications non autorisées mais on risquait la perte de données et de temps pour l’élève c’est pourquoi j’ai décidé de minimiser les fenêtres interdites durant le Stream, mais je ne pouvais pas empêcher les élèves de les agrandir à nouveau. C’est pourquoi je les minimise toutes les secondes.

# Références

## Webographie

Cases à cocher uniquement à certains niveaux du TreeView : <http://dotnetfollower.com/wordpress/2011/05/winforms-treeview-hide-checkbox-of-treenode/>

Bouton dans le TreeView : <http://dotnetfollower.com/wordpress/2011/05/winforms-treeview-hide-checkbox-of-treenode/>

Mettre à jour les binding List dans une application multithread : <https://stackoverflow.com/questions/4823481/bindinglist-not-updating-bound-listbox>

Hook pour capturer l’événement des touches du clavier : <https://www.codeproject.com/Articles/19004/A-Simple-C-Global-Low-Level-Keyboard-Hook>

Empêcher l’utilisation de la souris pendent le Stream ou la prise de control : <https://stackoverflow.com/questions/2698673/disabling-mouse-movement-and-clicks-altogether-in-c-sharp>

Minimiser les fenêtres qui ne sont pas autorisée : <https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-showwindow>

## Glossaire

TreeView : Affichage en forme de branche qui se sépare en plusieurs.

Stream : Vidéo en directe

Garbage Collector : Moyen de gérer la mémoire dans les langages de haut niveau.

Hook : Crochet pour attraper quelque chose.

NavLogeur : Projet de navigateur C# WPF basé sur WebView2 créé par Marie-Ange Barras.

.NET : Est un Framework cross-plateforme open-source.

Framework : Infrastructure de développement.

Open-Source : License de libre distribution.

BindingList : Manière de donner une source de donnée à une listbox.

DataSource : Manière de donner une source de donnée à une listbox.

# Bilan

## Améliorations possibles

Améliorer la stabilité des programmes : Les programmes ont la plupart des fonctionnalités demandé, cela m’a forcé à développer des nouvelles fonctionnalités plutôt que de m’assurer de la stabilité des anciennes. Ils restent utilisables mais ne peuvent pas encore être considérer stables.

Améliorer la qualité des Stream : Certaines images ne sont pas reçues correctement et il manque le bas de l’image ce qui rend le Stream désagréable à regarder si cela arrive trop souvent.

Création d’une interface pour modifier les listes de configurations depuis l’application professeur. Cela empêcherait les professeurs de faire des erreurs lorsqu’ils modifient manuellement les listes.

Les navigateurs sélénium ne charge pas d’extension ni de thème ce qui peut nuire à l’utilisation de l’élève. Le but étant que les élèves utilisent ces navigateurs plutôt que les conventionnels sans extension et thème j’ai peur qu’ils ne soient peu utilisés.

Créer un projet de tests unitaire pour couvrir la plupart des fonctions des applications. Cela permettrait facilement de vérifier si des changements ont introduits des bugs ou non.

## Objectifs atteints

### Élève

Blocage du clavier et de la souris.

Blocage des applications interdites.

Navigateur intégré.

Sauvegarde de l’IP du professeur.

Bloque les sites non autorisés sur Sélénium.

Annonce sa fermeture.

### Professeur

Affichage du maximum d’information.

Affichages des captures désirées.

Affichage individuel des élèves.

Historique des urls visités.

Arrêt d’un client.

Sauvegarde des captures d’écrans.

Stockage de listes de configuration.

Annonce sa fermeture

## Conclusion

Bien que satisfait du travail accompli le manque de temps m’a forcé à développer de nouvelles fonctionnalités plutôt que de stabiliser les anciennes ce qui m’a mené à une application qui a la plupart des fonctionnalités demandé mais aurait besoin de 20 heures pour être complètement utilisable. J’ai tout de même beaucoup appris et réussit des objectifs qui me semblait très difficiles (Blocage de l’utilisateur). Malgré que l’intégration de NavLogeur ne se soit pas passé comme prévue j’ai réussi à trouver une alternative respectable avec Sélénium. Je suis donc très content du travail réalisé

# Annexe WebSummary

**Imepro Gouvernon Stan, canton Neuchâtel, CPNE-TI**

**Environnement**

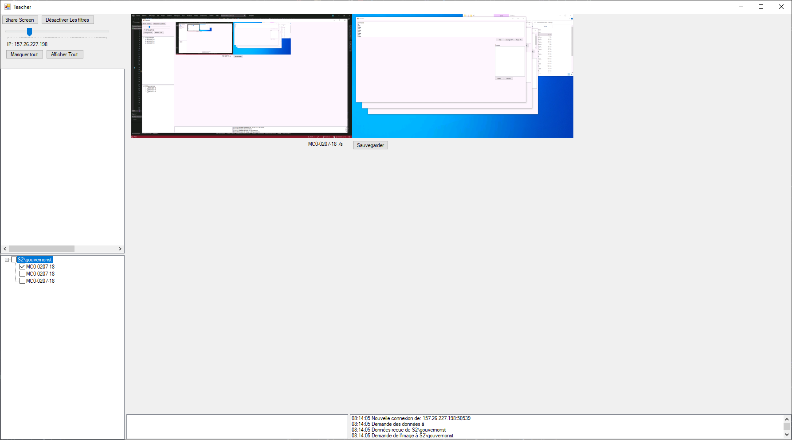
Étudiant au CPNE-TI depuis maintenant 3 ans, je me suis habitué à l’utilisation du logiciel Impero. Il permet aux enseignant de monitorer leur classe, de faire des présentations à toute la classe et d’aider un élève en prenant le control de son ordinateur.

**Situation de départ**

Mon projet à début durant la spécialisation ou j’ai créé deux applications pour remplacer Impero. Mais il leur manquait beaucoup de fonctionnalités : affichage de toutes les captures, options pour le Stream, prise de contrôle à distance etc…

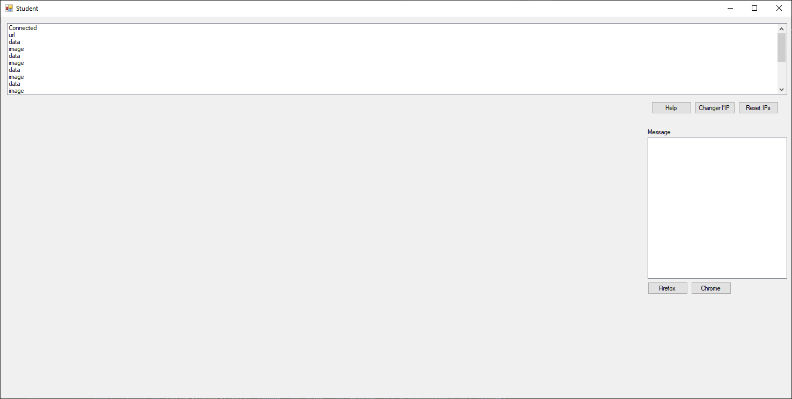
**Objectifs**

Mon objectif a été d’implémenter des fonctionnalités déjà présente dans Impero tel que le blocage des élèves pendant un Stream. J’ai aussi créé de nouvelle fonctionnalité qui rendrait l’utilisation d’Imepro plus facile qu’Impero comme le moyen d’afficher toutes les captures d’écran des élèves en même temps. Étant moi-même un élève j’avais connaissance d’un moyen de regagner le control pendant un Stream sur Impero ce que j’ai corrigé sur Imepro.

**Représentation graphique**

Vous pouvez voir à droite les applications finales, en haut l’application du professeur qui permet de voir les élèves et leurs captures d’écran. En bas l’application de l’élève qui permet principalement de recevoir le Stream.

**Voie Suivie**

J’ai fait tous mon développement en C# car le langage est enseigné dans l’école et permet de développer des interfaces graphiques facilement. De plus il à fallut que l’application élève soit tout de même attractive pour les élèves elle ne devait pas devenir uniquement un outil de surveillance pour les professeurs.

**Résultat**

Le résultat est satisfaisant mais manque de finitions, beaucoup de fonctionnalité son présent au détriment de leur stabilité. Les applications restent utilisables mais mériterait quelques patches pour corriger les problèmes. Je suis tout de même très content d’avoir réussi à implémenter certaines fonctionnalités tel que le blocage des élèves.

**Imepro Gouvernon Stan, canton Neuchâtel, CPNE-TI**

# Annexe Manuel utilisateur

## Professeur

Au démarrage l’application essaye de déterminer quelle IP elle doit utiliser mais s’il y en a plusieurs vous aurez une fenêtre qui vous demanderas de choisir. Il faut que vous sélectionniez l’adresse qui est sur le même réseau que vos élèves.

Après le démarrage l’application demande et reçoit des données des élèves toutes les 15 secondes. Vous pouvez voir ces communications dans la ListBox en bas à droite de l’application.

Pour afficher les captures d’écran des élèves vous pouvez sélectionnez les élèves que vous voulez voir dans le TreeView en bas à gauche. Vous pouvez changer la taille des captures d’écran avec le slider en haut à gauche de l’interface.

En haut à gauche vous pouvez voir dans un TreeView tous les élèves connectés, en ouvrant les branches vous pourrez aussi voir les processus lancés par l’élève ainsi que les urls qu’il a visité sur chaque navigateur.

Le TreeView en bas à gauche vous permet aussi d’afficher une interface individuelle pour un élève. Pour cela il vous suffit de double cliquer sur un élève.

Maintenant que l’interface individuel est ouverte vous avez de nouvelles possibilités.

Arrêter l’application de l’élève.

Tuer un process sélectionné

Sauvegarder l’image actuel dans vos téléchargements

Prendre le contrôle de l’élève.

Le bouton tout en haut à gauche vous permet de partager votre écran.

Une nouvelle interface apparait et vous permet de choisir plusieurs options pour votre diffusion :

Les élèves à qui diffuser.

La priorité du Stream (pleine écran, fenêtré, au-dessus, blocage de l’élève).

Ainsi que les applications autorisées pendant la diffusion.

## Élève

En tant qu’élève l’application n’est pas très complexe, elle démarre automatiquement mais reste cachée. Si vous voulez l’ouvrir vous pouvez cliquer sur le TrayIcon en bas à droit de votre écran. Cela vous permettra de lancer des navigateurs web tel que chrome ou Firefox.

Lorsque l’application s’ouvre toute seul vous pourrez y voir soit un Stream de votre professeur soit un message de votre professeur.

Dans le cas d’un Stream l’application peut forcer son affichage au premier plan ou bloquer votre clavier et votre souris. Ne vous inquiétez pas c’est normal c’est le professeur qui le choisi au moment de lancer le Stream.

La seule autre chose que vous pouvez voir dans l’application sont les interactions de votre application avec l’application du professeur.

Au démarrage l’application peut vous demander l’adresse IP de votre professeur si c’est le cas demandez-lui son IP, elle est affichée sur son application. L’adresse que vous entrez sera sauvegardé pour la réessayer la semaine prochaine durant la même demi-journée

Si l’application ne se connecte pas à l’application du professeur c’est surement que l’IP du professeur à changer et que l’IP sauvegardé n’est plus la bonne. Dans ce cas-là vous pouvez pressez sur le bouton « Changer d’IP »

Si l’application ne semble pas recevoir le Stream de votre professeur vos pouvez cliquer sur le bouton « Help » et suivre les instructions données.

# Annexe Journal de travail

Lundi 24.04.2023

Objectifs Planification Afficher/Entrer l’IP Maitre

Durée 10 périodes

Déroulement

7h50-8h20 : Lecture du cahier des charges.

8h20-12h40 : Création de l’échéancier.

13h25-13h50 Affichage de l’adresse IP du professeur.

13h50-15h00 : L’élève doit entrer l’adresse de son professeur.

15h00.16h40 : Fin de la planification.

Problèmes / solutions

Synthèse

J’ai commencé la matinée en lisant le cahier des charges, puis j’ai utilisé le modèle d’échéancier fait avec M. Hêche pour faire ma planification pour le projet. J’ai planifié les livrables ainsi que la durée de chaque tâche. J’ai ensuite changé d’activité pour commencer les modifications au système de connexion. Désormais l’adresse IP du professeur est affichée sur son application et les élèves doivent entrer l’adresse du professeur au démarrage. Je suis ensuite retourné sur ma planification pour la terminer.

Mardi 25.04.2023

Objectifs Mémoriser les adresses Finir livrable Connexion

Durée 10 périodes

Déroulement

7h50-10h00 : Sauver/Charger les adresses des professeurs.

10h00-11h55 : Annonce de fermeture et reconnexion.

12h40-15h00 : Début du TreeView et affichages urls/processus.

15h00-16h40 : Création d’un deuxième arbre pour afficher ou non les captures.

Problèmes / solutions

* Sérialiser les adresses IP.
* Vérifier quelles sont correcte puis les stocker en string.
* Mettre des cases à cocher qu’au premier niveau du TreeView.
* Ne semble pas possible simplement : <http://dotnetfollower.com/wordpress/2011/05/winforms-treeview-hide-checkbox-of-treenode/>
* Un autre TreeView pour la sélection. Le premier pour voir les détails.
* Mettre des boutons dans le TreeView.
* Peut-être : <https://stackoverflow.com/questions/8672735/how-do-i-add-a-button-beside-each-node-of-a-treeview>
* Utiliser les cases à cocher dans le second TreeView.

Synthèse

Après un matin très productif où j’ai terminé les objectifs de la journée c’est-à-dire la sauvegarde des adresses des professeurs ainsi qu’une déconnection et une reconnexion parfaitement fonctionnel, j’ai pu commencer le prochain livrable : La modification complète de l’interface de l’application des professeurs. J’ai commencé par ajouter un TreeView pour voir les détails de tous les élèves et un autre pour sélectionner les captures d’écran que l’on veut voir.

Mercredi 26.04.2023

Objectifs Afficher les miniatures Afficher les informations

Zoomer dans les miniatures

Durée 10 périodes

Déroulement

7h50-8h40 : Choix du professeur de l’IP qu’il veut utiliser.

8h40-11h50 : Affichages des miniatures.

12h40-13h25 : Suite de l’affichage des miniatures

13h25-15h00 : Affichage des informations supplémentaires.

15h00-16h40 : Zoom dans les miniatures

Problèmes / solutions

* Au démarrage l’application professeur prenait la première adresse ipv4.
* Nouvelle interface pour choisir l’IP que l’on veut.
* Les captures d’écran ne s’affiches pas.
* Elles s’affiches partiellement à l’origine.
* Les captures d’écrans ne s’affichent pas complétements.
* La classe qui contient la capture et les labels est aussi un contrôle qu’il faut redimensionner.

Synthèse

Après avoir réalisé un problème avec l’adresse IP du maître que j’ai résolu. J’ai commencé à faire l’affichage des miniatures lorsqu’elles sont checkées dans les cases à cocher. Après de multiples problèmes j’ai fini par avoir un résultat satisfaisant auquel j’ai ajouté les informations supplémentaires en dessous des miniatures. Je me suis ensuite attaqué au zoom qui ne fut pas très compliqué.

Mardi 02.05.2023

Objectifs Finir les miniatures Rapport pour l’expert

Commencer les historiques

Durée 10 périodes

Déroulement

7h50-9h00 : Avancement dans le rapport.

9h00-10h30 : Fin des miniatures.

10h30-10h45 : Démo des TreeView ainsi que des miniatures.

10h45-11h10 : Correction de problèmes mineurs.

11h10-11h50 : Préparation pour la rencontre avec l’expert.

12h40-13h20 : Amélioration de l’historique des communications.

13h20-13h40 : Entretient avec l’expert.

13h40-16h40 : Création de l’historique des urls.

Problèmes / solutions

* Les miniatures sortent de l’écran et on ne peut pas scroller.
* Placer les miniatures dans un panel pour pouvoir scroller si les miniatures sont trop grandes.
* Le système en place ne permettait pas un historique des urls.
* Création d’une nouvelle classe pour résoudre le problème.

Synthèse

J’ai commencé par préparer la rencontre avec l’expert en faisant de l’avance dans mon rapport qui était jusqu’à la presque vide. J’ai ensuite voulu finir les miniatures pour pouvoir faire ma seconde démonstration sur les TreeView et les miniatures. Après avoir relevé quelques problèmes légers durant la démonstration je les ais corrigés. (Faire plus de place pour les miniatures/pouvoir scroller dans les miniatures) J’ai ensuite finalisé ma rencontre avec l’expert en préparent une synthèse du travail accompli. Après la pause j’ai eu le temps de commencer le prochain livrable en améliorant l’historique en directe des communications. J’ai ensuite fait mon entretien avec l’expert puis j’ai continué le livrable en créant un historique pour les urls.

Mercredi 03.05.2023

Objectifs Finaliser l’historique des urls Affichage individuel

Sauvegarde des ScreenShot Envoi de messages

Listes de filtres

Durée 10 périodes

Déroulement

7h50-8h20 : Fin de l’historiques des urls.

8h20-9h00 : Déplacement de la logique du client dans la classe.

9h00-9h45 : Affichage individuel d’un élève.

10h05-11h50 : Sauvegarde des captures d’écran

12h40-14h20 : Envoi de messages du professeur

14h20-14h35 : Troisième démonstration.

14h35-15h55 : Afficher différemment les processus grâce aux listes.

15h55-16h40 : Documentation

Problèmes / solutions

* Prise de control à distance.
* Utiliser le Hook pour détourner la souris et le clavier.
* L’id de l’élève n’était pas sauvegardé pendant l’update.
* Ajout de la sauvegarde. (Tout fonctionnait quand même je ne sais pourquoi.)
* Le système d’affichage individuel n’était plus tout à fait compatible avec le reste.
* Création de l’affichage individuel lors du double clic sur un Node ordinateur dans les TreeView.
* Adapter l’affichage individuel pour qu’il puisse lui aussi afficher l’historique des urls.
* Retirer les listbox de la classe Student
* Ne semble pas possible car les binding List de fonctionnent pas sur plusieurs threads : <https://stackoverflow.com/questions/4823481/bindinglist-not-updating-bound-listbox>

Synthèse

J’ai commencé la journée en finnisant l’historique des urls, puis après une discussion avec mon professeur nous avons décidé de déplacer toute la logique du client dans la classe au lieu du formulaire. J’ai ensuite continué mes taches en adaptant le système d’affichage individuel à la nouvelle interface du professeur. Pour finir la matinée j’ai implémenté la sauvegarde des captures d’écran à deux endroits : dans les miniatures et dans l’affichage individuel. La fin du livrable est simple il a suffi de permettre au professeur d’envoyer des messages aux élèves. Après avoir fait la démonstration et avoir discuté de la futilité d’envoyer les listes de filtres aux élèves, j’ai commencé à coder les listes de filtre chez le professeur. Puis j’ai terminé par un période de documentation.

Lundi 08.05.2023

Objectifs Filtre pour les urls Affichage redimensionnable

Durée 10 périodes

Déroulement

7h50-8h35 : Ajout d’un filtre pour les urls.

8h35-9h25 : Faire fonctionner l’affichage individuel avec les filtres.

9h25-9h40 : Création de la classe de configuration.

9h40-11h20 : Création d’un affichage redimensionnable.

11h20-11h50 : Activer ou non les filtres.

12h40-16h40 : Conversion du projet client en WPF.

Problèmes / solutions

* Accéder aux variables de configuration dans plusieurs formulaires.
* Création d’une classe statique de configuration.
* Les Interfaces ne sont pas redimensionnables.
* Utilisation massive des SplitContainer.
* Les deux projets n’utilisent pas la même version de .NET.
* Ne pas référencer la bibliothèque de classe.
* Ajouter la classe en lien.

Synthèse

J’ai commencé par coder la liste de filtre pour les urls. Puis il a fallu faire fonctionner l’interface individuel avec tous les filtres. Puis le problème des variables de configuration c’est posé. Pour le résoudre j’ai créé une classe statique de configuration. J’ai ensuite réglé un problème évoqué au dernier livrable : les interfaces ne sont pas redimensionnables. J’ai ensuite passé toute l’après-midi à essayer faire fonctionner le projet WPF.

Mardi 09.05.2023

Objectifs Bloquer les sites non autorisés Activer/Désactiver filtres

Intégration de Navlogeur Priorité du Stream

Durée 10 périodes

Déroulement

7h50-8h35 : Envoi de la liste des sites autorisés.

8h35-10h00 : Blocage des autres sites.

10h00-11h25 : Activer ou non les filtres.

11h25-11h50 : Démo filtrage.

12h40-13h30 : Permettre la sélection des élèves pour la diffusion.

13h30-15h00 : Test de l’utilisation de sélénium.

15h00-16h40 : Début de la priorité du Stream.

Problèmes / solutions

* Abandon du projet en WPF
* Utilisation de WebView2 dans Windows Form.
* Utilisation de Sélénium.
* Permettre la suppression d’une url dans le TreeView.
* Bloquer l’utilisateur.

Synthèse

Après une discussion avec mon supérieur professionnel l’intégration du projet Windows Form en WPF a été abandonné. Pour la remplacer j’ai utilisé WebView2 dans du Windows Form pour simuler un navigateur. J’ai donc pu ensuite envoyer la liste des sites à bloquer à tous les élèves. J’ai ensuite fait la démo pour le quatrième rendu. Puis en début d’après-midi j’ai permis au professeur de choisir les élèves qu’il veut pour la diffusion de son écran. Pour avoir un navigateur intégrer au programme j’ai utilisé sélénium qui semble fonctionner correctement. Pour finir j’ai commencé le choix de la priorité du Stream pour le professeur.

Mercredi 10.05.2023

Objectifs Blocage de la souris blocage du clavier

Focus autorisé

Durée 10 périodes

Déroulement

7h50-10h00 : Blocage de la souris.

10h00-15h00 : Blocage du clavier.

15h00-16h40 : Choix du focus autorisé.

Problèmes / solutions

* Bloquer le clavier.
* Utiliser un Hook sur les touches pour les annuler. <https://www.codeproject.com/Articles/19004/A-Simple-C-Global-Low-Level-Keyboard-Hook>
* La technique du Hook échouait à cause d’un delegate.
* Stocker le delegate au lieu de le recréer.
* Après un CTRL+ALT+DEL l’élève reprenait le contrôle de sa souris.
* Désactivation de la souris toutes les secondes.

Synthèse

La première partie de la matinée a été dédiée à empêcher l’élève d’avoir le contrôle durant le Stream du professeur. Tout le reste de la journée à servi à empêcher l’élève d’utiliser le clavier durant le Stream du professeur. J’ai tout de même pu durant les deux dernières périodes commencer le choix du focus autorisé durant le Stream.

Lundi 15.05.2023

Objectifs Finir la priorité du Stream Focus pendant le Stream

Documentation

Durée 10 périodes

Déroulement

7h50-8h30 : Le choix de la priorité est terminé.

8h30-11h55 : Choix du focus autorisé pendant le Stream.

12h40-14h40 : Choix du focus autorisé pendant le Stream.

14h40-16h40 : Documentation.

Problèmes / solutions

* Trouver une fonction que minimise les fenêtres.
* La fonction ShowWindow dans user32.dll est faite pour cela.
* Empêcher l’utilisation de certaines applications durant le Stream sans les fermer.
* Toutes les secondes je minimise les fenêtres non autorisées.

Synthèse

Après avoir peaufiné la priorité du Stream je me suis lancé dans le focus. Il m’a causé bien des problèmes mais la solution était finalement simple. J’ai dû trouver une fonction qui me permettait de minimiser n’importe quelle fenêtre puis ne minimiser que les fenêtres interdites pendant le Stream. Finalement j’ai fait de la documentation.

Mardi 16.05.2023

Objectifs Streaming de l’élève Documentation

Durée 10 périodes

Déroulement

7h50-11h00 : Streaming de l’écran de l’élève sur l’application professeur.

11h00-11h50 : Documentation.

12h40-16h40 : Documentation.

Problèmes / solutions

* Le streaming de l’élève n’est pas stable.
* Documentation en retard
* J’ai passé du temps dessus.

Synthèse

Après avoir commencé le streaming de l’élève et être arrivé à une solution fonctionnelle mais instable j’ai décidé de faire de la documentation. J’espère la terminer demain dans la matinée pour essayer ensuite de finir le contrôle à distance.

Mercredi 17.05.2023

Objectifs Documentation

Durée 10 périodes

Déroulement

7h50-11h50 : Documentation.

12h40-16h40 : Documentation.

Synthèse

J’ai dû finir mon rapport pour le rendre pour cela j’ai dû faire mon WebSummary ainsi que le manuel utilisateur. Pour finir j’ai créé mon document de code.

# Annexe Planification

