Imepro

Application de monitoring et d’aide des élèves

École : CIFOM, École technique

Atelier : Salle 207

Langage : C#

Auteur : Stan Gouvernon

Supérieur : Alain Jeanmaire

Date : 24.01.2023-22.03.2023

Table des matières

[Présentation du projet 2](#_Toc133418731)

[Prérequis 2](#_Toc133418732)

[Matériel et logiciels 2](#_Toc133418733)

[Connaissances requises 2](#_Toc133418734)

[Cahier des charges 2](#_Toc133418735)

[But du projet 2](#_Toc133418736)

[Maquettes des applications 2](#_Toc133418737)

[Fonctionnalités prévues 2](#_Toc133418738)

[Client 2](#_Toc133418739)

[Maître 2](#_Toc133418740)

[Communication 2](#_Toc133418741)

[Schéma de fonctionnement 2](#_Toc133418742)

[Schéma de classe 2](#_Toc133418743)

[Gestion du projet 2](#_Toc133418744)

[Mise en place 2](#_Toc133418745)

[Matériel 2](#_Toc133418746)

[Logiciel 2](#_Toc133418747)

[Application cliente 2](#_Toc133418748)

[Interface 2](#_Toc133418749)

[Démarrage 2](#_Toc133418750)

[Trouver les processus lancés par défaut 2](#_Toc133418751)

[Trouver les processus lancés par l’utilisateur 2](#_Toc133418752)

[Arrêter les processus lancés par l’utilisateur 2](#_Toc133418753)

[Capture d’écran des deux moniteurs 2](#_Toc133418754)

[Récupération des urls ouverts 2](#_Toc133418755)

[ClientData 2](#_Toc133418756)

[Recevoir les demandes 2](#_Toc133418757)

[Arrêt 2](#_Toc133418758)

[Classe ClientForm 2](#_Toc133418759)

[Application Maître 2](#_Toc133418760)

[Interface 3](#_Toc133418761)

[Démarrage 3](#_Toc133418762)

[Résumé 3](#_Toc133418763)

[Envoi de requêtes 3](#_Toc133418764)

[Affichage des données client 3](#_Toc133418765)

[Arrêt 3](#_Toc133418766)

[Classe MaitreForm 3](#_Toc133418767)

[Liaisons entre les deux applications 3](#_Toc133418768)

[Connexion 3](#_Toc133418769)

[Socket 3](#_Toc133418770)

[Données 3](#_Toc133418771)

[Images 3](#_Toc133418772)

[Flux d’image 3](#_Toc133418773)

[Demande d’arrêt d’un processus 3](#_Toc133418774)

[Tests 3](#_Toc133418775)

[Méthode de test 3](#_Toc133418776)

[Problèmes et solutions 3](#_Toc133418777)

[Références 3](#_Toc133418778)

[Webographie 3](#_Toc133418779)

[Glossaire 3](#_Toc133418780)

[Bilan 3](#_Toc133418781)

[Améliorations possibles 3](#_Toc133418782)

[Objectifs atteints 3](#_Toc133418783)

[Client 3](#_Toc133418784)

[Maître 3](#_Toc133418785)

[Communication 3](#_Toc133418786)

[Conclusion 3](#_Toc133418787)

# 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

Connaissance en développement orienté objet en C#

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

Connaissance sur le fonctionnement des applications.

## Cahier des charges

## But du projet

## Maquettes des applications

## Fonctionnalités prévues

### Client

### Maître

### Communication

## Schéma de fonctionnement

## Schéma de classe

# 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.

# 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.

# Application Élève

# Application Professeur

# Liaisons entre les deux applications

# Tests

## Méthode de test

Les tests ont été effectué de deux manières différentes : par des tests unitaires et par des tests visuels. 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. C’est pourquoi dans ces cas-là j’ai préféré faire des tests visuels. Qui sont plus redondant mais aussi plus fiables dans certains cas.

# Problèmes et solutions

## TreeView customisé

Afficher des casses 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.

## 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 étés 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 client Windows Forms en projet WPF. Cela s’est avéré très difficile car les versions utilisé 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 à 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 client en WPF

## WebView2 n’est pas complet

Pour avoir un navigateur intégré à mon client 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... Au final 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.

# 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/>

## Glossaire

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

# Bilan

## Améliorations possibles

## Objectifs atteints

### Client

### Maître

### Communication

## Conclusion